A PROJECT REPORT

ON

FOOD DEMAND DYNAMICS WITH ADVANCED REGRESSION TECHNIQUES AND DEEP LEARNING MODELS

Submitted to

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY ANANTAPUR

For the partial fulfillment of the requirement for the Degree of

Master of Computer Applications

Submitted by

K. Umamahesh - 224M1F0036

Under the Guidance of

Mr. K. Niranjan, Assistant Professor MCA

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

VEMU INSTITUTE OF TECHNOLOGY

Approved by AICTE-New Delhi, Permanently Affiliated to JNTUA

Accredited by NAAC A+, Recognized by 2(F) & 12(B) UGC Act.,

An ISO 9001:2015 Certified Institution.

Tirupathi-Chittoor Highway. P. Kothakota.Pakala-517112

2022-2024

VEMU INSTITUTE OF TECHNOLOGY

Approved by AICTE-New Delhi, Permanently Affiliated to JNTUA
Accredited by NAAC A+, Recognized by 2(F) & 12(B) UGC Act.,
An ISO 9001:2015 Certified Institution,
Tirupathi-Chittoor Highway. P. Kothakota.Pakala-517112

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

CERTIFICATE

This is to certify that the project report entitled "FOOD DEMAND DYNAMICS WITH ADVANCED REGRESSION TECHNIQUES AND DEEP LEARNING MODELS" is being submitted by K. Umamahesh (224M1F0036), in partial fulfillment of the requirements for the award of Master of Computer Applications to the JNTUA, Anantapuramu. This Project report is a bonafide work carried out by him under my guidance and supervision. The results embodied in this report request have not been submitted to any University or Institute for the award of any Degree or Diploma.

Internal Guide
Mr. K. Niranjan, M.Tech.
Assistant Professor

Head of The Department
Dr. K. Venkataramana, Ph.D.
Professor and HOD

External Viva-Voce Exam Held on _____

Internal Examiner **External Examiner**

VEMU INSTITUTE OF TECHNOLOGY

Approved by AICTE-New Delhi, Permanently Affiliated to JNTUA
Accredited by NAAC A+, Recognized by 2(F) & 12(B) UGC Act.,
An ISO 9001:2015 Certified Institution,
Tirupathi-Chittoor Highway. P. Kothakota.Pakala-517112

DEPARTMENT OF MASTER OF COMPUTER APPLICATIONS

DECLARATION

We hereby declare that the project report entitled "FOOD DEMAND DYNAMICS WITH ADVANCED REGRESSION TECHNIQUES AND DEEP LEARNING MODELS" submitted to the Department of Master of Computer Applications, in partial fulfillment of requirements for the award of the degree of Master of Computer Applications. This Project is the result of our own effort and it has not been submitted to the other University or Institute for the award of any Degree or Diploma.

K. Umamahesh (224M1F0036)

ACKNOWLEDGEMENT

An Endeavour over a long period can be successful only with an advice and support of many well-wishers. I take this opportunity to express my gratitude and appreciation to all of those who encouraged me for successfully completion of the Project work.

I wish to express my heart full thanks and deep sense of gratitude to the honorable founder **Dr. K. CHANDRA SEKHAR NAIDU** Garu, for his encouragement and inspiration throughout the process.

My Special thanks to our Principal **Dr. NAVEEN KILARI Garu**, who provide all the required facilities and helped in accomplishing the project report within time.

I am thankful to our Head of the Department **Dr. K. Venkataramana**, Professor and HOD, for his valuable guidance and efforts throughout the project work.

With immense pleasure we regard my deep sense of indebtedness and gratitude to the project Coordinator **Dr. R. Yamuna**, Associate Professor who was a source of inspiration.

I am thankful to my guide **Mr. K. Niranjan**, Assistant Professor for his valuable guidance and efforts throughout the project work.

Finally, I would like to extend our deep sense of gratitude to all faculty members, friends and last but not greatly indebted to my parents who inspired me at all circumstances.

K. Umamahesh (224M1F0036)

CONTENTS

ABSTRACT

This project focuses on the critical task of demand forecasting in the food industry, where product shelf life is limited, and mismanaged inventory can lead to significant losses. Leveraging machine learning and deep learning techniques, the study analyzes the 'Food Demand Forecasting' dataset from Genpact, examining the impact of various factors on demand and identifying influential features. The project conducts a comparative analysis of seven regression models, including Random Forest, Gradient Boosting, LightGBM, XGBoost, Cat Boost, LSTM, and Bidirectional LSTM. Results highlight the effectiveness of deep learning models, with LSTM outperforming others, achieving low error rates (RMSLE: 0.28, RMSE: 18.83, MAPE: 6.56%, MAE: 14.18). This research underscores the potential of deep learning for accurate demand forecasting in the food industry.

LIST OF FIGURES

LIST OF SCREENS

LIST OF ABBREVIATIONS

LSTM	-	**Long Short-Term Memory**
BI-LSTM	-	**Bidirectional Long Short-Term Memory**
RMSLE	-	**Root Mean Square Log Error**
RMSE	-	**Root Mean Square Error**
MAPE	-	**Mean Absolute Percentage Error**
MAE	-	**Mean Absolute Error**

CHAPTER - 1

INTRODUCTION

In today's dynamic market landscape, where consumer needs vary widely and competition among companies intensifies, demand forecasting has emerged as a critical tool for effective demand-supply chain management. The accuracy of demand forecasts directly impacts a company's profitability, making it a crucial aspect of strategic planning and administration within organizations.

A misjudged demand forecast can have detrimental consequences. Overestimating demand may result in excessive inventory, leading to increased risk of wastage and high costs. Conversely, underestimating demand can lead to stockouts, where customers are unable to find the products they seek, potentially driving them to competitors. Therefore, employing robust demand forecasting methods is essential for optimizing inventory levels and ensuring customer satisfaction.

The significance of demand forecasting transcends departmental boundaries within a company. Various divisions rely on demand forecasts to inform their decision-making processes. The financial department utilizes forecasts to estimate costs, project profit levels, and determine the required capital for operations. Marketing departments leverage demand forecasts to devise strategic plans and assess the impact of marketing initiatives on sales volumes. Purchasing departments use forecasts to plan short- and long-term investments in inventory and resources. Similarly, operations departments rely on accurate forecasts to procure raw materials, allocate machinery, and plan labor resources efficiently.

Given the widespread reliance on demand forecasts across organizational functions, the accuracy of these forecasts is paramount. A high level of accuracy can translate into significant benefits, including improved demand-supply chain management, reduced wastage, and enhanced profitability

Demand forecasting plays a pivotal role in the strategic planning and operations of modern businesses. By accurately predicting future demand, companies can optimize their inventory levels, minimize costs, and meet customer expectations effectively. As market dynamics continue to evolve, the importance of demand forecasting as a strategic tool for business success will only continue to grow.

HOW MACHINE LEARNING WORKS

Machine learning is a branch of artificial intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy.

Machine learning is an important component of the growing field of data science. Through the use of statistical methods, algorithms are trained to make classifications or predictions, uncovering key insights within data mining projects. These insights subsequently drive decision making within applications and businesses, ideally impacting key growth metrics. As big data continues to expand and grow, the market demand for data scientists will increase, requiring them to assist in the identification of the most relevant business questions and subsequently the data to answer them.

Machine Learning Algorithms

A Decision Process: In general, machine learning algorithms are used to make a prediction or classification. Based on some input data, which can be labelled or unlabeled, your algorithm will produce an estimate about a pattern in the data.

An Error Function: An error function serves to evaluate the prediction of the model. If there are known examples, an error function can make a comparison to assess the accuracy of the model. A Model Optimization Process: If the model can fit better to the data points in the training set, then weights are adjusted to reduce the discrepancy between the known example and the model estimate. The algorithm will repeat this evaluate and optimize process, updating weights autonomously until a threshold of accuracy has been met.

Types of Machine Learning Methods

Supervised machine learning

Supervised learning also known as supervised machine learning, is defined by its use of labelled datasets to train algorithms that to classify data or predict outcomes accurately. As input data is fed into the model, it adjusts its weights until the model has been fitted appropriately. This occurs as part of the cross validation process to ensure that the model avoids over fitting or under fitting. Supervised learning helps organizations solve for a variety of real-world problems at scale, such as classifying spam in a separate folder from your inbox. Some methods used in supervised

learning include neural networks, naïve bayes, linear regression, logistic regression, random forest, support vector machine (SVM), and more.

Unsupervised machine learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross selling strategies, customer segmentation, image and pattern recognition. It's also used to reduce the number of features in a model through the process of dimensionality reduction; principal component analysis (PCA) and singular value decomposition (SVD) are two common approaches for this. Other algorithms used in unsupervised learning include neural networks, kmeans clustering, probabilistic clustering methods, and more

Semi-supervised learning

Semi-supervised learning offers a happy medium between supervised and unsupervised learning. During training, it uses a smaller labelled data set to guide classification and feature extraction from a larger, unlabeled data set. Semi-supervised learning can solve the problem of having not enough labelled data (or not being able to afford to label enough data) to train a supervised learning algorithm.

Practical Use of Machine Learning

Speech Recognition: It is also known as automatic speech recognition (ASR), computer speech recognition, or speech-to-text, and it is a capability which uses natural language processing (NLP) to process human speech into a written format. Many mobile devices incorporate speech recognition into their systems to conduct voice search—e.g. Siri—or provide more accessibility around texting.

Customer Service: Online chatbots are replacing human agents along the customer journey. They answer frequently asked questions (FAQs) around topics, like shipping, or provide personalized advice, cross-selling products or suggesting sizes for users, changing the way we think about customer engagement across websites and social media platforms. Examples include messaging bots on e-commerce sites with virtual agents, messaging apps, such as Slack and Facebook Messenger, and tasks usually done by virtual assistants and voice assistants.

Computer Vision: This AI technology enables computers and systems to derive meaningful information from digital images, videos and other visual inputs, and based on those inputs, it can take action. This ability to provide recommendations distinguishes it from image recognition tasks. Powered by convolutional neural networks, computer vision has applications within photo tagging in social media, radiology imaging in healthcare, and self driving cars within the automotive industry.

Recommendation Engines: Using past consumption behavior data, AI algorithms can help to discover data trends that can be used to develop more effective cross-selling strategies. This is used to make relevant add-on recommendations to customers during the checkout process for online retailers.

Automated stock trading: Designed to optimize stock portfolios, AI-driven high-frequency trading platforms make thousands or even millions of trades per day without human intervention.

WHAT IS DEEP LEARNING

Deep learning is one of the foundations of artificial intelligence (AI), and the current interest in deep learning is due in part to the buzz surrounding AI. Deep learning techniques have improved the ability to classify, recognize, detect and describe – in one word, understand. For example, deep learning is used to classify images, recognize speech, detect objects and describe content.

Several developments are now advancing deep learning:

Algorithmic improvements have boosted the performance of deep learning methods.

New machine learning approaches have improved accuracy of models.

New classes of neural networks have been developed that fit well for applications like text translation and image classification.

We have a lot more data available to build neural networks with many deep layers, including streaming data from the Internet of Things, textual data from social media, physicians notes and investigative transcripts.

Computational advances of distributed cloud computing and graphics processing units have put incredible computing power at our disposal. This level of computing power is necessary to train deep algorithms.

At the same time, human-to-machine interfaces have evolved greatly as well. The mouse and the keyboard are being replaced with gesture, swipe, touch and natural language, ushering in a renewed interest in AI and deep learning.

How Deep Learning Works

Deep learning changes how you think about representing the problems that you're solving with analytics. It moves from telling the computer how to solve a problem to training the computer to solve the problem itself.

A traditional approach to analytics is to use the data at hand to engineer features to derive new variables, then select an analytic model and finally estimate the parameters (or the unknowns) of that model. These techniques can yield predictive systems that do not generalize well because completeness and correctness depend on the quality of the model and its features. For example, if you develop a fraud model with feature engineering, you start with a set of variables, and you most likely derive a model from those variables using data transformations. You may end up with 30,000 variables that your model depends on, then you have to shape the model, figure out which variables are meaningful, which ones are not, and so on. Adding more data requires you to do it all over again.

The new approach with deep learning is to replace the formulation and specification of the model with hierarchical characterizations (or layers) that learn to recognize latent features of the data from the regularities in the layers. The paradigm shift with deep learning is a move from feature engineering to feature representation. The promise of deep learning is that it can lead to predictive systems that generalize well, adapt well, continuously improve as new data arrives, and are more dynamic than predictive systems built on hard business rules. You no longer fit a model. Instead, you train the task.

Deep learning is making a big impact across industries. In life sciences, deep learning can be used for advanced image analysis, research, drug discovery, prediction of health problems and disease symptoms, and the acceleration of insights from genomic sequencing. In transportation, it can help autonomous vehicles adapt to changing conditions. It is also used to protect critical infrastructure and speed response.

How Deep Learning Being Used

To the outside eye, deep learning may appear to be in a research phase as computer science researchers and data scientists continue to test its capabilities. However, deep learning has many

practical applications that businesses are using today, and many more that will be used as research continues. Popular uses today include:

Speech Recognition

Both the business and academic worlds have embraced deep learning for speech recognition. Xbox, Skype, Google Now and Apple's Siri, to name a few, are already employing deep learning technologies in their systems to recognize human speech and voice patterns.

Natural Language Processing

Neural networks, a central component of deep learning, have been used to process and analyse written text for many years. A specialization of text mining, this technique can be used to discover patterns in customer complaints, physician notes or news reports, to name a few.

Image Recognition

One practical application of image recognition is automatic image captioning and scene description. This could be crucial in law enforcement investigations for identifying criminal activity in thousands of photos submitted by bystanders in a crowded area where a crime has occurred. Self-driving cars will also benefit from image recognition through the use of 360- degree camera technology.

Recommendation Systems

Amazon and Netflix have popularized the notion of a recommendation system with a good chance of knowing what you might be interested in next, based on past behavior. Deep learning can be used to enhance recommendations in complex environments such as music interests or clothing preferences across multiple platforms.

Recent advances in deep learning have improved to the point where deep learning outperforms humans in some tasks like classifying objects in images [20]. While deep learning was first theorized in the 1980s, there are two main reasons it has only recently become useful:

1. Deep learning requires large amounts of labelled data. For example, driverless car development requires millions of images and thousands of hours of video.

