# A PROJECT REPORT ON

**FOOD DEMAND FORECASTING**

***Mini project submitted in partial fulfillment of the requirements for the***

***award of the degree of***

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY (2017-2021)

**BY**

Korlepara PavanBalu 17241A1289

Saripelly SaiKrishna 17241A12A9

Telange Kapil 17241A12B2

Chalasani SaiSurya 17241A1274

***Under the Esteemed guidance off***

Dr. N.V Ganapathi Raju

**Professor**, **Dept. of IT**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

**GOKARAJU RANGARAJU INSTITUTE OF ENGINEERING AND TECHNOLOGY**

**(AUTONOMOUS)**

**HYDERABAD 500090**



# CERTIFICATE

This is to certify that it is a bonafide record of Mini Project work entitled **“Food demand forecasting”** done by **korlepara PavanBalu(17241A1289), Saripelly SaiKrishna(17241A12A9), Telangi Kapil(17241A12B2)**, **Chalasani** **SaiSurya(17241A1274)** students of **B.Tech (IT)** in the Department of Information Technology, Gokaraju Rangaraju Institute of Engineering and Technology during the period 2017-2021 in the partial fulfillment of the requirements for the award of degree of **BACHELOR OF TECHNOLOGY IN INFORMATION TECHNOLOGY** from JNTU, Hyderabad.

Dr. N.V Ganapathi Raju Dr. K. Prasanna Lakshmi

(Internal Project Guide) (Head of Department)

(Project External)

# ACKNOWLEDGEMENT

We take the immense pleasure in expressing gratitude to our internal guide Dr. N.V Ganapathi Raju, Professor, Information Technology, GRIET. We express our sincere thanks for his/her encouragement, suggestions and support, which provided the impetus and paved the way for the successful completion of the project work.

We wish to express our gratitude to **Dr. K. Prasanna Lakshmi**, **P. Gopala Krishna, K.Archana** and **Y. Prashanthi**, our Project Co-coordinators, for their constant support in the course of the project.

We express our sincere thanks to **Dr. Jandhyala N Murthy,** Director, GRIET**,** and **Dr. J. Praveen,** Principal, GRIET**,** for providing us the conductive environment for carrying through our academic schedules and project with ease.

We also take this opportunity to convey our sincere thanks to the teaching and non-teaching staff of GRIET College, Hyderabad.

Email: Pavanbalu.Korlepara@gmail.com Email:saikrishnasaripelly98@gmail.com

Contact No: 9490789935 Contact No: 9182390164

Emailkapilsai040@gmail.com Email: Saisuryachalasani196@gmail.com

Contact No: 8074008852 Contact No: 9949451621

# DECLARATION

This is to certify that the project entitled “**Food Demand Forecasting**” is a bonafied work done by us in partial fulfillment of the requirements for the award of the degree **BACHELOR OF ENGINEERING IN INFORMATION TECHNOLOGY** from **Jawaharlal Nehru Technological University, Hyderabad.**

We also declare that this project is a result of our own effort and has not been copied or imitated from any source. Citations from any websites, books and paper publications are mentioned in the Bibliography.

This work was not submitted earlier at any other University or Institute for the award of any degree.

1. 17241A1289   PAVANBALU K

2. 17241A12A9   SAIKRISHNA S

3. 17241A12B2    KAPIL T

4. 17241A1274     SAISURYA CH

**CONTENTS**

**TITLE**

Certificate

Acknowledgement

Declaration

**ABSTRACT**

1. **INTRODUCTION**
   1. Introduction to Project
   2. Project Overview
      1. Existing System
      2. Proposed System
2. LITERATURE SURVEY
3. REQUIREMENTS ENGINEERING
4. SYSTEM REQUIREMENTS
   1. Software Requirements
5. SYSTEM DESIGN
   1. UML Diagrams
   2. Use Case Diagram
   3. Class Diagram
   4. Sequence Diagram
   5. Activity Diagram
   6. Collaboration Diagram
6. IMPLEMENTATION
   1. Source Code
   2. Training Modules
7. RESULTS
8. CONCLUSION AND FUTURE ENHANCEMENTS
9. BIBLIOGRAPHY

**ABSTRACT**

Demand forecasting is a key component to every growing online business. Without proper demand forecasting processes in place, it can be nearly impossible to have the right amount of stock on hand at any given time.

A food delivery service has to deal with a lot of perishable raw materials which makes it all the more important for such a company to accurately forecast daily and weekly demand.

In economic terms demand can be defined as the quantity of a product or service which people are willing and able to buy during a given period of time.

In consequence, demand forecasting is the art of predicting the level of demand which might occur at some future point or period of time.

The strengths, weaknesses, and limitations of the principal forecasting methods and their applicability in the field of tourism and travel are described.

Sections deal with the need for forecasting, the problems faced by forecasters and forecasting as an aid to management decision making.

**1.Introduction**

**1.1Introduction to project**

Our project aim is food demand forecasting, where we will predict the demand of food for future period. Without proper demand forecasting processes in place, it can be nearly impossible to have the right amount of stock on hand at any given time. A food delivery service has to deal with a lot of perishable raw materials which makes it all the more important for such a company 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 our competitors. In this project, we get a taste of demand forecasting challenge using a real dataset.

**1.2Project Overview**

Our client is a meal delivery company which operates in multiple cities. They have various fulfillment centers in these cities for dispatching meal orders to their customers. The client wants us to help these centers with demand forecasting for upcoming weeks so that these centers will plan the stock of raw materials accordingly.

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. Secondly, staffing of the centers is also one area wherein accurate demand forecasts are really helpful. Given the following information, the task is to predict the demand for the next 10 weeks (Weeks: 146-155) for the center-meal combinations in the test set:

* Historical data of demand for a product-center combination (Weeks: 1 to 145)
* Product(Meal) features such as category, sub-category, current price and discount
* Information for fulfillment center like center area, city information etc.

**1.2.1 Existing system**

Currently, there are only few approaches in prediciting the food demand.The client may go with the expert opinions, future prices or climate influencing forecasts despite having different underlying assumptions and mechanisms. The existing system gives a brief idea to the meal delivery companies to predict the food demand, but those are not accurate results. However, reporting of model accuracy and uncertainty was uncommon, leading to difficulties in making evidence-based decisions about which forecasts to trust.