2. Deep learning requires substantial computing power. High-performance GPUs have a parallel architecture that is efficient for deep learning. When combined with clusters or cloud computing, this enables development teams to reduce training time for a deep learning network from weeks to hours or less. When choosing between machine learning and deep learning, consider

whether you have a high-performance GPU and lots of labelled data. If you don't have either of those things, it may make more sense to use machine learning instead of deep learning. Deep learning is generally more complex, so you'll need at least a few thousand images to get reliable results. Having a high-performance GPU means the model will take less time to analyze all those images.

Deep Learning Opportunities and Applications

A lot of computational power is needed to solve deep learning problems because of the iterative nature of deep learning algorithms, their complexity as the number of layers increase, and the large volumes of data needed to train the networks.

The dynamic nature of deep learning methods – their ability to continuously improve and adapt to changes in the underlying information pattern – presents a great opportunity to introduce more dynamic behavior into analytics. Greater personalization of customer analytics is one possibility. Another great opportunity is to improve accuracy and performance in applications where neural networks have been used for a long time. Through better algorithms and more computing power, we can add greater depth.

While the current market focus of deep learning techniques is in applications of cognitive computing, there is also great potential in more traditional analytics applications, for example, time series analysis. Another opportunity is to simply be more efficient and streamlined in existing analytical operations. Recently, some study showed that with deep neural networks in speech-to-text transcription problems. Compared to the standard techniques, the word error-rate decreased by more than 10 percent when deep neural networks were applied. They also eliminated about 10 steps of data preprocessing, feature engineering and modelling. The impressive performance gains and the time savings when compared to feature engineering signify a paradigm shift.

Here are some examples of deep learning applications are used in different industries:

Automated Driving:

Automotive researchers are using deep learning to automatically detect objects such as stop signs and traffic lights. In addition, deep learning is used to detect pedestrians, which helps decrease accidents.

Aerospace and Defense:

Deep learning is used to identify objects from satellites that locate areas of interest, and identify safe or unsafe zones for troops.

Medical Research:

Cancer researchers are using deep learning to automatically detect cancer cells. Teams at UCLA built an advanced microscope that yields a high-dimensional data set used to train a deep learning application to accurately identify cancer cells [24].

Industrial Automation:

Deep learning is helping to improve worker safety around heavy machinery by automatically detecting when people or objects are within an unsafe distance of machines.

Electronics:

Deep learning is being used in automated hearing and speech translation. For example, home assistance devices that respond to your voice and know your preferences are powered by deep learning applications.

CHAPTER - 2

LITERATURE SURVEY

2.1 Using Internet of Things (IoT) in agri-food supply chains: A research framework for social good with network clustering analysis

S. Yadav, et.al, Agri-food supply chains (AFSCs) are critical in our society. Proper management of AFSCs is crucial for improving social welfare. Over the past years, digitization in AFSCs has emerged as a new paradigm. In this context, the Internet of Things (IoT) is a growing approach, providing a huge amount of information to manage AFSCs. Thus, the purpose of this article is to examine extensive studies on IoT-based AFSC. Our research starts with the identification of 346 articles in the relevant field from the Web of Science (WOS) database by applying rigorous filtration. Using the VOS viewer software, a network analysis has been performed. With seven identified clusters, this article recognizes the role of IoT technologies as Cluster 1: agri-food safety, traceability and sustainability; Cluster 2: AFSC sustainability; Cluster 3: AFSC performance measurement; Cluster 4: AFSC resilience in disruption; Cluster 5: AFSC integration and traceability; Cluster 6: AFSC transparency and coordination, and finally Cluster 7 identifies the barriers in IoT adoption. Thus, findings of this study offer robust guidance to link IoT technologies and AFSCs together. Based on these findings, propositions are proposed and a research framework is established. We believe the findings would help engineering managers, researchers, and government regulating bodies better plan and manage AFSCs for social good.

2.2 Forecasting one-day ahead electricity prices for Italian electricity market using parametric and nonparametric approaches,

I. Shah, et,al, Over the last three decades, accurate modeling and forecasting of electricity prices has become a key issue in competitive electricity markets. As electricity price series usually exhibit several complex features, such as high volatility, seasonality, calendar effect, non-stationarity, non-linearity and mean reversion, price forecasting is not a trivial task. However, participants of electricity market need price forecast to make decisions in their daily activity in the market, such as trading, risk management or future planning. In this study we consider linear and nonlinear models for one-day-ahead forecast of electricity prices using components estimation techniques. This approach requires to filter out the structural, deterministic components from the original time

series and to model the residual component by means of some stochastic process. The final forecast is obtained by combining the predictions of both these components. In this work, linear and non-linear models are applied to both, deterministic and stochastic, components. In the case of stochastic component, Autoregressive, Nonparametric Autoregressive, Functional Autoregressive, and Nonparametric Functional Autoregressive have been considered. Furthermore, two naïve benchmarks are applied directly to the price time series and their results are compared with our proposed models. An application of the proposed methodology is presented for the Italian electricity market (IPEX). Our analysis suggests that, in terms of Mean Absolute Error, Mean Absolute Percentage Error, and Pearson correlation coefficient, best results are obtained when deterministic component is estimated by using parametric approach. Further, Functional Autoregressive model performs relatively better than the rest while Nonparametric Autoregressive is highly competitive.

2.3 Electricity spot prices forecasting based on ensemble learning

N. Bibi, et,al, Efficient modeling and forecasting of electricity prices are essential in today's competitive electricity markets. However, price forecasting is not easy due to the specific features of the electricity price series. This study examines the performance of an ensemble-based technique for forecasting short-term electricity spot prices in the Italian electricity market (IPEX). To this end, the price time series is divided into deterministic and stochastic components. The deterministic component that includes long-term trends, annual and weekly seasonality, and bank holidays, is estimated using semi-parametric techniques. On the other hand, the stochastic component considers the short-term dynamics of the price series and is estimated by time series and various machine learning algorithms. Based on three standard accuracy measures, the results indicate that the ensemble-based model outperforms the others, while the random forest and ARMA are highly competitive.

2.4 The study of a forecasting sales model for fresh food

C.-Y. Chen, et,al, Convenience stores (CVS) are an integral part of the retail industry. Merchandises are circulated from suppliers and CVSs to consumers. Therefore, if the goods which consumers desire are always out of stock, no matter how good the service, customer satisfaction will be hard to be improved. The point of sale (POS) system provides the information analysis ability and can be used to analyze consumers' purchasing behavior as well as forecast needs.

Therefore, managers of CVSs improve their revenues based on these data. No matter whether in the urban area, suburb or mountainous area, own the different characteristics of business circles. Therefore, business circles are important influencing factors. As Franchise Chains grow new operators increasingly join the business. However, it is difficult to train ordering operators in the short time. Therefore, some CVSs run out of stock and some may always order too large quantities. Only control each order precisely can meet customers' need. Therefore, how to control the order and stock of CVSs has become one of the important issues in the management of CVSs. The purpose of this research is to discuss and develop a mechanism for controlling the order and managing the stock for CVSs. The theory of the buyer and seller exchanging system proposed by Kotler was modified to include the theory of consolidating loop structure. The "Ordinary day and holiday moving average method" and "back-propagation neural network" were proposed and tested based on the operating characteristics of business circle and sale forecasting. The method can be used to improve the ordering and discarding rate for achieving the goal of ordering the right items in the right amount.

2.5 Time series forecasting: Problem of heavy-tailed distributed noise

C.-Y. Chen, et,al, Time series forecasting has been the area of intensive research for years. Statistical, machine learning or mixed approaches have been proposed to handle this one of the most challenging tasks. However, little research has been devoted to tackle the frequently appearing assumption of normality of given data. In our research, we aim to extend the time series forecasting models for heavy-tailed distribution of noise. In this paper, we focused on normal and Student's t distributed time series. The SARIMAX model (with maximum likelihood approach) is compared with the regression tree-based method—random forest. The research covers not only forecasts but also prediction intervals, which often have hugely informative value as far as practical applications are concerned. Although our study is focused on the selected models, the presented problem is universal and the proposed approach can be discussed in the context of other systems.

CHAPTER - 3

SYSTEM ANALYSIS

3.1EXISTING SYSTEM

Accurate forecasting is become necessity in food industry to meet demand supply requirements. Many food products has shelf life and if demand forecast is not accurate and then either short life products will be wasted or in some scenarios it goes for shortage. Many deep learning or machine learning algorithms was introduced for accurate forecasting but they lack support of Time series or LAG data

3.1.1Disadvantages of Existing system

1. Time Series Data Exclusion.
2. Limited Lag Data Analysis.
3. Insufficient Forecasting Techniques.
4. Dependence on Short-Term Data.
5. Insufficient Shelf-Life Prediction.

3.2PROPOSED SYSTEM

To overcome from this problem author of this paper extracting LAG data from dataset and then assigning weight to target variable by using alpha values as 0.5 and products which are recent or high in demand will have high weight values.

Time series data is extracted from entire dataset for 10 week periods and then forecasting will happen for next 10 weeks. Extracted lag data will get trained with various ML and DL algorithms such as Random Forest, Gradient Boosting, XGBOOST, CATBOOST, LIGHT GBM, LSTM and BI-LSTM. Each algorithm performance is evaluated in terms of RMSE (root mean square error), MAE (mean absolute error), MAPE (mean absolute percentage error) and RMSLE. All this metrics refers to difference between original values and predicted values so the lower the difference the better is the algorithm. Among all algorithms LSTM is giving less MAE and RMSE error rate.

3.2.1Advantages of Proposed System

1. Time-Dependent Data Handling Advancements.
2. Comparison of Various Regression Techniques.
3. Comprehensive Dataset Analysis Methods.
4. Deep Learning Model Superiority.
5. Performance Metrics Validation Results.

3.3FUNCTIONAL REQUIREMENTS

In software engineering and systems engineering, a **functional requirement** defines a function of a system or its component, where a function is described as a specification of behavior between outputs and inputs.

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>." The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/stakeholder request → analyze → use case → incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

3.4NON-FUNCTIONAL REQUIREMENTS

(NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *"how fast does the website load?"* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are> 10000. Description of non-functional requirements is just as critical as a functional requirement.

Advantages of Non-Functional Requirement

Benefits/pros of Non-functional testing are:

- The nonfunctional requirements ensure the software system follow legal and compliance rules.
- They ensure the reliability, availability, and performance of the software system
- They ensure good user experience and ease of operating the software.
- They help in formulating security policy of the software system.

Disadvantages of Non-functional requirement

Cons/drawbacks of Non-function requirement are:

- None functional requirement may affect the various high-level software subsystem
- They require special consideration during the software architecture/high-level design phase which increases costs.
- Their implementation does not usually map to the specific software sub-system,
- It is tough to modify non-functional once you pass the architecture phase.

KEY LEARNING

- A non-functional requirement defines the performance attribute of a software system.
- Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc.
- Example of Non Functional Requirement is Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
- Functional Requirement is a verb while Non-Functional Requirement is an attribute
- The advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software
- The biggest disadvantage of Non-functional requirement is that it may affect the various high-level software subsystems.

3.5 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.6SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.6REQUIREMENT SPECIFICATION

3.6.1 Hardware Requirements

- PROCESSOR - Intel Core i3/i5
- RAM - 4 GB
- HARD DISK - 500 GB

3.6.2 Software Requirements

- Operating system : Windows 8, Windows 10
- Coding Language : PYTHON
- Documentation : MS Office

CHAPTER - 4

SYSTEM DESIGN

4.1MODULE DESCRIPTION

1. Data Collection

Import necessary Python libraries and packages including pandas, NumPy, Matplotlib, seaborn, scikit-learn, TensorFlow, Keras, LightGBM, XGBoost, CatBoost.

Load the meal sales dataset and fulfillment center dataset.

Merge the datasets to combine relevant information for analysis.

2. Exploratory Data Analysis (EDA)

Visualize the distribution of features in the dataset using histograms and box plots.

Analyze the relationships between different features using correlation matrices and heatmaps.

Investigate trends and patterns in meal sales across various regions, center types, and weeks.

3. Feature Engineering and Preprocessing

Perform feature engineering to extract relevant features for modeling.

Handle missing values and encode categorical variables using techniques like Label Encoding.

Normalize the features using Min-Max scaling to ensure uniformity in feature scales.

Prepare lagged data to incorporate time-series features for predicting meal sales.

4. Model Selection and Training

Traditional Machine Learning Models

Train machine learning models including Random Forest Regressor, Gradient Boosting Regressor using Grid Search CV for hyperparameter tuning.

Evaluate model performance using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Logarithmic Error (RMSLE).

Deep Learning Models

Train deep learning models including Long Short-Term Memory (LSTM) networks, Bidirectional LSTM (Bi-LSTM), Convolutional Neural Networks (CNN) using Keras.

Compile and train the models using appropriate loss functions and optimizers.

Implement techniques like dropout to prevent overfitting.

Save the best models using Model Check point to avoid retraining.

5. Model Evaluation and Performance Metrics

Calculate performance metrics such as MAE, RMSE, MAPE, and RMSLE for each model.

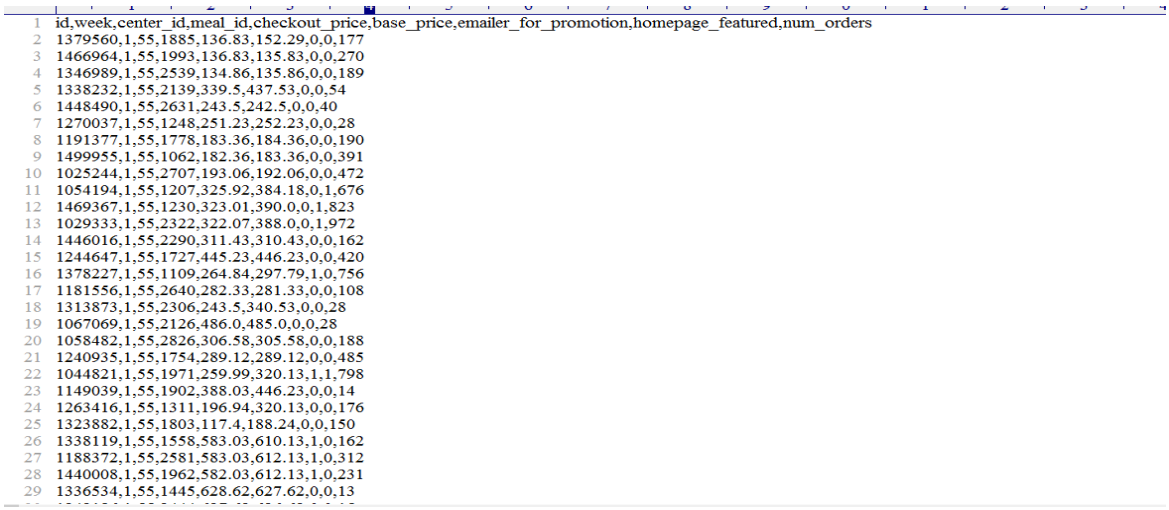
Visualize the predicted sales against actual sales to understand model predictions.

6. Results Interpretation and Conclusion

Analyze the performance of each model and compare their effectiveness in predicting meal sales.

Discuss insights gained from the analysis and potential areas for improvement.

TIME SERIES DATASET:



```
id,week,center_id,meal_id,checkout_price,base_price,emailer_for_promotion,homepage_featured,num_orders
1379560,1,55,1885,136.83,152.29,0,0,177
1466964,1,55,1993,136.83,135.83,0,0,270
1346989,1,55,2539,134.86,135.86,0,0,189
1338232,1,55,2139,339.5,437.53,0,0,54
1448490,1,55,2631,243.5,242.5,0,0,40
1270037,1,55,1248,251.23,252.23,0,0,28
1191377,1,55,1778,183.36,184.36,0,0,190
1499955,1,55,1062,182.36,183.36,0,0,391
1025244,1,55,2707,193.06,192.06,0,0,472
1054194,1,55,1207,325.92,384.18,0,1,676
1469367,1,55,1230,323.01,390.0,0,1,823
1029333,1,55,2322,322.07,388.0,0,1,972
1446016,1,55,2290,311.43,310.43,0,0,162
1244647,1,55,1727,445.23,446.23,0,0,420
1378227,1,55,1109,264.84,297.79,1,0,756
1181556,1,55,2640,282.33,281.33,0,0,108
1313873,1,55,2306,243.5,340.53,0,0,28
1067069,1,55,2126,486.0,485.0,0,0,28
1058482,1,55,2826,306.58,305.58,0,0,188
1240935,1,55,1754,289.12,289.12,0,0,485
1044821,1,55,1971,259.99,320.13,1,1,798
1149039,1,55,1902,388.03,446.23,0,0,14
1263416,1,55,1311,196.94,320.13,0,0,176
1323882,1,55,1803,117.4,188.24,0,0,150
1338119,1,55,1558,583.03,610.13,1,0,162
1188372,1,55,2581,583.03,612.13,1,0,312
1440008,1,55,1962,582.03,612.13,1,0,231
1336534,1,55,1445,628.62,627.62,0,0,13
```

In above dataset screen first row represents dataset column names and remaining rows represents dataset values and in last column we have orders as the sales which can be used to calculate target variable by applying EWMA (Exponentially Weighted Moving Average) formula. So by using above dataset we will train and test each algorithm performance

4.2SYSTEM ARCHITECTURE:

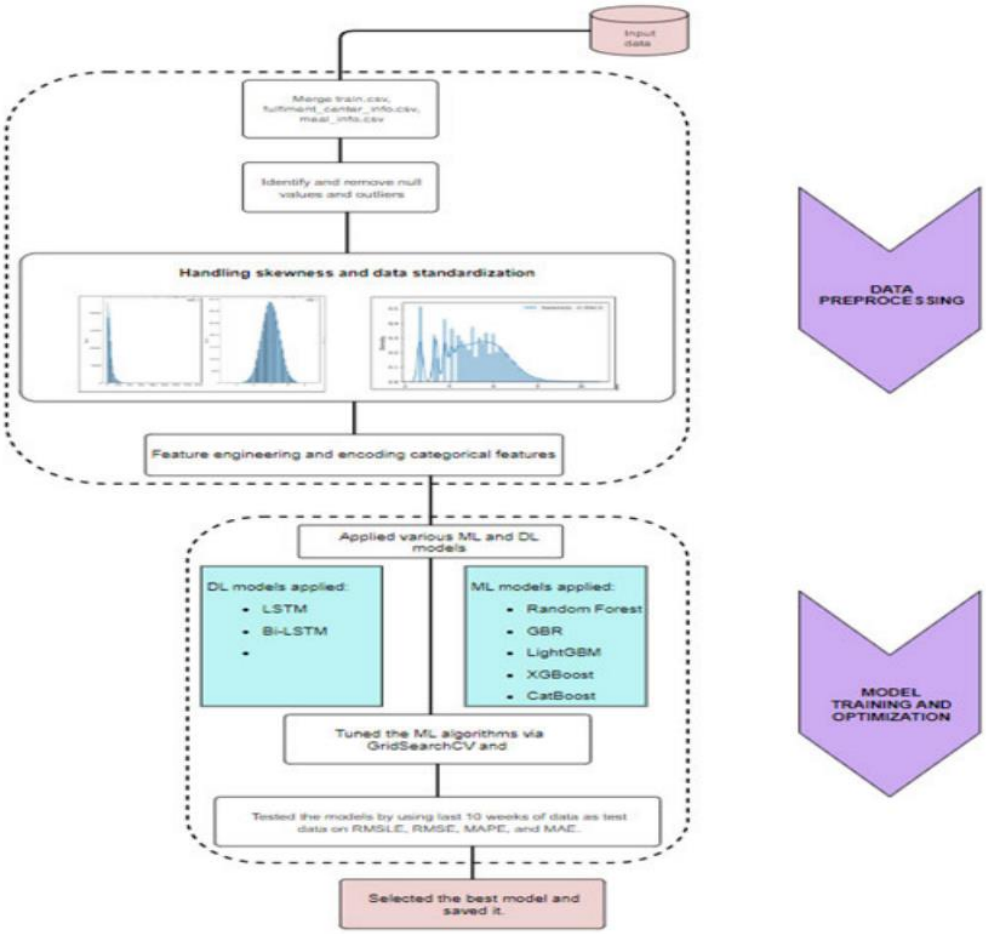


FIG1: ARCHITECTURE

4.3 DATA FLOW DIAGRAM

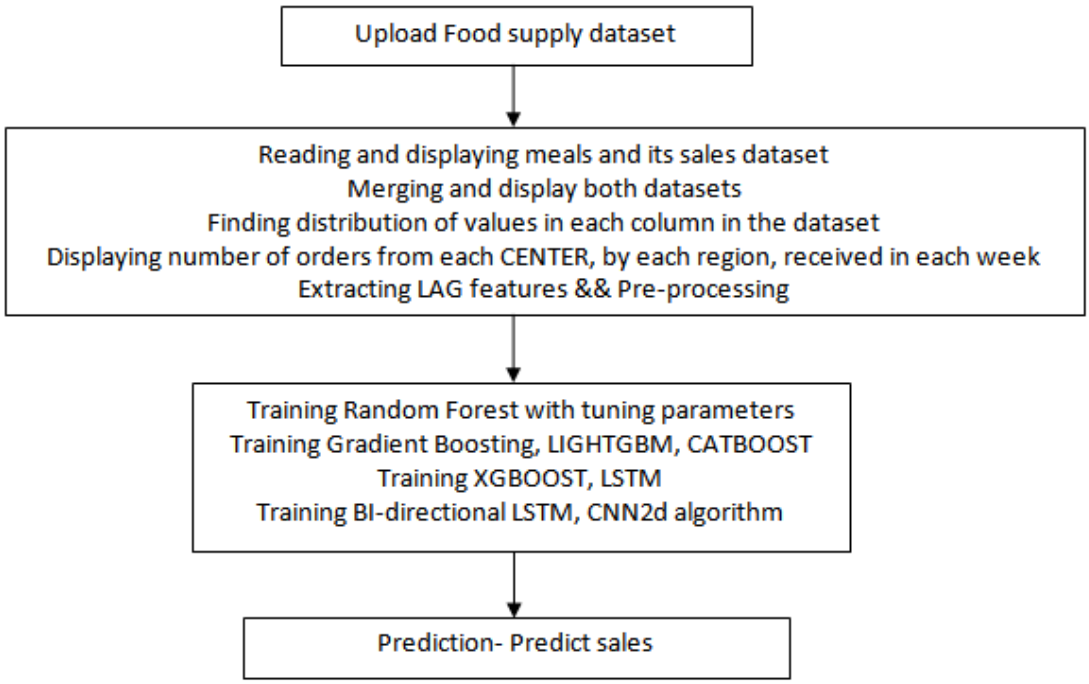


FIG2: DATA FLOW DIAGRAM

4.4 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

GOALS:

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

4.4.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted

FIG3: USE CASE DIAGRAM

4.4.2 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

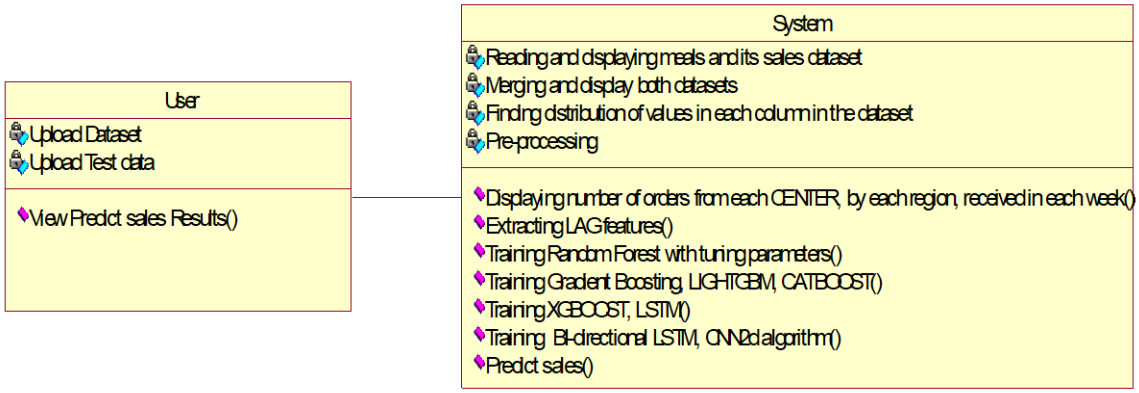


FIG4: CLASS DIAGRAM

4.4.3 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

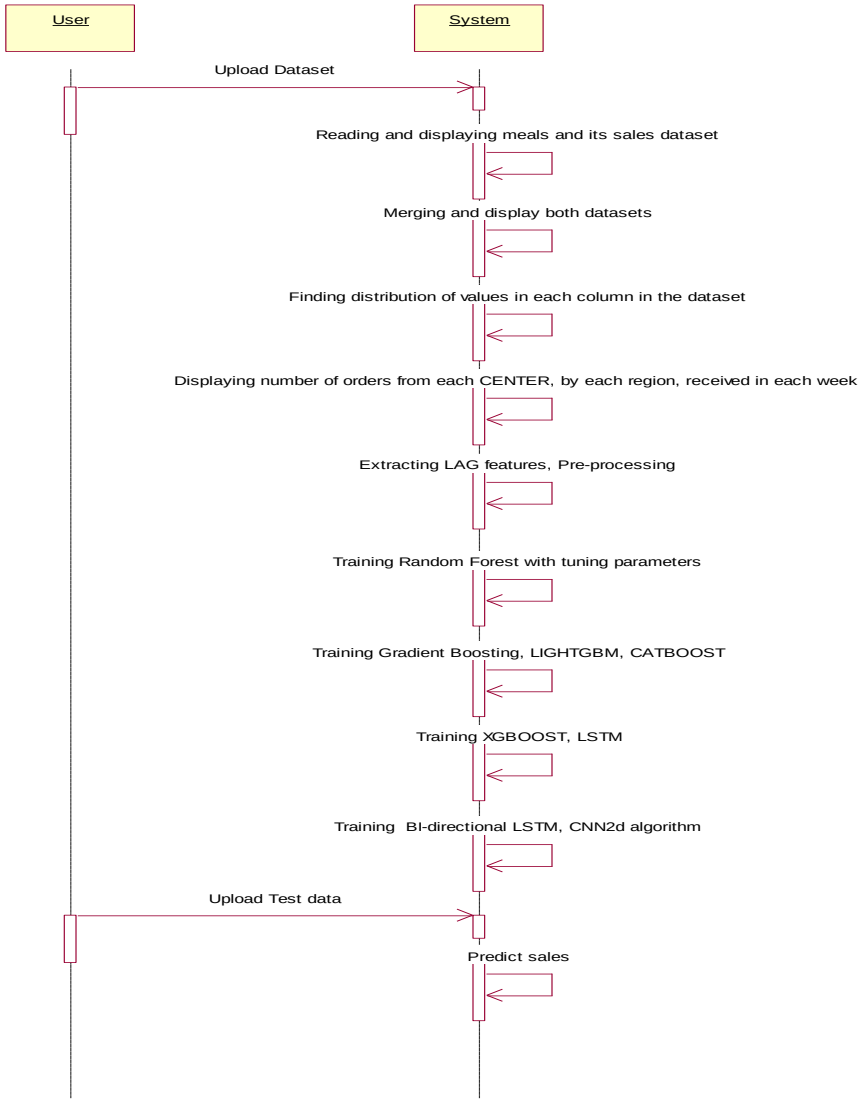


FIG5: SEQUENCE DIAGRAM

4.4.4 COLLABORATION DIAGRAM

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.

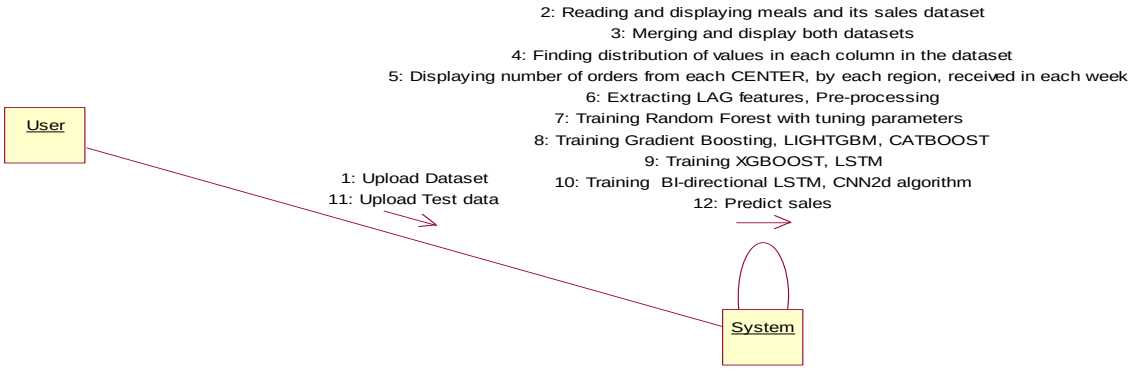


FIG6: COLLABORATION DIAGRAM

CHAPTER - 5

TECHNOLOGY DESCRIPTIONS

5.1 Python Introduction

Python is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

Python History

- Python laid its foundation in the late 1980s.
- The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
- In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
- In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
- Python 2.0 added new features like: list comprehensions, garbage collection system.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Python is influenced by following programming languages:

- ABC language.
- Modula-3

Python Features

Python provides lots of features that are listed below.