**1.2.2 Proposed system**

Here we present a systematic review of the food demand literature—including a meta-analysis of papers reporting average global food demand predictions—and test the effect of model complexity on predictions. After performing different experiments with features, data format,algorithms, parameters,etc. The ensemble of tuned lightGBM and XGBoost performs better than others(with least errors and better solution).

# 2.LITERATURE SURVEY

A literature survey or a literature review in a project report is that section which shows the various analyses and research made in the field of your interest and the results already published, taking into account the various parameters of the project and the extent of the project.

**2.1Data Science**

* Data science is a multidisciplinary blend of **data inference, algorithm development, and technology** in order to solve analytically complex problems.
* This aspect of data science is all about uncovering findings from data. Diving in at a granular level to mine and understand complex behaviours, trends, and inferences. It's about surfacing hidden insight that can help enable companies to make smarter business decisions. For example:
* Netflix data mines movie viewing patterns to understand what drives user interest and uses that to make decisions on which Netflix original series to produce.
* Target identifies what are major customer segments within its base and the unique shopping behaviours within those segments, which helps to guide messaging to different market audiences.
* Proctor & Gamble utilizes time series models to more clearly understand future demand, which help plan for production levels more optimally.

**2.2Machine Learning (ML)**

Machine learning (ML) is a category of algorithm that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output while updating outputs as new data becomes available.

* Machine learning is of two types
* Supervised learning
* Unsupervised learning

**2.2.1Supervised Learning**

In supervised learning, we are given a data set and already know what our correct output should look like, having the idea that there is a relationship between the input and the output. Supervised learning problem further categorized into regression and classification problems.

#### 2.2.1.1 Regression

In regression problem, we are trying to predict results within a continuous output, meaning that we are trying to map input variables to some continuous function. To exemplify, given data about the size of houses on the real estate market, try to predict their price. Another example would be, given a picture of a person, we must predict their age or gender.

#### 2.2.1.2 Classification

Classification, on the other hand, is finding the category of the input variable, or in more academic terms, mapping input variables into discrete categories. Ideal sentence to find a classification problem would be, whetherthisorthat, like, yes or no, 0 or 1, true or false. For example, from the example of house price given above, if we change the output to “Sells for more or less than asking price,” then it is a classification problem. Another example is, given a patient with tumor, we must predict whetherthe tumor is malignant or benign.

**2.2.2Unsupervised Learning**

On the contrary to Supervised learning, unsupervised learning allows us to approach problems with little or no idea what our results should look like. We can derive structure from data where we don’t necessarily know the effect of the variables.

We can derive this structure by clustering the data based on relationships among the variables in the data. With Unsupervised learning there is no feedback based on the prediction results. For example, take a collection of 1,000,000 different genes, and find a way to automatically group these genes into groups that are somehow similar or related by different variables, such as lifespan, location, roles, and so on. This is a good example of clustering. Whereas, for a non-clustering problem such as “Cocktail Party Problem”, it helps in identifying voices music from a mesh of sounds at a cocktail party.

**3.REQUIREMENTS ENGINEERING**

Requirement Engineering is the process of defining, documenting and maintaining the requirements. It is a process of gathering and defining service provided by the system. Requirements Engineering Process consists of the following main activities: Requirements elicitation

**3.1Hardware Requirements:**

64 bit desktop processor

wifi capable device

**3.2Software Requirements:**

windows 10/linux OS

python interpreter

**4.SYSTEM REQUIREMENTS:**

## System requirements are the required specifications a device must have in order to use certain [hardware](https://techterms.com/definition/hardware) or [software](https://techterms.com/definition/software). For example, a computer may require a specific I/O [port](https://techterms.com/definition/port) to work with a [peripheral device](https://techterms.com/definition/peripheral). A smart phone may need a specific [operating system](https://techterms.com/definition/operating_system) to run a particular [app](https://techterms.com/definition/app).

## **4.1 SOFTWARE REQUIREMENTS:**

Our project requires various software and its components.

**4.1.1 Jupyter Notebook**

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

**4.1.2 Python**

Python is an interpreter, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Van Rossum led the language community until stepping down as leader in July 2018.

**4.1.3 Machine Learning Packages in built in Python**

**4.1.3.1 Pandas:**

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

**4.1.3.2 SciPy**

**SciPy** is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

**4.1.3.3 Numpy:**

NumPy is a Python package which stands for 'Numerical Python'. It is the core library for scientific computing, which contains a powerful n-dimensional array object, provide tools for integrating C, C++ etc.

**4.1.3.4 Matplotlib:**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

**4.1.3.5 Seaborn:**

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures. Here is some of the functionality that seaborn offers: A dataset-oriented API for examining relationships between multiple variables.

**4.1.3.6 LightGBM:**

LightGBM is a gradient boosting framework that uses tree based learning algorithms. It is designed to be distributed and efficient with the following advantages: ... Support of parallel and GPU learning. Capable of handling large-scale data.

**4.1.3.7 XGBoost:**

It is an implementation of gradient boosting machines created by Tianqi Chen, now with contributions from many developers. It belongs to a broader collection of tools under the umbrella of the Distributed Machine Learning Community or DMLC who are also the creators of the popular mxnet deep learning library.

**4.1.3.8 Warnings:**

Warning control in Python Programs. It displays certain message but program continues. Warnings are issued to alert the user of certain conditions which aren't exactly exceptions. Typically warning appears if some deprecated usage of certain programming element like keyword/function/class etc. is found.Jan 17, 2019

**4.1.4 Sklearn:**

**4.1.4.1.sklearn.preprocessing:**

Preprocessing data. The sklearn. preprocessing package provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators. In general, learning algorithms benefit from standardization of the data set

from sklearn.preprocessing import LabelEncoder

**4.1.4.2.sklearn.model\_selection:**

Quick utility that wraps input validation and next(ShuffleSplit(). split(X, y)) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

from sklearn.model\_selection import train\_test\_split, GridSearchCV

**4.1.4.3.sklearn.ensemble:**

Ensemble learning uses multiple machine learning models to try to make better predictions on a dataset. An ensemble model works by training different models on a dataset and having each model make predictions individually.