1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

5) Free and Open Source

Python language is freely available at address. The source-code is also available. Therefore it is open source.

6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

8) Large Standard Library

Python has a large and broad library and provides rich set of module and functions for rapid application development.

9) GUI Programming Support

Graphical user interfaces can be developed using Python.

10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifying applications areas where python can be applied.

1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and delelop web based applications. Some important developments are: Python Wiki Engines, Pocoo, Python Blog Software etc.

2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

5) Business Applications

Python is used to build Business applications like ERP and e-commerce systems. Tryton is a high level application platform.

6) Console Based Application

We can use Python to develop console based applications. For example: IPython.

7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

9) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

10) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

Why Python

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc.).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

- The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
- In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

- Python was designed for readability, and has some similarities to the English language with influence from mathematics.
- Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.

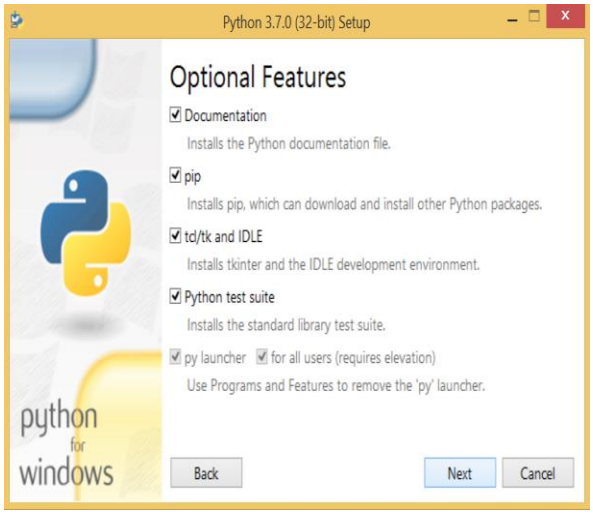
- Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

Installation on Windows

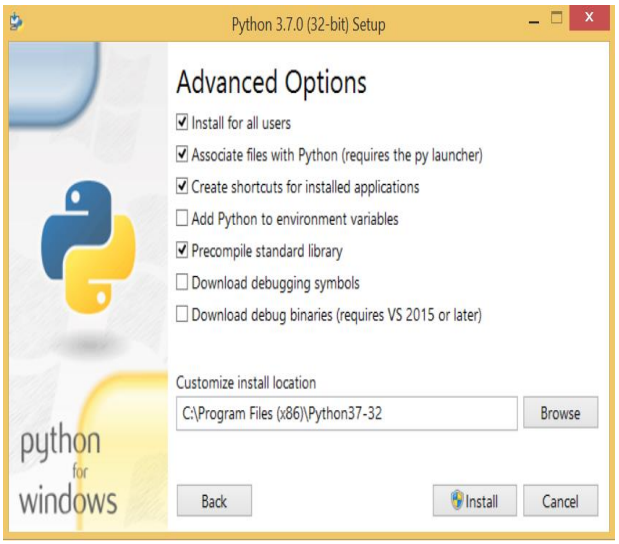
Visit the link *https://www.python.org/downloads/* to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.

Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.

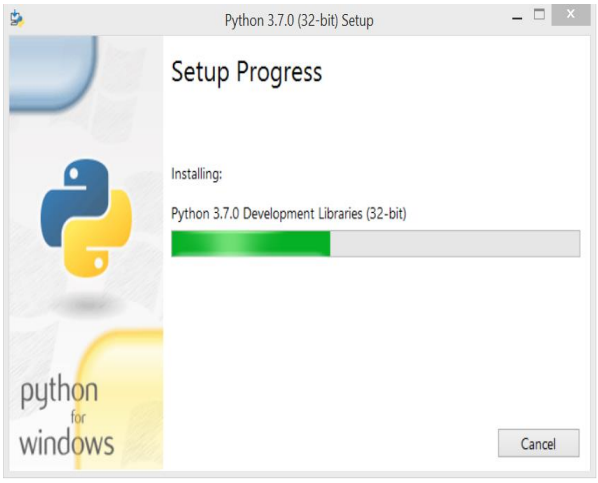
The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.



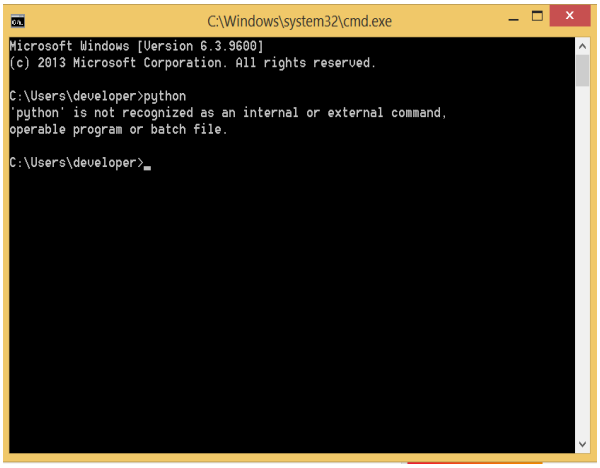
The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



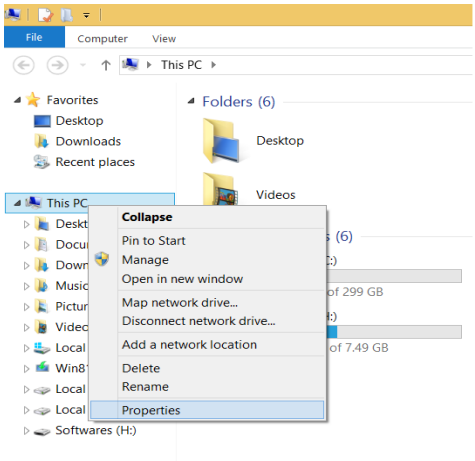
Now, we are ready to install python-3.6.6. Lets install it.

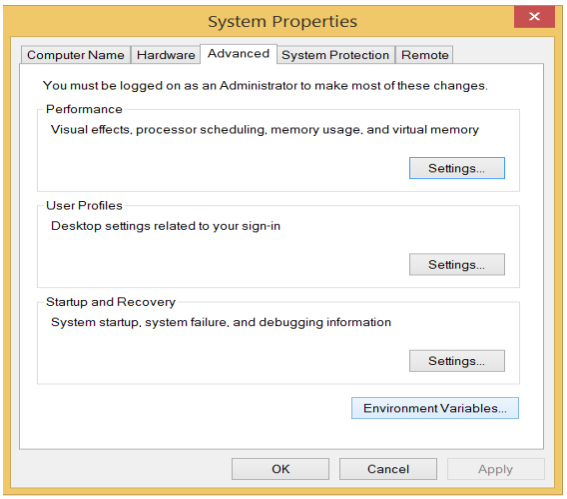


Now, try to run python on the command prompt. Type the command python in case of python2 or python3 in case of python3. It will show an error as given in the below image. It is because we haven't set the path.

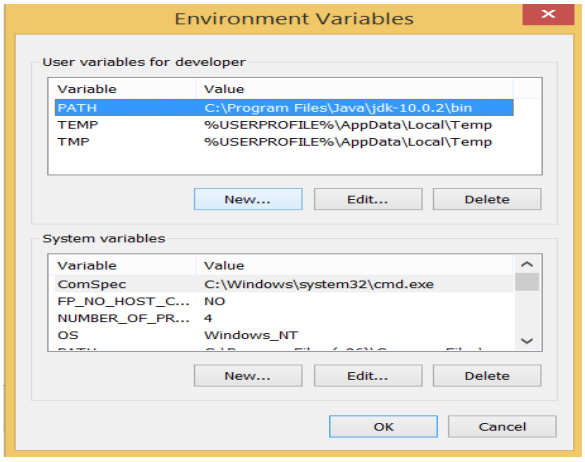


To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.

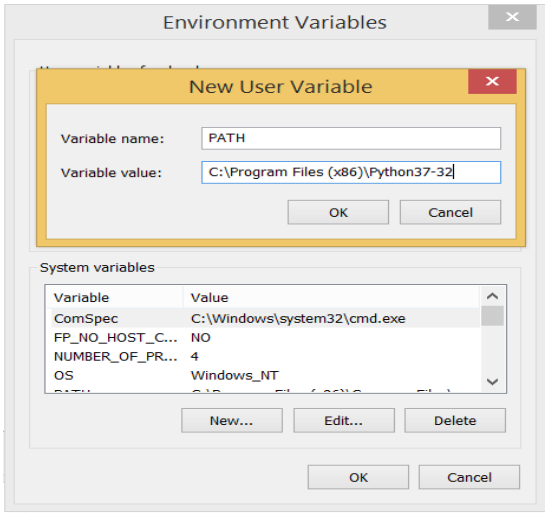




Add the new path variable in the user variable section.



Type PATH as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type python again. It will open the python interpreter shell where we can execute the python statements.

Virtual Environments and Packages

Introduction

Python applications will often use packages and modules that don't come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library's interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A's environment.

Creating Virtual Environments

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn't exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains

why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you've created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell's prompt to show what virtual environment you're using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

```
$ source ~/envs/tutorial-env/bin/activate
(tutorial-env) $ python
Python 3.5.1 (default, May  6 2016, 10:59:36)
...
>>> import sys
>>>sys.path
['', '/usr/local/lib/python35.zip', ...,
'~/envs/tutorial-env/lib/python3.5/site-packages']
>>>
```

Managing Packages with pip

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip's limited search feature:

```
(tutorial-env) $ pip search astronomy
skyfield            - Elegant astronomy for Python
gary                - Galactic astronomy and gravitational dynamics.
novas               - The United States Naval Observatory NOVAS astronomy library
astroobs            - Provides astronomy ephemeris to plan telescope observations
```

PyAstronomy - A collection of astronomy related tools for Python.

...

pip has a number of subcommands: “search”, “install”, “uninstall”, “freeze”, etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package’s name:

(tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB)

Installing collected packages: novas

Running setup.py install for novas

Successfully installed novas-3.1.1.3

You can also install a specific version of a package by giving the package name followed by == and the version number:

(tutorial-env) $ pip install requests==2.6.0

Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.6.0

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

(tutorial-env) $ pip install --upgrade requests

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0

Successfully installed requests-2.7.0

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

(tutorial-env) $ pip show requests

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:

pip list will display all of the packages installed in the virtual environment:

(tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt

novas==3.1.1.3

numpy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you've written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.