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import AdaBoostRegressor

**4.1.4.4.sklearn.tree:**

A decision tree is a flowchart-like tree structure where an internal node represents feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome. The topmost node in a decision tree is known as the root node. It learns to partition on the basis of the attribute value.

from sklearn.tree import DecisionTreeRegressor

**4.1.4.5.sklearn.neighbors:**

sklearn.neighbors provides functionality for unsupervised and supervised neighbors- based learning methods. Unsupervised nearest neighbors is the foundation of many other learning methods, notably manifold learning and spectral clustering. Supervised neighbors-based learning comes in two flavors: classification for data with discrete labels, and regression for data with continuous labels

from sklearn.neighbors import KNeighborsRegressor

**5.SYSTEM DESIGN**

As the strategic value of software increases for many companies, the industry looks for techniques to automate the production of software and to improve quality and reduce cost and time-to-market. These techniques include component technology, visual programming, patterns and frameworks. Businesses also seek techniques to manage the complexity of systems as they increase in scope and scale. In particular, they recognize the need to solve recurring architectural problems, such as physical distribution, concurrency, replication, security, load balancing and fault tolerance. The Unified Modelling Language (UML) was designed to respond to these needs.

**5.1Introduction to UML**

UML (Unified Modeling Language) is a modeling language used by software developers. UML can be used to develop diagrams and provide users with ready-to-use, expressive modeling examples. UML can be used for modeling a system independent of a platform language. UML is a graphical language for visualizing, specifying, constructing, and documenting information about software-intensive systems. UML gives a standard way to write a system model, covering conceptual ideas.

The following UML diagrams are created to better understand the project

* Use Case Diagram
* Class Diagram
* Activity Diagram
* Sequence Diagram
* Collaboration Diagram

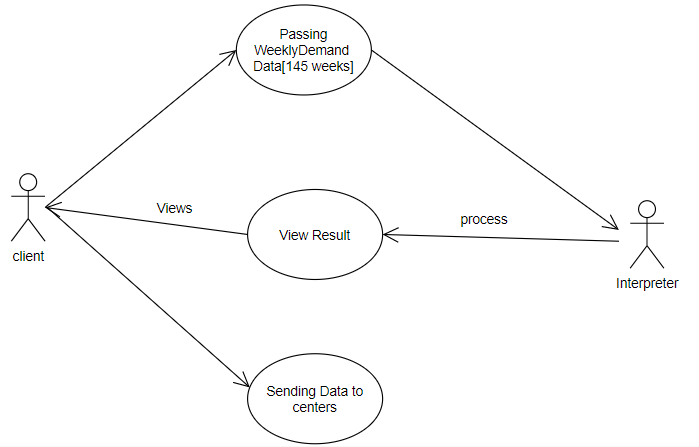
**5.2 Usecase Diagram**

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.

Only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML, there are five diagrams available to model the dynamic nature and use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature, there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. Use case diagrams consists of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

Hence to model the entire system, a number of use case diagrams are used.

****

* The above use case diagram has 2 actors and 3 uses cases.

Actors: - Client and Interpreter

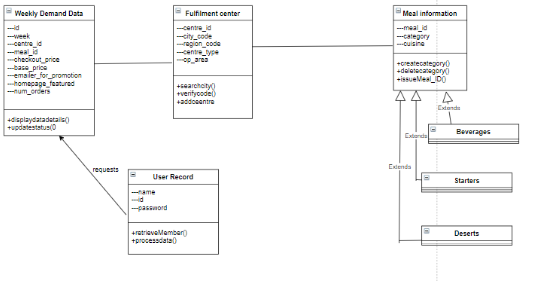
Use cases: - Passing weekly demand data, view result and sending data to centers

**5.3 Class Diagram**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

****

* The class diagram contains four classes.

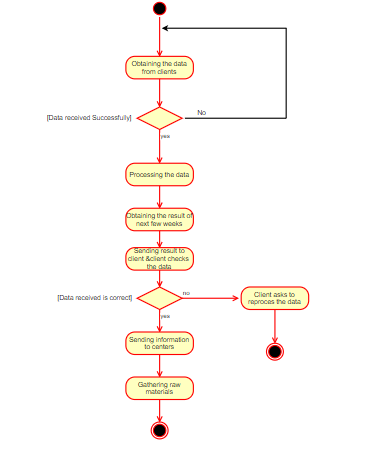
Class: -weekly demand data, fulfilment center, meal information and user record

**5.4 Activity Diagram**

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

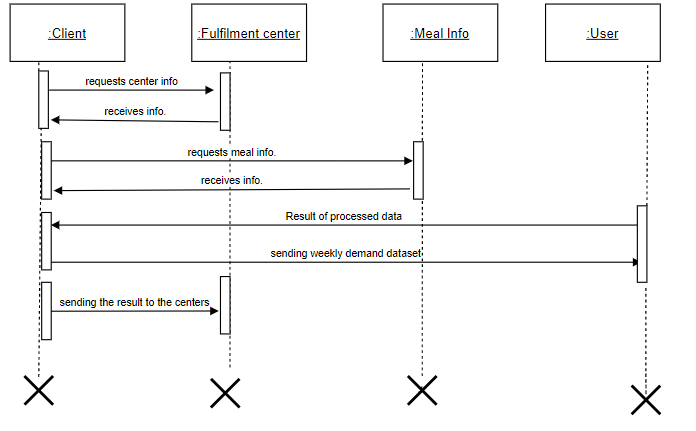
****

* An activity diagram consists of activity states and action states, transactions, objects.
* It starts with initial state and ends with final state symbols.

**5.5 Sequence Diagram**

UML Sequence diagrams are interaction diagrams that detail how operations are carried out. As sequence diagrams can be used to capture the interaction between objects in the context of a collaboration, one of the primary uses of sequence diagrams is in the transition from requirements expressed as use cases to the next and more formal level of refinement. Use cases are often refined into one or more sequence diagrams.