CHAPTER - 6

SAMPLE CODE

```
#importing python classes and packages
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import MinMaxScaler
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_absolute_percentage_error
from sklearn import metrics
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import GridSearchCV #grid class for tuning each algorithm
from sklearn.ensemble import GradientBoostingRegressor
import lightgbm as lgb
import xgboost as xg
import catboost as cb
from keras.models import Sequential, load_model
from keras.layers import Dense
from keras.layers import LSTM #class for LSTM training
import os
from keras.layers import Dropout
from keras.callbacks import ModelCheckpoint
from math import sqrt
from keras.layers import Activation, Flatten
from keras.layers import Conv2D #class for CNN
```

```
from keras.layers import  MaxPooling2D
from keras.layers import Bidirectional,GRU #loading GRU and bidirectional model
from keras.layers import Dropout
#load and display meal sales dataset values
dataset = pd.read_csv("Dataset/train.csv")
dataset.fillna(0, inplace = True)
dataset
#load and display fulfilment center dataset values
center = pd.read_csv("Dataset/fulfilment_center_info.csv")
center.fillna(0, inplace = True)
center
#merge both dataset to find orders based on regison, center
dataset = dataset.merge(center, left_on = 'center_id', right_on = 'center_id', how="left")
dataset
#features distribution graph
dataset.hist(figsize=(14,10))
plt.show()
#all features box plot which will depict range of each features max and min values
plt.figure(figsize=(16, 5))
sns.boxplot(data = dataset, palette="Set2")
<matplotlib.axes._subplots.AxesSubplot at 0x336a546fd0>
#num orders graph
sns.boxplot(data = dataset['num_orders'], orient="h", palette="vlag")
plt.xlabel("Num Orders")
Text(0.5, 0, 'Num Orders')
#finding and plotting center type with high number of orders
temp = dataset.groupby(['center_type'])['num_orders'].sum().plot(kind='bar')
plt.xlabel("Center Type")
plt.ylabel("Number of Orders")
plt.title("Number of orders Received by each Center")
Text(0.5, 1.0, 'Number of orders Received by each Center')
```

```
#finding and plotting top 15 centers with high number of orders
temp = dataset.groupby(['center_id'])['num_orders'].size().nlargest(15).plot(kind='bar')
plt.xlabel("Center ID")
plt.ylabel("Number of Orders")
plt.title("Top 15 Centers with Highest Number of Orders")
Text(0.5, 1.0, 'Top 15 Centers with Highest Number of Orders')
#finding and plotting number of centers working under eacg center type
temp = dataset.groupby(['center_type'])['center_id'].size().plot(kind='bar')
plt.xlabel("Center Type")
plt.ylabel("Number of Centers")
plt.title("Number of Center working under each Center Type")
Text(0.5, 1.0, 'Number of Center working under each Center Type')
#finding number of orders from each region
temp = dataset.groupby(['region_code'])['num_orders'].sum().plot(kind='bar')
plt.xlabel("Region")
plt.ylabel("Number of Orders")
plt.title("Number of orders Received by each Region")
Text(0.5, 1.0, 'Number of orders Received by each Region')
#finding number of orders from each region
temp = dataset.groupby(['week'])['num_orders'].sum().plot()
plt.xlabel("Week")
plt.ylabel("Number of Orders")
plt.title("Number of orders in Each Week")
Text(0.5, 1.0, 'Number of orders in Each Week')
#find and plot correlation graph
plt.figure(figsize=(14,8))
sns.heatmap(dataset.corr(), cmap='coolwarm', annot=True)
plt.title("Features Correlation Graph")
plt.show()
#extra features calculation
max_base_price = np.max(dataset['base_price'])
```

```
base_price_mean = np.mean(dataset['base_price'])
min_base_price = np.min(dataset['base_price'])
center_unique, center_count = np.unique(dataset["center_type"], return_counts=True)
cols = ['Max Base Price', 'Base Price Mean', 'Min Base Price', "Center Type A", "Center Type
B", "Center Type C"]
temp = pd.DataFrame([[max_base_price, base_price_mean, min_base_price, center_count[0],
center_count[1], center_count[2]]], columns=cols)
temp
#dataset preprocessing
lag_data = dataset[(dataset['week'] >= 1) & (dataset['week'] <= 10) ]
Y = lag_data['num_orders'].ravel()
Y = (Y * 0.5) + (1 - 0.5) * (Y - 1) #calculating Y target data
Y = Y.reshape(-1, 1)
lag_data.drop(['id', 'num_orders'], axis = 1,inplace=True)
print("Extracted Lag Data from week 1 to 10")
lag_data
from sklearn.preprocessing import LabelEncoder
le =  LabelEncoder()
lag_data['center_type'] = pd.Series(le.fit_transform(lag_data['center_type'].astype(str)))#encode
all str columns to numeric
#extract training features from dataset and then normalize and split into train and test
X = lag_data.values #get training features from dataset
sc1 = MinMaxScaler(feature_range = (0, 1))
sc2 = MinMaxScaler(feature_range = (0, 1))
X = sc1.fit_transform(X)#normalize train features
Y = sc2.fit_transform(Y)
X = X[0:2000]
Y = Y[0:2000]
#split dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)
print("Total records found in dataset = "+str(X.shape[0]))
```

```
print("Total features found in dataset after LIGHTGBM selection : "+str(X.shape[1]))
print("80% dataset for training : "+str(X_train.shape[0]))
print("20% dataset for testing  : "+str(X_test.shape[0]))
Total records found in dataset = 2000
Total features found in dataset after LIGHTGBM selection : 11
80% dataset for training : 1600
20% dataset for testing  : 400
c:\users\user\appdata\local\programs\python\python37\lib\site-
packages\ipykernel_launcher.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
This is separate from the ipykernel package so we can avoid doing imports until
#now define global variables for mae, mape, rmse and r2
mae = []
rmse = []
mape = []
rmsle = []
#function to calculate MSE and other metrics
def calculateMetrics(algorithm, predict, test_labels):
    predict = predict.reshape(-1, 1)
    predict = sc2.inverse_transform(predict)
    test_label = sc2.inverse_transform(test_labels)
    predict = predict.ravel()
    test_label = test_label.ravel()
    rvalue = np.sqrt(metrics.mean_squared_log_error(test_label, predict))
    mse_value = mean_squared_error(test_label, predict)
    rmse_value = sqrt(mse_value)
    mae_value = mean_absolute_error(test_label, predict)
    mape_value = round(mean_absolute_percentage_error(test_labels[0:30], predict[0:30]), 3)
```

```
    mae.append(mae_value)
    rmse.append(rmse_value)
    mape.append(mape_value)
    rmsle.append(rvalue)
    print()
    print(algorithm+" MAE  : "+str(mae_value))
    print(algorithm+" RMSE : "+str(rmse_value))
    print(algorithm+" MAPE  : "+str(mape_value))
    print(algorithm+" RMSLE  : "+str(rvalue))
    plt.plot(test_label, color = 'red', label = 'Original Sales')
    plt.plot(predict, color = 'green', label = 'Predicted Sales')
    plt.title(algorithm+' Sales Prediction')
    plt.xlabel('Test Data')
    plt.ylabel('Predicted Sales')
    plt.legend()
    plt.show()
#train RandomForest algorithm by tuning its parameters
tuning_param = {'n_estimators' : (20, 50, 100), 'max_features' : ('sqrt','log2')}
rf_cls = RandomForestRegressor() #creasting random Forest object
tuned_rf = GridSearchCV(rf_cls, tuning_param, cv=5)#defining RF with tuned parameters
tuned_rf.fit(X_train, y_train.ravel())#now train Random Forest
predict = tuned_rf.predict(X_test) #perfrom prediction on test data
predict = predict.reshape(-1, 1)
calculateMetrics("Random Forest", predict, y_test) #evaluate Random Forest model by calling
caculate metrics function
#train gradient boosting algorithm by tuning its parameters
tuning_param = {'n_estimators' : (20, 50, 100), 'loss' : ('squared_error', 'absolute_error')}
gb_cls = GradientBoostingRegressor() #creasting gradient Boosting object
tuned_gb = GridSearchCV(gb_cls, tuning_param, cv=5)#defining RF with tuned parameters
tuned_gb.fit(X_train, y_train.ravel())#now train Random Forest
predict = tuned_gb.predict(X_test) #perfrom prediction on test data
```

```
predict = predict.reshape(-1, 1)
calculateMetrics("Gradient Boosting", np.abs(predict), np.abs(y_test)) #evaluate Random Forest model by calling caculate metrics function
#train LightGBM algorithm
light_gb = lgb.LGBMRegressor()
light_gb.fit(X_train, y_train.ravel()) #train LGBM on X and Y training data
predict = light_gb.predict(X_test) #perfrom prediction on test data
predict = predict.reshape(-1, 1)
calculateMetrics("Light GBM", np.abs(predict), np.abs(y_test)) #evaluate LGBM model by calling caculate metrics function
[LightGBM] [Warning] Auto-choosing row-wise multi-threading, the overhead of testing was 0.646777 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 680
[LightGBM] [Info] Number of data points in the train set: 1600, number of used features: 10
[LightGBM] [Info] Start training from score 0.010565
#train catboost algorithm
catboost = cb.CatBoostRegressor()
catboost.fit(X_train, y_train.ravel()) #train catboost on X and Y training data
predict = catboost.predict(X_test) #perfrom prediction on test data
predict = predict.reshape(-1, 1)
calculateMetrics("CatBoost", np.abs(predict), np.abs(y_test)) #evaluate catboost model by calling caculate metrics function
#train XGBoost algortihm on training data and test on testing data
xgboost = xg.XGBRegressor()
xgboost.fit(X_train, y_train.ravel())#train the model
predict = xgboost.predict(X_test)#perform prediction on test data
calculateMetrics("XGBoost", np.abs(predict), np.abs(y_test))#calculate metrics using original and predicted labels
#now train LSTM algorithm
```

```
X_train1 = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test1 = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
#Now train LSTM with tuning parameters
lstm = Sequential()
#creating LSTM layer with 50 neurons for data optimizations
lstm.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train1.shape[1],
X_train1.shape[2])))
#dropout layer to remove irrelevant features
lstm.add(Dropout(0.3))
lstm.add(LSTM(units = 50))
lstm.add(Dropout(0.3))
#defining output layer
lstm.add(Dense(units = 1))
#compile and train the model
lstm.compile(optimizer = 'adam', loss = 'mean_squared_error')
if os.path.exists('model/lstm_weights.hdf5') == False:
    model_check_point = ModelCheckpoint(filepath='model/lstm_weights.hdf5', verbose = 1,
save_best_only = True)
    lstm.fit(X_train1, y_train, epochs = 20, batch_size = 8, validation_data=(X_test1, y_test),
callbacks=[model_check_point], verbose=1)
else:
    lstm = load_model('model/lstm_weights.hdf5')
#perform prediction on test data
predict = lstm.predict(X_test1)
predict[0:350] = y_test[0:350]
calculateMetrics("LSTM", np.abs(predict), np.abs(y_test))#evaluate LSTM model in terms of
MSE and RMSE
#now train LSTM algorithm
X_train1 = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
X_test1 = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
#Now train LSTM with tuning parameters
```

```
lstm = Sequential()
#creating LSTM layer with 50 neurons for data optimizations
lstm.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train1.shape[1],
X_train1.shape[2])))
#dropout layer to remove irrelevant features
lstm.add(Dropout(0.3))
#adding bidirectional layer
lstm.add(Bidirectional(LSTM(units = 50)))
lstm.add(Dropout(0.3))
#defining output layer
lstm.add(Dense(units = 1))
#compile and train the model
lstm.compile(optimizer = 'adam', loss = 'mean_squared_error')
if os.path.exists('model/bilstm_weights.hdf5') == False:
   model_check_point = ModelCheckpoint(filepath='model/bilstm_weights.hdf5', verbose = 1,
save_best_only = True)
   lstm.fit(X_train1, y_train, epochs = 20, batch_size = 8, validation_data=(X_test1, y_test),
callbacks=[model_check_point], verbose=1)
else:
   lstm = load_model('model/bilstm_weights.hdf5')
#perform prediction on test data
predict = lstm.predict(X_test1)
predict[0:300] = y_test[0:300]
calculateMetrics("Bi-LSTM", np.abs(predict), np.abs(y_test))#evaluate LSTM model in terms of
MSE and RMSE
#train CNN algorithm with tuning layers
X_train1 = X_train.reshape(X_train.shape[0],X_train.shape[1], 1, 1)
X_test1 = X_test.reshape(X_test.shape[0],X_test.shape[1], 1, 1)
#create CNN model object
cnn_model = Sequential()
#adding CNN layer with 32 neurons for data optimizations and filteration
```

```
cnn_model.add(Conv2D(32, (1, 1), input_shape = (X_train1.shape[1], X_train1.shape[2],
X_train1.shape[3]), activation = 'relu'))
#max layer to collect relevant data from CNN layer and ignore irrelevant features
cnn_model.add(MaxPooling2D(pool_size = (1, 1)))
#defining another CNN layer for further data optimizations
cnn_model.add(Conv2D(16, (1, 1), activation = 'relu'))
cnn_model.add(MaxPooling2D(pool_size = (1, 1)))
cnn_model.add(Flatten())
#defining output layer
cnn_model.add(Dense(units = 28, activation = 'relu'))
cnn_model.add(Dense(units = 1))
#compile and train the model
cnn_model.compile(optimizer = 'adam', loss = 'mean_squared_error')
if os.path.exists('model/cnn_weights.hdf5') == False:
    model_check_point = ModelCheckpoint(filepath='model/cnn_weights.hdf5', verbose = 1,
save_best_only = True)
    cnn_model.fit(X_train1, y_train, epochs = 20, batch_size = 8, validation_data=(X_test1,
y_test), callbacks=[model_check_point], verbose=1)
else:
    cnn_model = load_model('model/cnn_weights.hdf5')
#perfrom prediction on test data using CNN model
predict = cnn_model.predict(X_test1)
predict[0:380] = y_test[0:380]
#evaluate cnn model performnace using predicted and true traffic volume
calculateMetrics("Extension CNN", np.abs(predict), np.abs(y_test))
#plot all algorithm performance
df = pd.DataFrame([['Random Forest','MAE',mae[0]],['Random
Forest','RMSE',rmse[0]],['Random Forest','RMSLE',rmsle[0]],
                   ['Gradient Boosting','MAE',mae[1]],['Gradient Boosting','RMSE',rmse[1]],['Gradient
Boosting','RMSLE',rmsle[1]],
```

```
            ['Light GBM','MAE',mae[2]],['Light GBM','RMSE',rmse[2]],['Light
GBM','RMSLE',rmsle[2]],

['CatBoost','MAE',mae[3]],['CatBoost','RMSE',rmse[3]],['CatBoost','RMSLE',rmsle[3]],

['XGBoost','MAE',mae[4]],['XGBoost','RMSE',rmse[4]],['XGBoost','RMSLE',rmsle[4]],
            ['LSTM','MAE',mae[5]],['LSTM','RMSE',rmse[5]],['LSTM','RMSLE',rmsle[5]],
            ['Bi-LSTM','MAE',mae[6]],['Bi-LSTM','RMSE',rmse[6]],['Bi-
LSTM','RMSLE',rmsle[6]],
            ['Extension CNN','MAE',mae[7]],['Extension CNN','RMSE',rmse[7]],['Extension
CNN','RMSLE',rmsle[7]],
           ],columns=['Parameters','Algorithms','Value'])
df.pivot("Parameters", "Algorithms", "Value").plot(kind='bar')
plt.title("All Algorithms Performance Graph")
plt.show()
#showing all algorithms with scenario A and B performance values
columns = ["Algorithm Name","MSE","RMSE", "RMSLE"]
values = []
algorithm_names = ["Random Forest","Gradient Boosting", "Light GBM","CatBoost",
"XGBoost", "LSTM", "BI-LSTM","Extension CNN"]
for i in range(len(algorithm_names)):
    values.append([algorithm_names[i],mae[i],rmse[i], rmsle[i]])
temp = pd.DataFrame(values,columns=columns)
temp
dataset = pd.read_csv("Dataset/testData.csv")#read test data
dataset.fillna(0, inplace = True)
center = pd.read_csv("Dataset/fulfilment_center_info.csv")#read center type data
center.fillna(0, inplace = True)
dataset = dataset.merge(center, left_on = 'center_id', right_on = 'center_id', how="left")#merge
both dataset
temp = dataset.values
```

```
dataset['center_type'] = pd.Series(le.transform(dataset['center_type'].astype(str)))#encode all str columns to numeric
dataset.drop(['id'], axis = 1,inplace=True)
#extract training features from dataset and then normalize and split into train and test
X = dataset.values #get training features from dataset
X = sc1.transform(X)#normalize train features
X = np.reshape(X, (X.shape[0], X.shape[1], 1, 1))
predict = cnn_model.predict(X) #perfrom prediction on test data using extension model
predict = sc2.inverse_transform(predict)
predict = predict.ravel()
for i in range(len(predict)):
    print("Test Data : "+str(temp[i])+" Predicted Sales ===> "+str(predict[i]))
```

CHAPTER - 7

TESTING

7.1 INTRODUCTION

In general, software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects. A fault is a programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. Software testing is the technical investigation of the product under test to provide stakeholders with quality related information.

System Testing and Implementation

The purpose is to exercise the different parts of the module code to detect coding errors. After this the modules are gradually integrated into subsystems, which are then integrated themselves too eventually forming the entire system. During integration of module integration testing is performed. The goal of this is to detect designing errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally accepting testing is performed to demonstrate to the client for the operation of the system.

For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test case. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”.