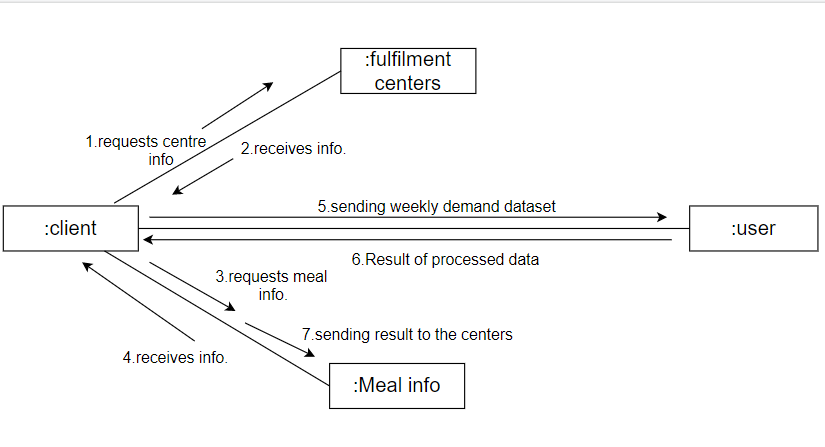
Sequence diagrams are time focus and they show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.

****

* The sequence diagram has four objects and various interactions (messages) between them.
* Activation boxes represents the time an object needs to complete a task

**5.6 Collaboration Diagram**

Collaboration diagrams are nothing different from sequence diagrams. Sequence diagrams show the flow of the system whereas the Collaboration diagrams show how objects interact with each other in the system. They are used to depict the object behavior of the system. Collaboration diagrams are also known as communication diagrams.

****

* Collaboration diagram is also called as communication diagram or interaction diagram.
* It is an illustration of the relationships and interactions among software objects.

**6.IMPLEMENTATION**

Implementationisthe process that turns strategies and plans into actions in order to accomplish strategic objectives and goals*.* Implementing your strategic plan is as important, or even more important, than your strategy. Implementation simply means carrying out the activities described in your work plan

**6.1SOURCE CODE:**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**import** **seaborn** **as** **sns**

**from** **lightgbm** **import** LGBMRegressor, plot\_importance

**from** **xgboost** **import** XGBRegressor

**from** **sklearn.model\_selection** **import** train\_test\_split, GridSearchCV

**from** **sklearn.preprocessing** **import** LabelEncoder

**from** **sklearn.metrics** **import** mean\_squared\_log\_error, mean\_squared\_error

**import** **warnings**

warnings.filterwarnings("ignore")

from sklearn import metrics

from sklearn.ensemble import RandomForestRegressor

from sklearn.ensemble import AdaBoostRegressor

**from sklearn.tree import DecisionTreeRegressor**

**from sklearn.neighbors import KNeighborsRegressor**

from sklearn.metrics import r2\_score

**Data Analysis**

### Analysing MeaL\_Info

meal\_info = pd.read\_csv("meal\_info.csv")

meal\_info.head()

meal\_info.info()

meal\_info.category.value\_counts()

meal\_info.cuisine.value\_counts()

**Analysing Fulfilment­\_Center\_Info**

fulfilment\_center\_info = pd.read\_csv("fulfilment\_center\_info.csv") fulfilment\_center\_info.head()

fulfilment\_center\_info.city\_code.value\_counts()

fulfilment\_center\_info.info()

fulfilment\_center\_info.region\_code.value\_counts()

fulfilment\_center\_info.center\_type.value\_counts()

fulfilment\_center\_info.op\_area.describe()

### Analysing Train­ Data

train\_data = pd.read\_csv("train.csv")

train\_data.head()

train\_data.info()

train\_data.week.value\_counts()

train\_data.center\_id.value\_counts()

train\_data.meal\_id.value\_counts()

train\_data.checkout\_price.describe()

train\_data.base\_price.describe()

train\_data.sort\_values('checkout\_price').head()

train\_data.num\_orders.describe()

train\_data.sort\_values('num\_orders').tail()

train\_data.emailer\_for\_promotion.value\_counts()

train\_data.homepage\_featured.value\_counts()

### Analysing test data

test\_data = pd.read\_csv("test.csv")

test\_data.head()  
test\_data.info()  
test\_data.week.value\_counts()  
test\_data.meal\_id.value\_counts()

test\_data.center\_id.value\_counts()

test\_data.checkout\_price.describe()

test\_data.base\_price.describe()

test\_data.emailer\_for\_promotion.value\_counts()

test\_data.homepage\_featured.value\_counts()

sns.lineplot(train['checkout\_price'],train['num\_orders'])

**Visualisation**

sns.lineplot(train\_data['checkout\_price'],train\_data['num\_orders'])

sns.boxplot(x='num\_orders',data=train\_data)

sns.jointplot(x='checkout\_price',y='num\_orders',data=train\_data)

# Data Preprocessing

#### Combining train and test data

train\_data\_without\_target = train\_data[train\_data.columns[train\_data.columns != 'num\_orders'].values]

total\_data = train\_data\_without\_target.append(test\_data, sort=**False**)

total\_data.tail()

#### Merge dataset total\_data = total\_data.merge(fulfilment\_center\_info, on='center\_id', how='left') total\_data.head() total\_data = total\_data.merge(meal\_info, on='meal\_id', how='left')

total\_data.head()

total\_data.info()

#### Derive new variable

meal\_base\_price = total\_data[['week', 'center\_id', 'meal\_id', 'base\_price']]

meal\_base\_price = meal\_base\_price.set\_index(['meal\_id', 'center\_id', 'week'])

meal\_base\_price = meal\_base\_price.sort\_index()

meal\_base\_price.head()

meal\_per\_center = train\_data[['week', 'center\_id', 'meal\_id', 'num\_orders']]

meal\_per\_center = meal\_per\_center.set\_index(['meal\_id', 'center\_id', 'week'])

meal\_per\_center = meal\_per\_center.sort\_index()

meal\_per\_center.head()  
meal\_across\_center = train\_data[['week', 'meal\_id', 'num\_orders']] meal\_across\_center = meal\_across\_center.set\_index(['meal\_id', 'week']) meal\_across\_center = meal\_across\_center.sort\_index() meal\_across\_center.head()

def average\_orders(row):

if (row.meal\_id, row.center\_id) in meal\_base\_price.index:

row['mean\_base\_price'] = meal\_base\_price.loc[(row.meal\_id, row.center\_id)].loc[:row.week].base\_price.mean()

else:

row['mean\_base\_price'] = row['base\_price']

week\_adj = row.week - 10

if (row.meal\_id, row.center\_id) in meal\_per\_center.index:

history = meal\_per\_center.loc[(row.meal\_id, row.center\_id)]

row['average\_orders\_13week'] = history.loc[(row.week-13):row.week].num\_orders.mean()