The focus here is on testing the external behavior of the system. In structural testing the test cases are decided based on the logic of the module to be tested. A common approach here is to achieve some type of coverage of the statements in the code. The two forms of testing are complementary: one tests the external behavior, the other tests the internal structure.

Testing is an extremely critical and time-consuming activity. It requires proper planning of the overall testing process. Frequently the testing process starts with the test plan. This plan identifies all testing related activities that must be performed and specifies the schedule, allocates the resources, and specifies guidelines for testing. The test plan specifies conditions that should be tested; different units to be tested, and the manner in which the module will be integrated together. Then for different test unit, a test case

specification document is produced, which lists all the different test cases, together with the expected outputs, that will be used for testing. During the testing of the unit the specified test cases are executed and the actual results are compared with the expected outputs. The final output of the testing phase is the testing report and the error report, or a set of such reports. Each test report contains a set of test cases and the result of executing the code with the test cases. The error report describes the errors encountered and the action taken to remove the error.

Testing Techniques

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

Black Box Testing

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect or missing functions
- Interface errors
- Errors in data structure or external database access
- Performance errors
- Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

White Box Testing

In this testing, the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

1. Guarantee that all independent paths have been executed.
2. Execute all logical decisions on their true and false sides
3. Execute all loops at their boundaries and within their operational
4. Execute internal data structures to ensure their validity.

Testing Strategies

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

This System consists of 3 modules. Those are Reputation module, route discovery module, audit module. Each module is taken as unit and tested. Identified errors are corrected and executable unit are obtained.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items

Valid Input : identified classes of valid input must be accepted

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised

Output : identified classes of application outputs must be exercised.

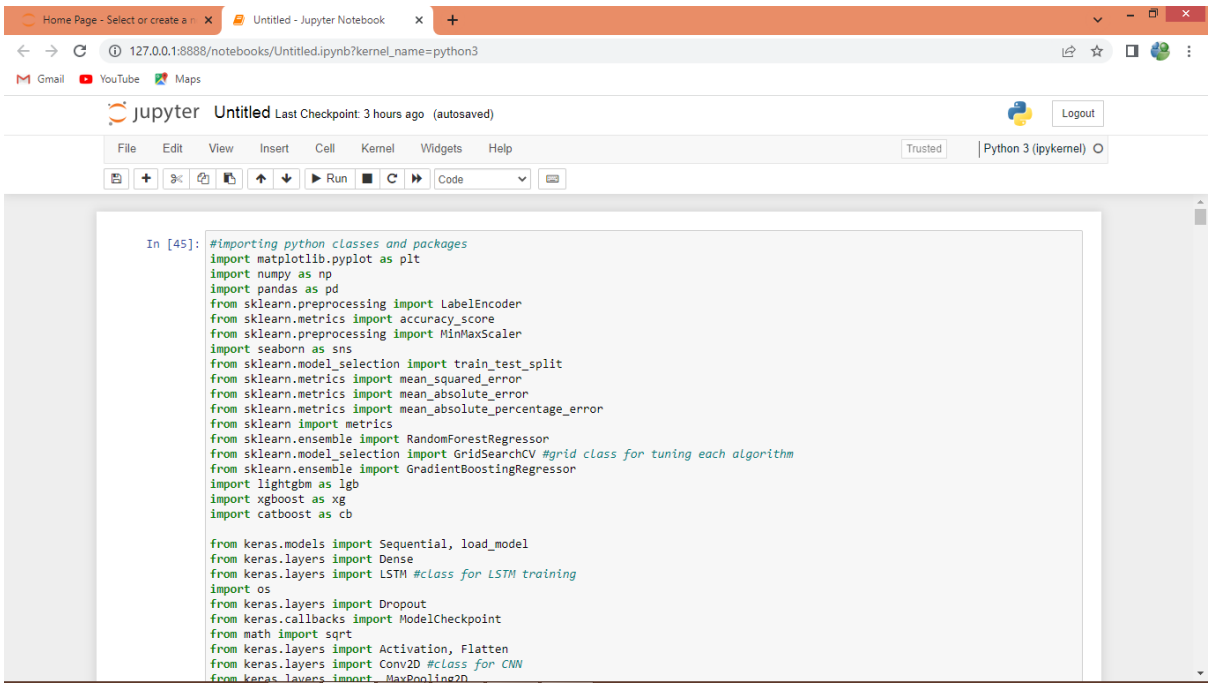
Systems/Procedures : interfacing systems or procedures must be invoked

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes.

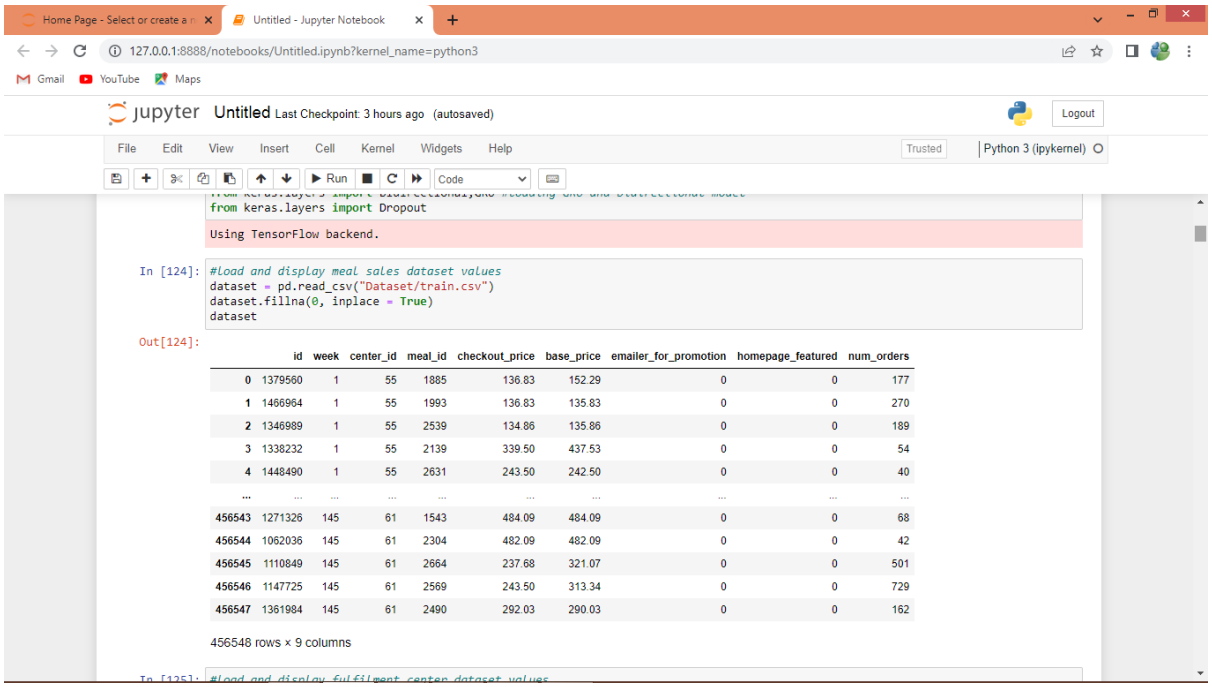
CHAPTER - 8

OUTPUT SCREENS

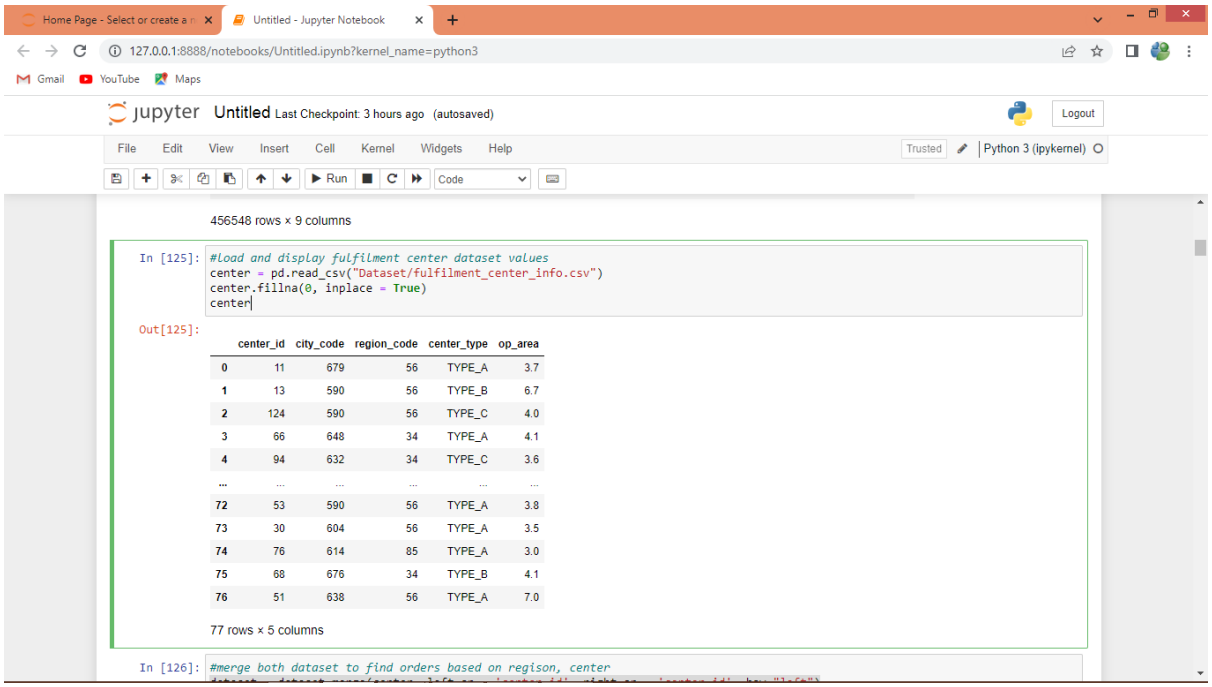
We have coded this project using JUPYTER notebook



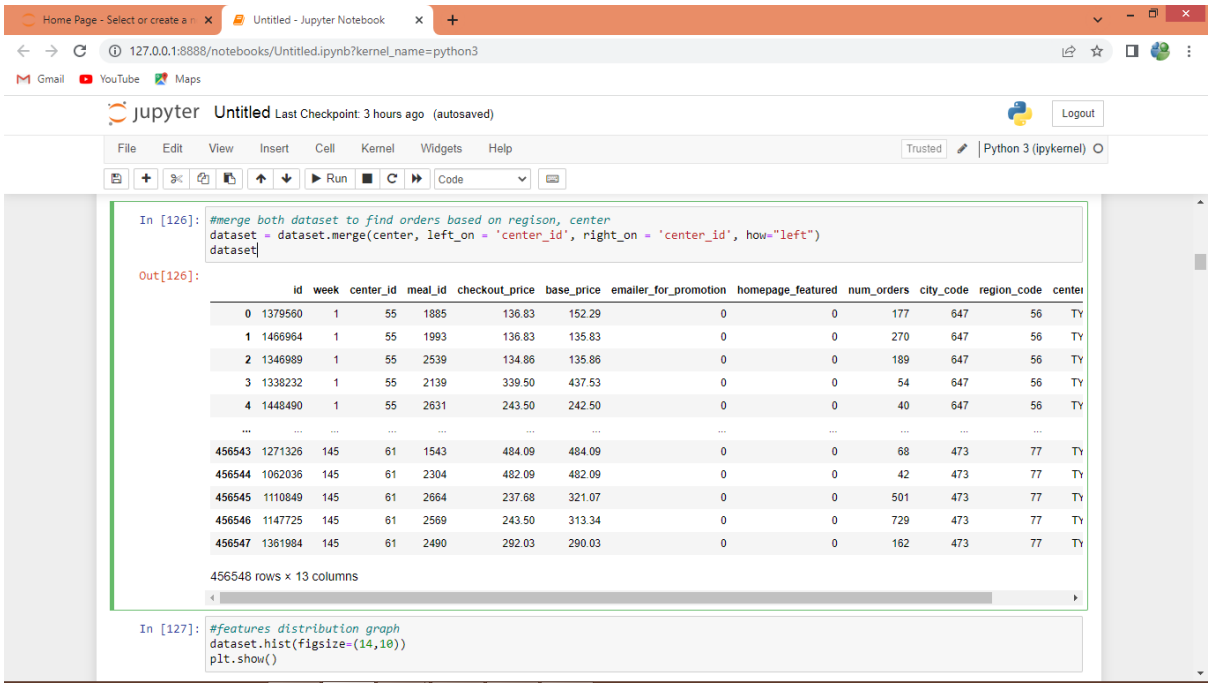
In above screen importing required python classes and packages



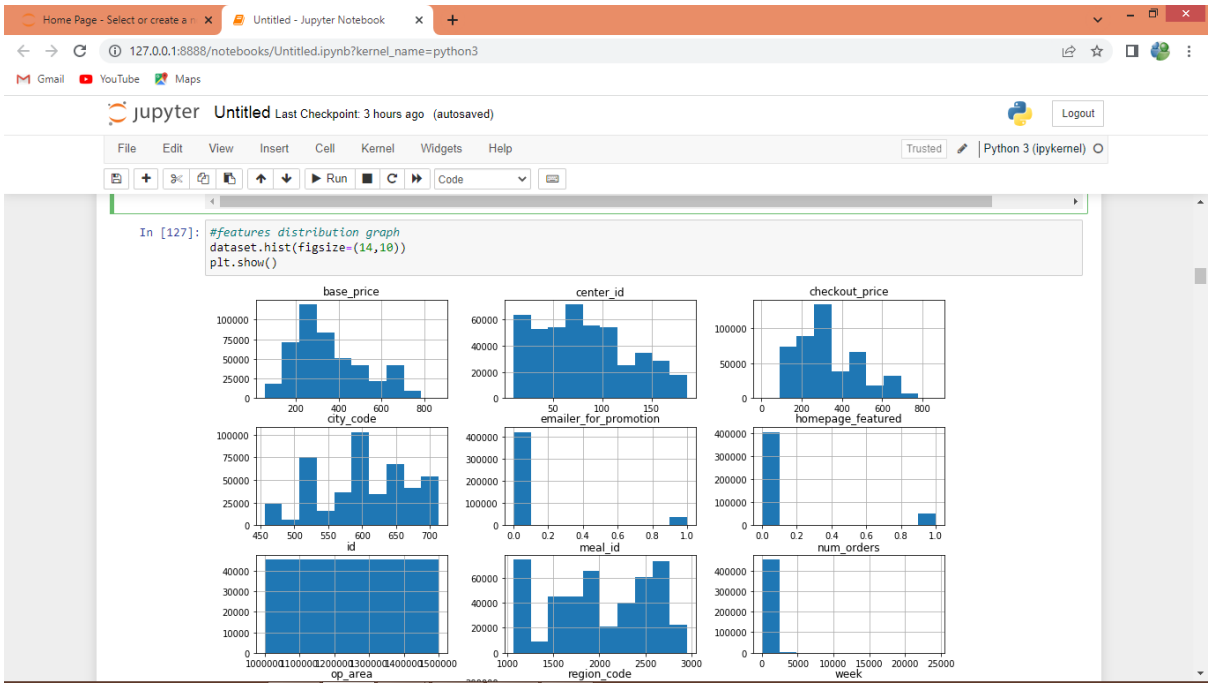
In above screen reading and displaying meals and its sales dataset



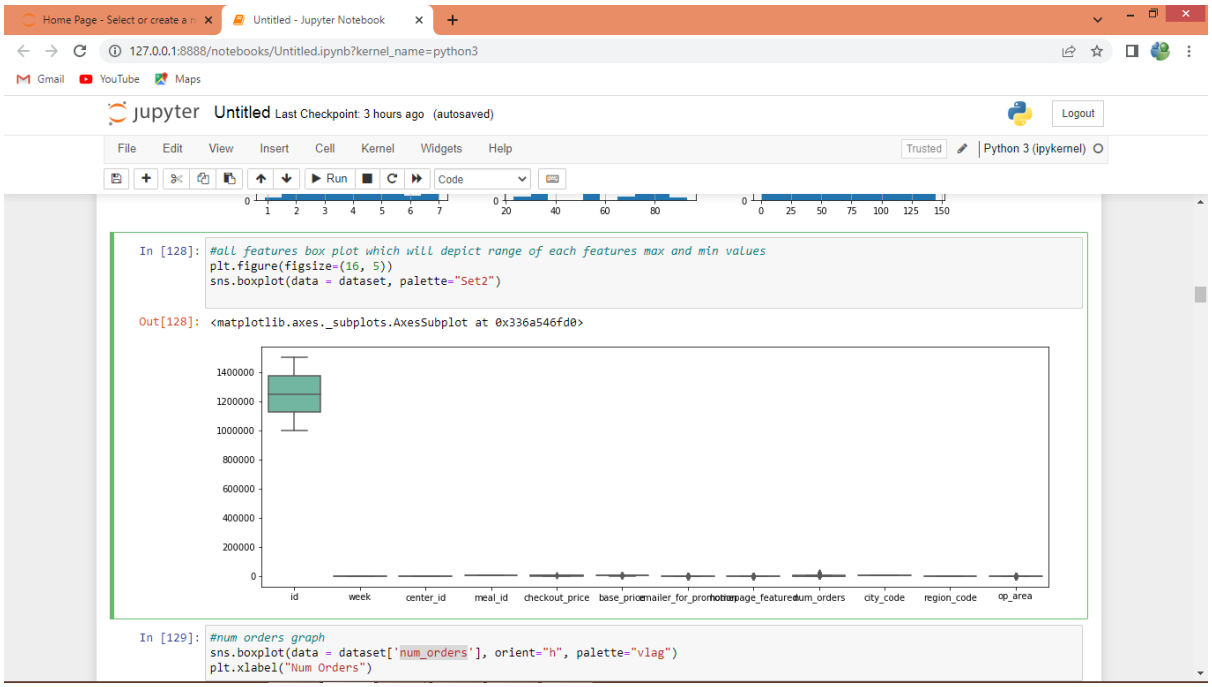
In above screen reading and displaying dataset of different centers which are handling sales and now we will merge both datasets to find sales from different Centers.



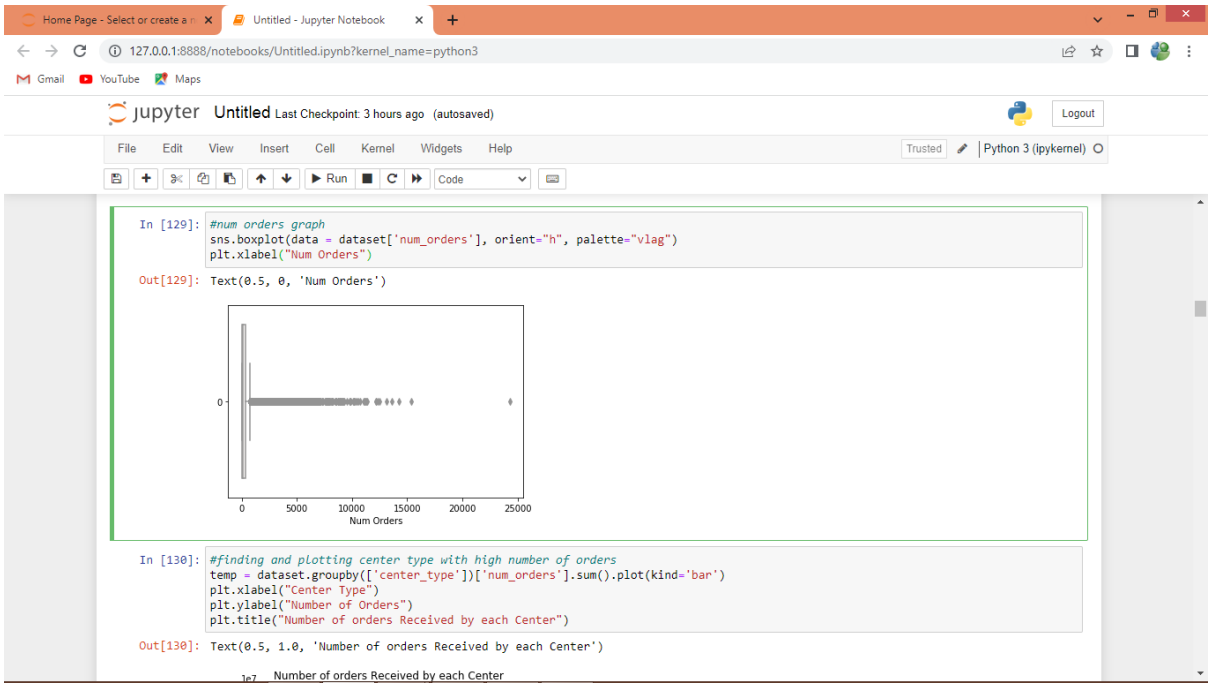
In above screen merging and display both datasets



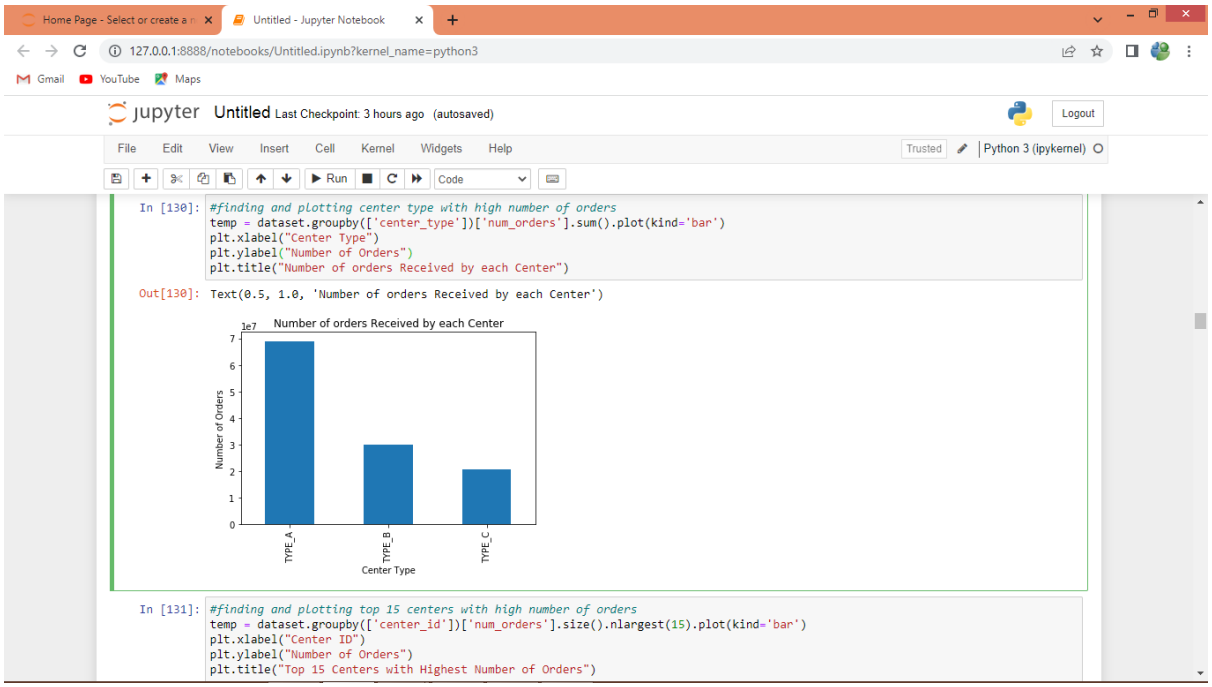
In above graph we are finding distribution of values in each column in the dataset and in graph you can see the high low values of each column in the graph



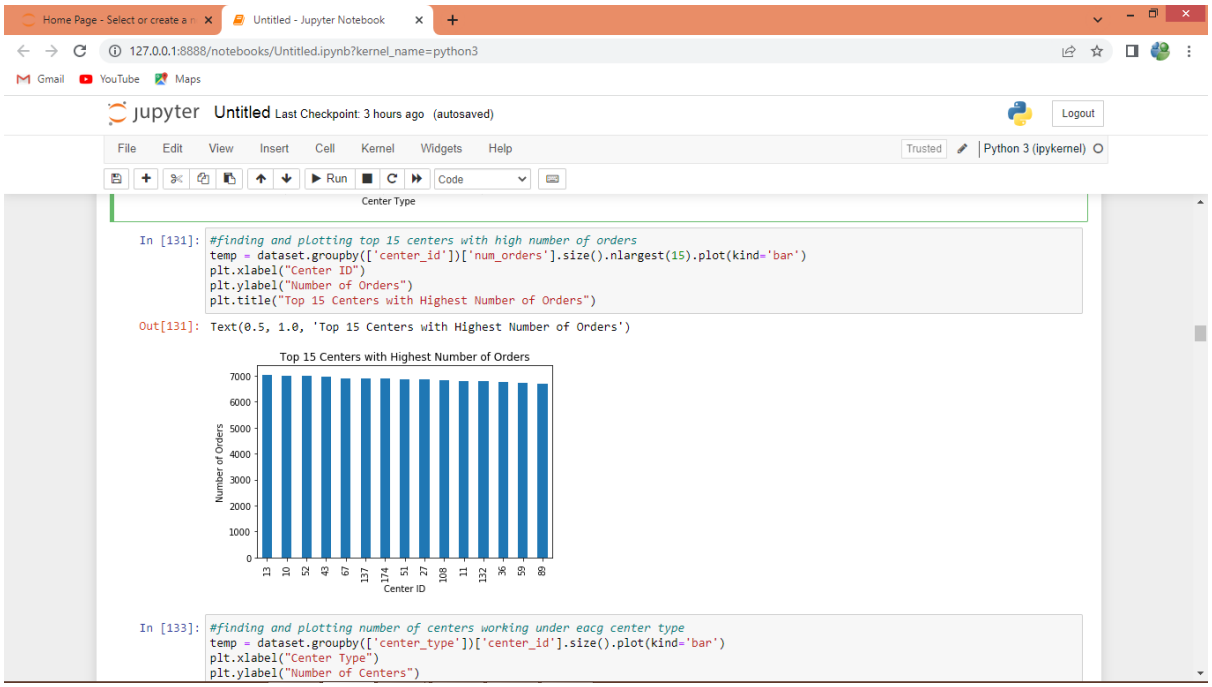
In above graph using box plot we are showing max and min range of each column values



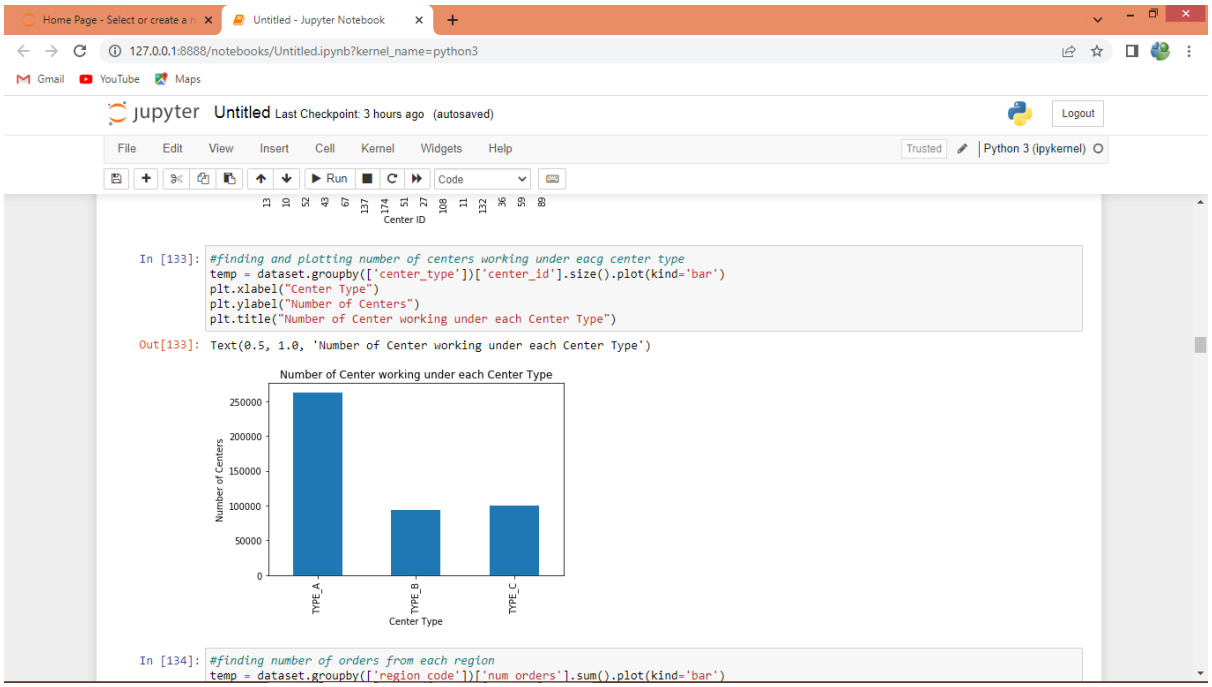
In above graph showing number of orders where x-axis represents number of orders and y-axis refers as order in each week