row['average\_orders\_26week'] = history.loc[(row.week-26):row.week].num\_orders.mean()

row['average\_orders\_52week'] = history.loc[(row.week-52):row.week].num\_orders.mean()

row['average\_orders\_13week\_adj'] = history.loc[(week\_adj-13):week\_adj].num\_orders.mean()

row['average\_orders\_26week\_adj'] = history.loc[(week\_adj-26):week\_adj].num\_orders.mean()

else:

row['average\_orders\_13week'] = 0

row['average\_orders\_26week'] = 0

row['average\_orders\_52week'] = 0

row['average\_orders\_13week\_adj'] = 0

row['average\_orders\_26week\_adj'] = 0

if row.meal\_id in meal\_across\_center.index:

history\_across = meal\_across\_center.loc[row.meal\_id]

row['average\_orders\_13week\_across'] = history\_across.loc[(row.week-13):row.week].num\_orders.mean()

row['average\_orders\_26week\_across'] = history\_across.loc[(row.week-26):row.week].num\_orders.mean()

row['average\_orders\_52week\_across'] = history\_across.loc[(row.week-52):row.week].num\_orders.mean()

row['average\_orders\_13week\_adj\_across'] = history\_across.loc[(week\_adj-13):week\_adj].num\_orders.mean()

row['average\_orders\_26week\_adj\_across'] = history\_across.loc[(week\_adj-26):week\_adj].num\_orders.mean()

else:

row['average\_orders\_13week\_across'] = 0

row['average\_orders\_26week\_across'] = 0

row['average\_orders\_52week\_across'] = 0

row['average\_orders\_13week\_adj\_across'] = 0

row['average\_orders\_26week\_adj\_across'] = 0

return row

total\_data = total\_data.apply(average\_orders, axis=1)

total\_data['average\_orders\_13week'].describe()  
total\_data['discount'] = total\_data['mean\_base\_price'] - total\_data['checkout\_price'] total\_data['discount'] = total\_data['discount'] / total\_data['mean\_base\_price'] total\_data.discount.describe()

total\_data['year'] = (((total\_data['week'] - 1)/52) + 1).astype('int')

total\_data.year.value\_counts()

total\_data['month'] = (((total\_data['week'] - 1)/4).astype('int') % 13) + 1

total\_data.month.value\_counts()

total\_data['quarter'] = (((total\_data['week'] - 1)/13).astype('int') % 4) + 1

total\_data.quarter.value\_counts()  
total\_data['week\_in\_month'] = (((total\_data['week'] - 1) % 4) + 1) total\_data.week\_in\_month.value\_counts()

#### Splitting total data into train and test

train\_data = train\_data[['id', 'num\_orders']].merge(total\_data, on='id', how='left')

train\_data.head()

training\_data.info()

train\_data.to\_csv('train\_feature.csv', index=**False**)

test\_data = test\_data[['id']].merge(total\_data, on='id', how='left')

test\_data.head()

test\_data.info()

test\_data.to\_csv('test\_feature.csv', index=**False**)

## **Defining constants**

target = 'num\_orders'

features = ['center\_id', 'meal\_id', 'checkout\_price', 'mean\_base\_price', 'discount', 'emailer\_for\_promotion',

'homepage\_featured', 'city\_code', 'center\_type', 'category', 'year', 'region\_code', 'month',

'week\_in\_month', 'cuisine', 'average\_orders\_26week\_adj', 'average\_orders\_52week',

'average\_orders\_26week', 'average\_orders\_26week\_adj\_across', 'average\_orders\_26week\_across']

categorical\_columns = ['week', 'center\_id', 'meal\_id', 'emailer\_for\_promotion', 'homepage\_featured',

'city\_code', 'region\_code', 'center\_type', 'category', 'cuisine', 'year', 'month', 'quarter',

'week\_in\_month']

encoded\_columns = ['center\_id\_55', 'meal\_id\_1885', 'emailer\_for\_promotion\_0', 'homepage\_featured\_0', 'city\_code\_647',

'region\_code\_56', 'center\_type\_TYPE\_C', 'category\_Beverages', 'cuisine\_Italian', 'year\_3',

'month\_1', 'week\_in\_month\_2']

# Reading Data

trainset = pd.read\_csv("train\_feature.csv", index\_col='id').fillna(0)

trainset.head()

trainset.cuisine.value\_counts()

testset = pd.read\_csv("test\_feature.csv", index\_col='id').fillna(0)

testset.head()

testset.info()

# Data Preprocessing

def preprocess(trainset, testset, remove\_outliers=False):

if remove\_outliers:

trainset = trainset[trainset.num\_orders <= 20000]

trainset = trainset[trainset.checkout\_price >= 3]

dataset = trainset.append(testset, sort=False).fillna(0)

for column in categorical\_columns:

dataset[column] = dataset[column].astype('category')

dataset = dataset[features]

dataset = pd.get\_dummies(dataset[features])

dataset = dataset.drop(encoded\_columns, axis=1)

trainset = trainset[[target]].join(dataset)

testset = testset[[]].join(dataset)

return trainset, testset

trainset, testset = preprocess(trainset, testset)

print("Trainset size: **{}**".format(trainset.shape))

print("Testset size: **{}**".format(testset.shape))

feature\_columns = trainset.columns

feature\_columns = feature\_columns[feature\_columns != target]

feature\_columns.shape

trainset, validationset = train\_test\_split(trainset, random\_state=41, test\_size=0.2)

X\_train, y\_train = trainset[feature\_columns], np.log(trainset[target])

X\_val, y\_val = validationset[feature\_columns], np.log(validationset[target])

print("Train set size: **{}**".format(X\_train.shape))

print("Validation set size: **{}**".format(X\_val.shape))

# 6.2Training Models

**Decision tree regression**

DT = DecisionTreeRegressor()

DT.fit(X\_train, y\_train)

y\_pred = DT.predict(X\_val)

y\_pred[y\_pred<0] = 0

print('RMSLE:', 100\*np.sqrt(metrics.mean\_squared\_log\_error(y\_val, y\_pred)))

score\_DT = r2\_score(y\_pred,y\_val)

score\_DT

**KNeighborsRegression**

KNN = KNeighborsRegressor()

KNN.fit(X\_train, y\_train)

y\_pred = KNN.predict(X\_val)

y\_pred[y\_pred<0] = 0

print('RMSLE:', 100\*np.sqrt(metrics.mean\_squared\_log\_error(y\_val, y\_pred)))

score\_KNN = r2\_score(y\_pred,y\_val)

score\_KNN

**RandomForestRegression**

RFR=RandomForestRegressor()

RFR.fit(X\_train, y\_train)

y\_pred = RFR.predict(X\_val)

y\_pred[y\_pred<0] = 0

print('RMSLE:', 100\*np.sqrt(metrics.mean\_squared\_log\_error(y\_val, y\_pred)))

score\_RFR = r2\_score(y\_pred,y\_val)

score\_RFR

**AdaBoostRegressor**

ABR=AdaBoostRegressor()

ABR.fit(X\_train, y\_train)

y\_pred = ABR.predict(X\_val)

y\_pred[y\_pred<0] = 0

print('RMSLE:', 100\*np.sqrt(metrics.mean\_squared\_log\_error(y\_val, y\_pred)))

score\_ABR = r2\_score(y\_pred,y\_val)

score\_ABR

### LightGBM

lgb = LGBMRegressor(importance\_type='gain')

params = { 'num\_leaves': [41, 51], 'n\_estimators': [230, 260], 'min\_child\_samples': [40, 45, 50], 'random\_state': [2019] }

lgb\_grid = GridSearchCV(lgb, params, cv=5, scoring='neg\_mean\_squared\_error', n\_jobs=8) lgb\_grid.fit(X\_train, y\_train)

lgb = lgb\_grid.best\_estimator\_ score =100\*np.sqrt(metrics.mean\_squared\_log\_error(y\_val, lgb.predict(X\_val))

score\_lgb = r2\_score(y\_val,lgb.predict(X\_val))

score\_lgb

print('Best Estimator: **{}**'.format(lgb))

print('Best Score on validation: **{}**'.format(score))

ax = plot\_importance(lgb\_grid.best\_estimator\_, max\_num\_features=50, height=0.8, figsize=(12, 10))

ax.grid(**False**)

plt.title("LightGBM - Feature Importance", fontsize=15)

plt.show()

### XGBoost

xgb = XGBRegressor(objective='reg:squarederror', random\_state=41,missing=0.0, n\_jobs=8,

max\_depth=9, n\_estimators=300, min\_child\_weight=45)

xgb.fit(X\_train, y\_train)

100\*np.sqrt(metrics.mean\_squared\_log\_error(y\_val, xgb.predict(X\_val))

score\_xgb = r2\_score(y\_val,xgb.predict(X\_val))

score\_xgb

# Submission

X\_test = testset[feature\_columns]

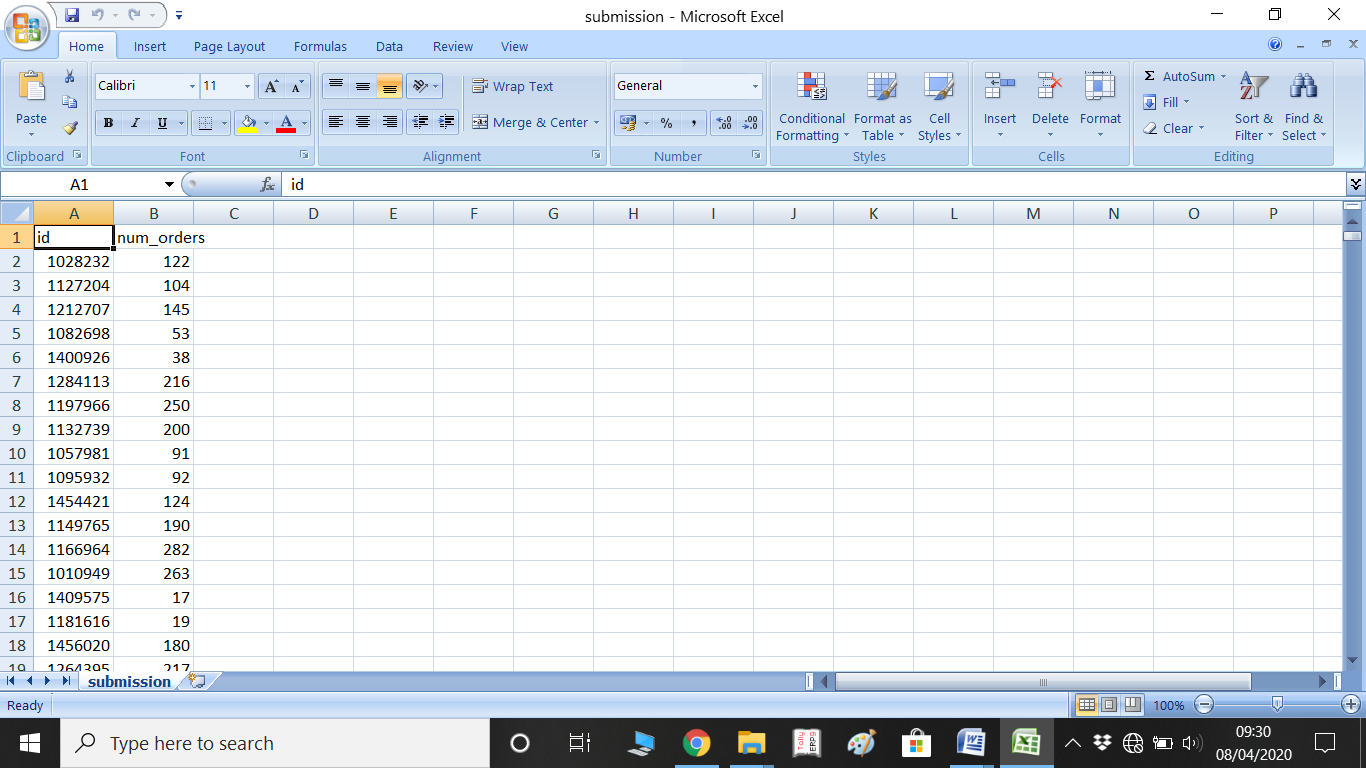
y\_pred = (np.exp(lgb.predict(X\_test)) + np.exp(xgb.predict(X\_test))) / 2

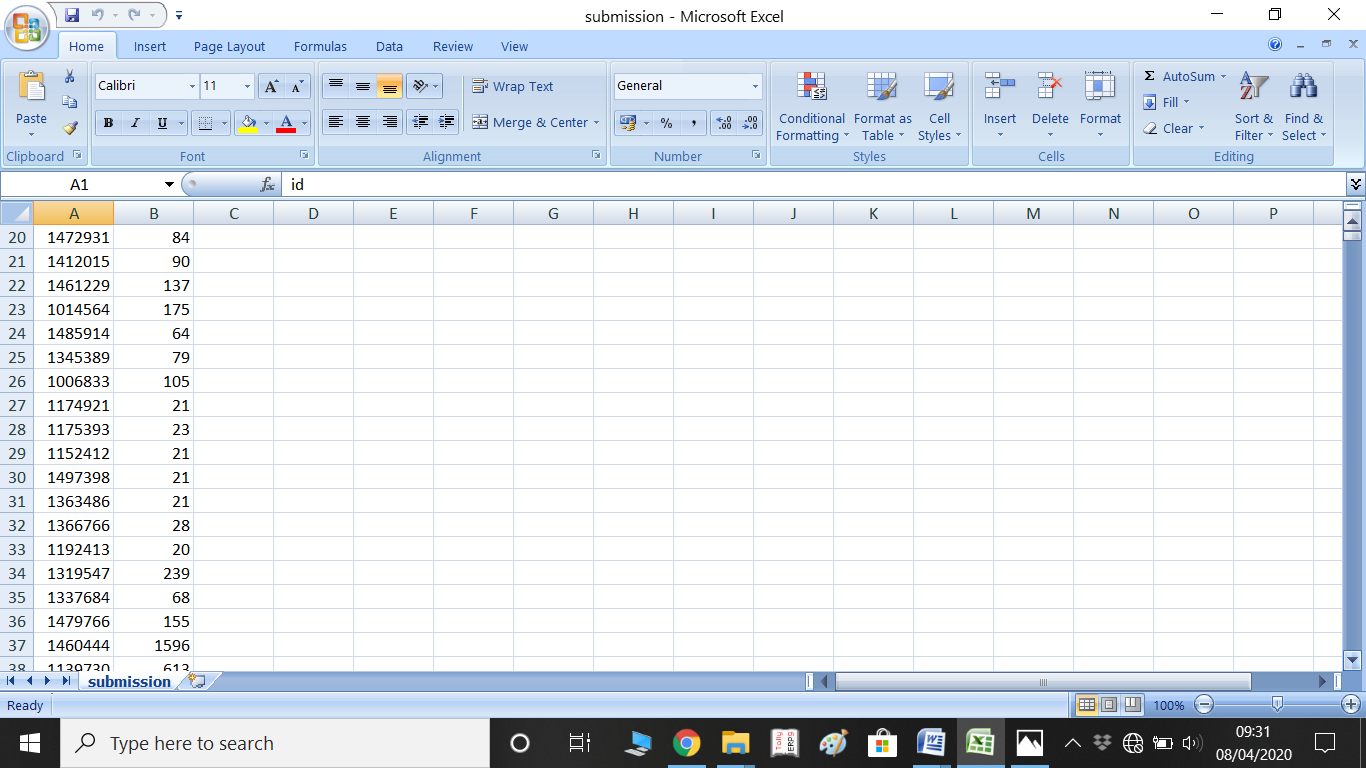
testset[target] = np.round(y\_pred, decimals=0)

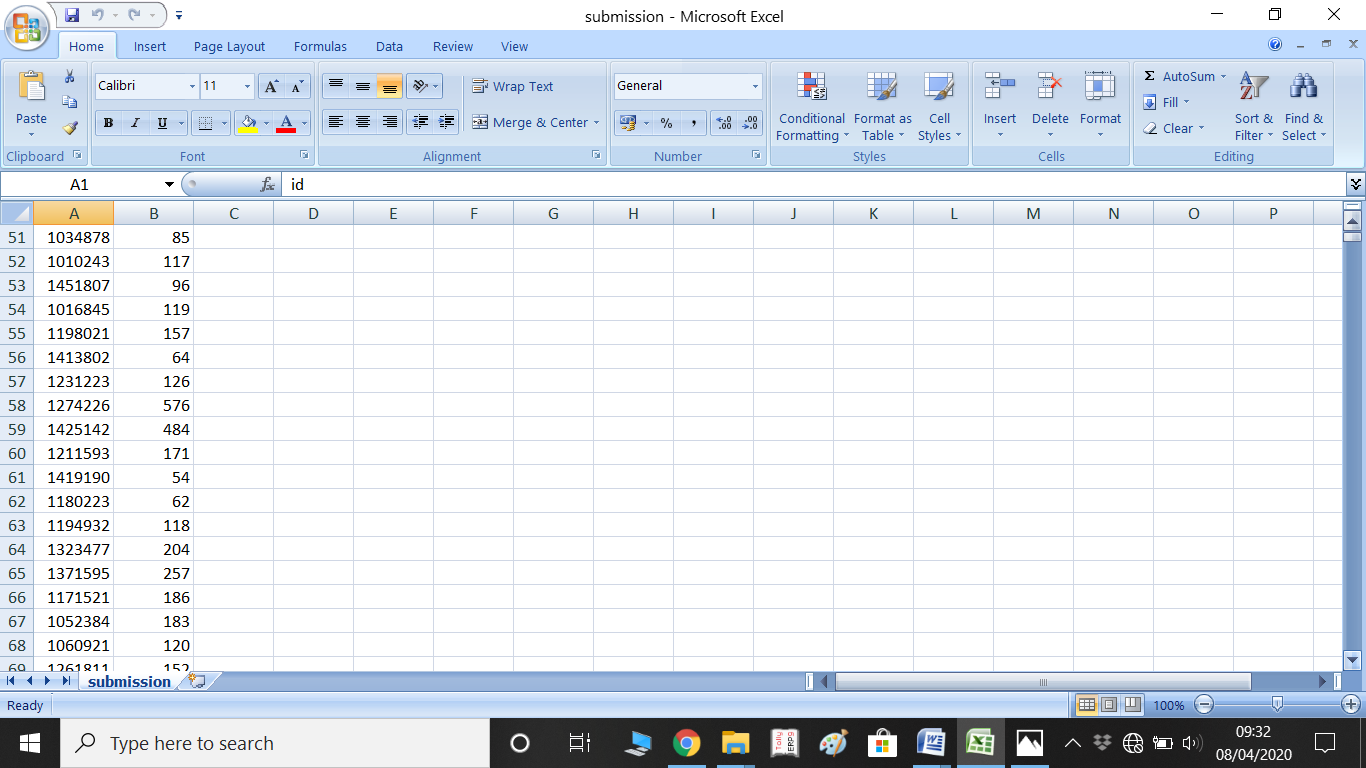
submission = testset[[target]]

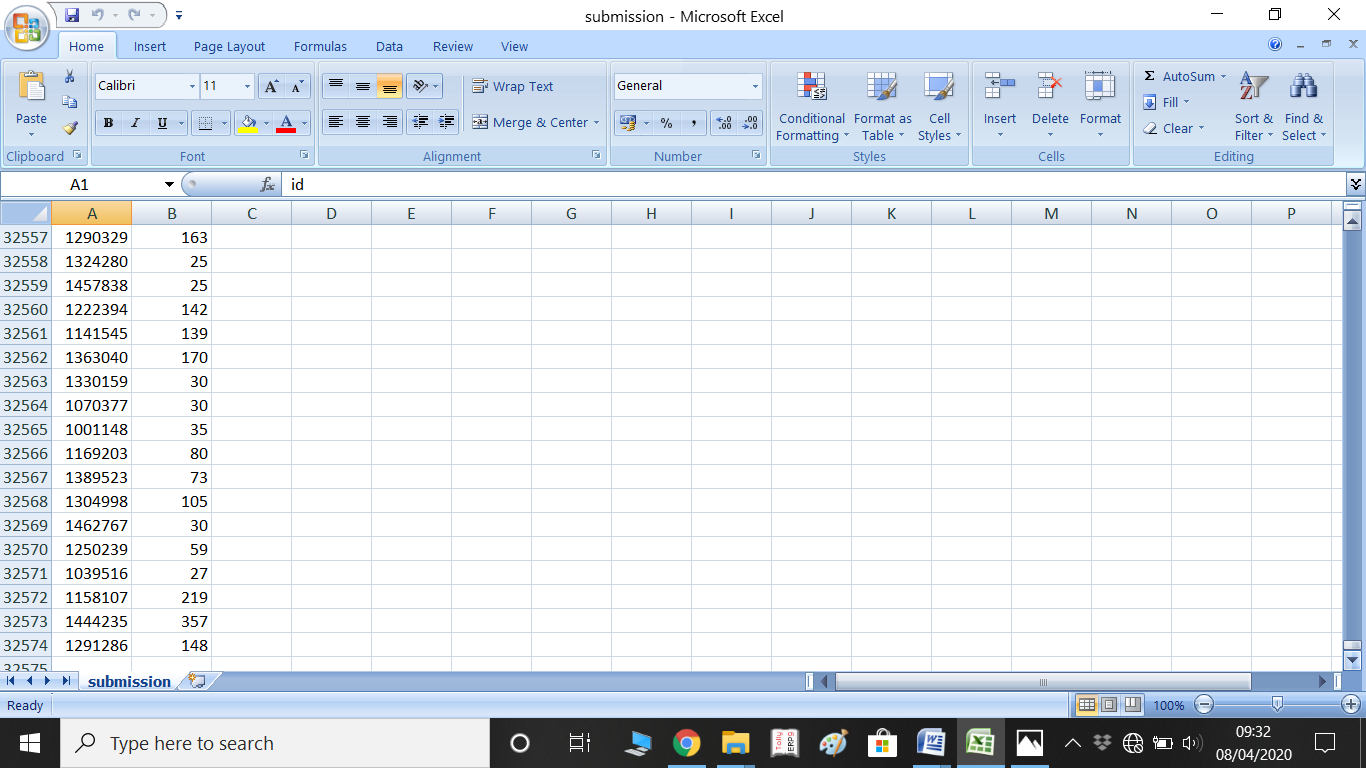
submission.to\_csv('ensemble.csv')

**7. RESULTS**









**8.CONCLUSION AND FUTURE ENHANCEMENTS**

# CONCLUSION:

# Our project deals with the prediction of food for meal delivery companies. we  help these companies with demand forecasting for upcoming weeks so that these centers will plan the stock of raw materials accordingly. After performing different experiments with features, data format,algorithms, parameters,etc. The ensemble of tuned lightGBM and XGBoost performs better than others.The evaluation metric of 100\*RMSLE final model is getting the result. It helps them by providing insights about the future demand based on present scenario so that they don’t end up in losses. So our solution will help in planning the stock of raw materials and staffing of the centers.

**FUTURE ENHANCEMENTS:**

We can build a model by which the meal companies can predict the food demand and can plan the stock of raw materials. We can enhance this project and can increase its scope. By creating a successful forecast demand of food, ensures that we have enough inventory for the upcoming period.

**9.Bibliography**

--> K. Karimi and H.J. Hamilton (2011), "Generation and Interpretation of Temporal Decision Rules", International Journal of Computer Information Systems and Industrial Management Applications, Volume 3

--> Kamiński, B.; Jakubczyk, M.; Szufel, P. (2017). "A framework for sensitivity analysis of decision trees". Central European Journal of Operations Research.

-->Joglekar, Sachin. "adaboost – Sachin Joglekar's blog". codesachin.wordpress.com.

-->Rojas, R. (2009). AdaBoost and the super bowl of classifiers a tutorial introduction to adaptive boosting. Freie University, Berlin, Tech. Rep.

-->Davies, Alex; Ghahramani, Zoubin (2014). "The Random Forest Kernel and other kernels for big data from random partitions".

-->Friedman, J. H. (February 1999). "Greedy Function Approximation: A Gradient Boosting Machine"

-->Ritter, A.; Muñoz-Carpena, R. (2013). "Performance evaluation of hydrological models: statistical significance for reducing subjectivity in goodness-of-fit assessments". Journal of Hydrology.

LINKS:

--->https://towardsdatascience.com/hyperparameter-tuning-c5619e7e6624

--->https://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean\_squared\_error.html

---> https://scikit-learn.org/stable/modules/preprocessing.html

--->https://datahack.analyticsvidhya.com/contest/genpact-machine-learning-hackathon-1/

--->https://towardsdatascience.com/why-and-how-to-use-merge-with-pandas-in-python-548600f7e738

--->https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d