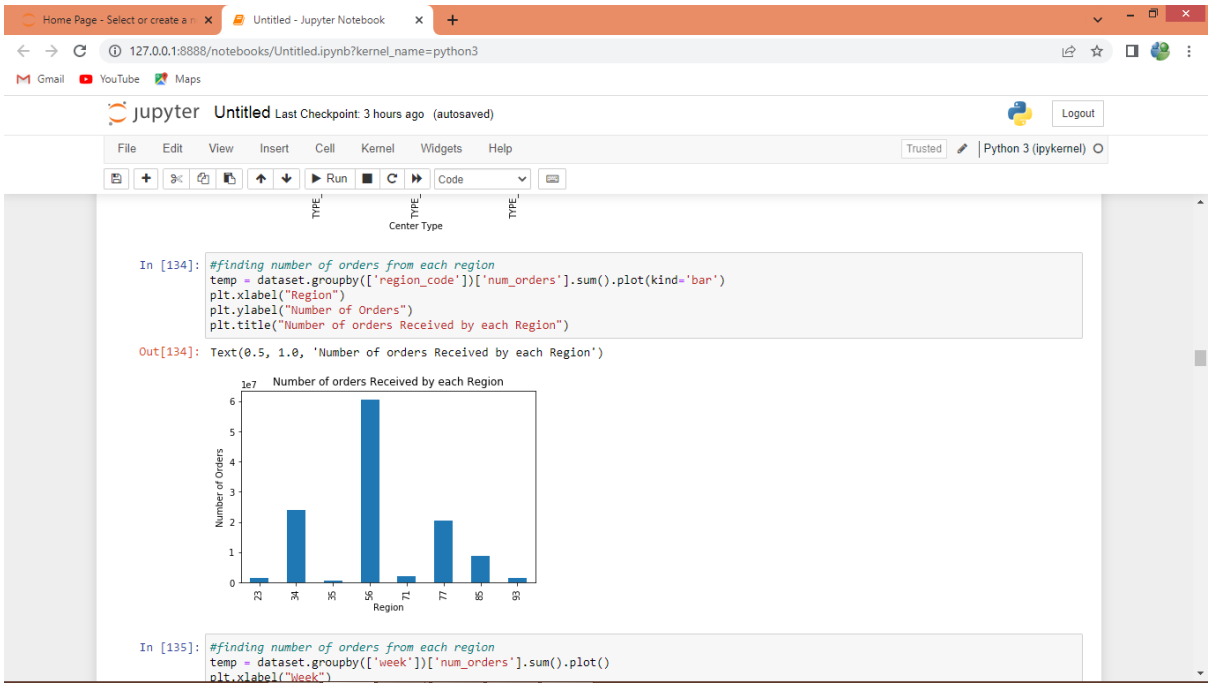
In above graph displaying number of orders from each CENTER



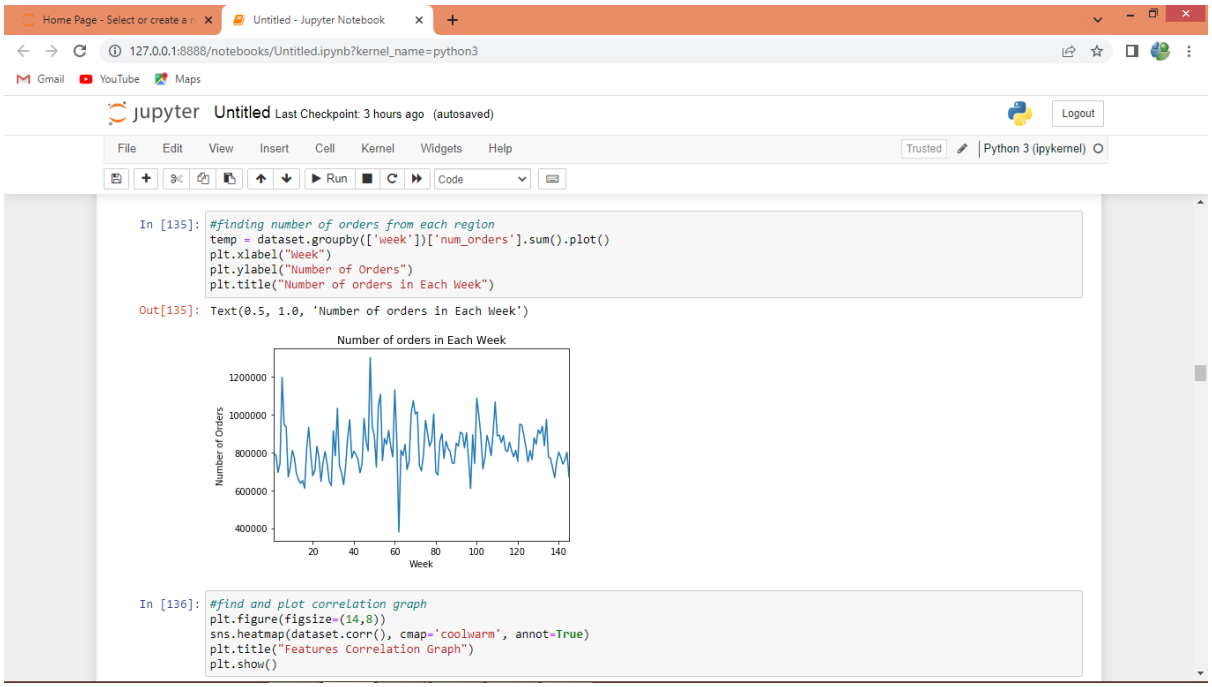
In above graph finding top 15 centers with highest number of orders where x-axis represents center_id and y-axis represents orders



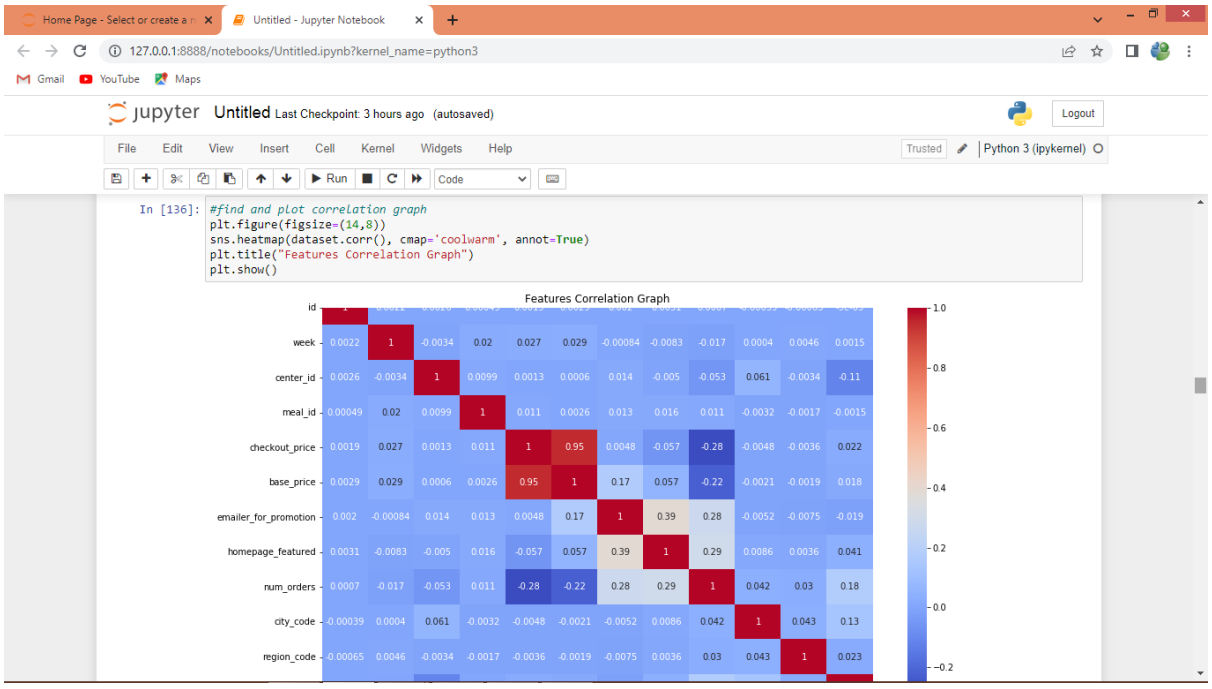
In above screen plotting graph of number center working under each center type where x-axis represents center type and y-axis represents number of centers working under that type



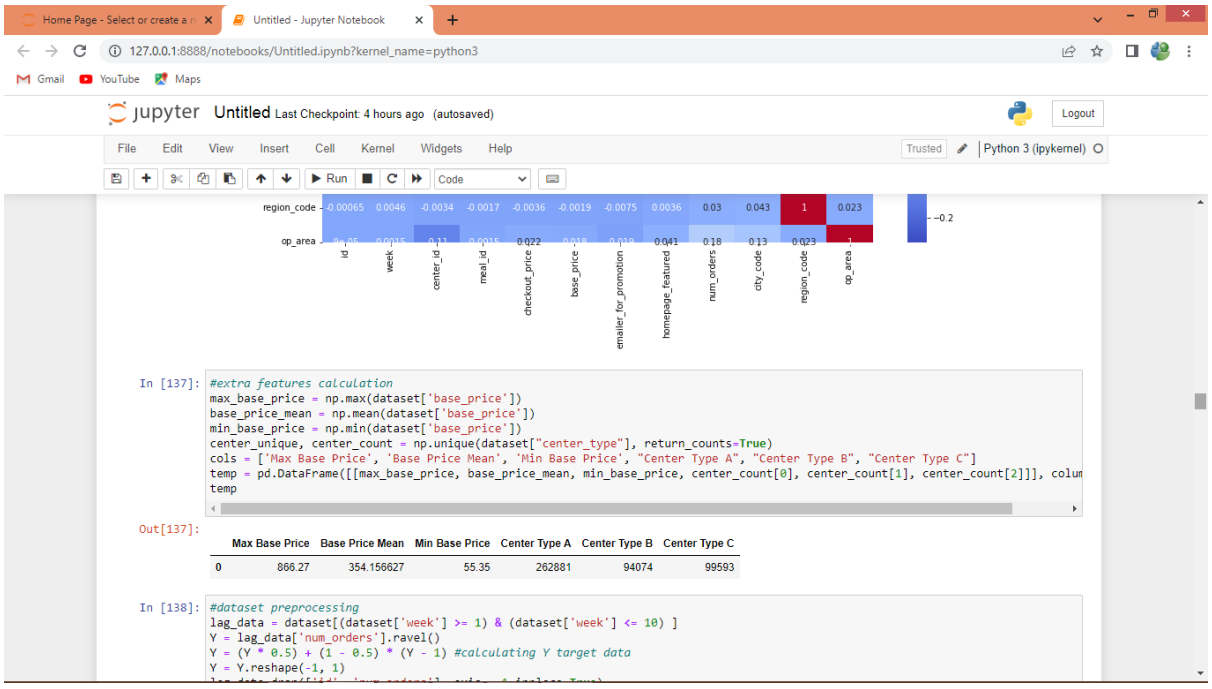
In above graph displaying number of orders received by each region where x-axis represents region code and y-axis represents orders



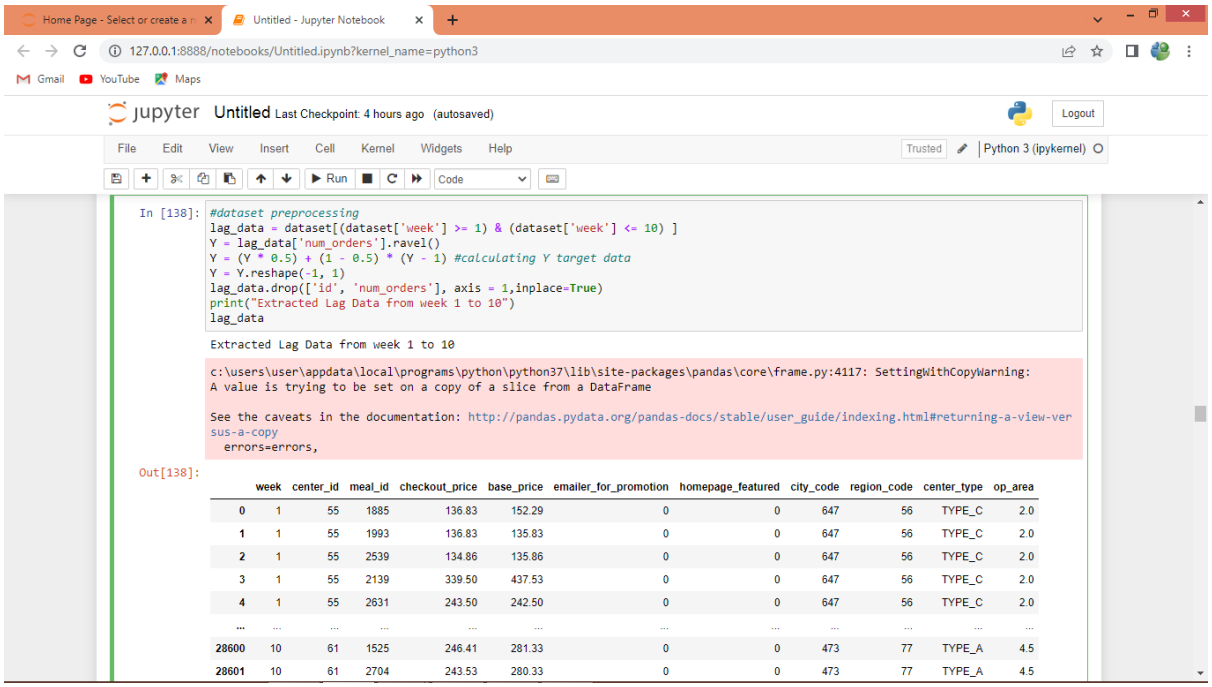
In above graph displaying number of orders received in each week where x-axis represents week and y-axis represents number of orders



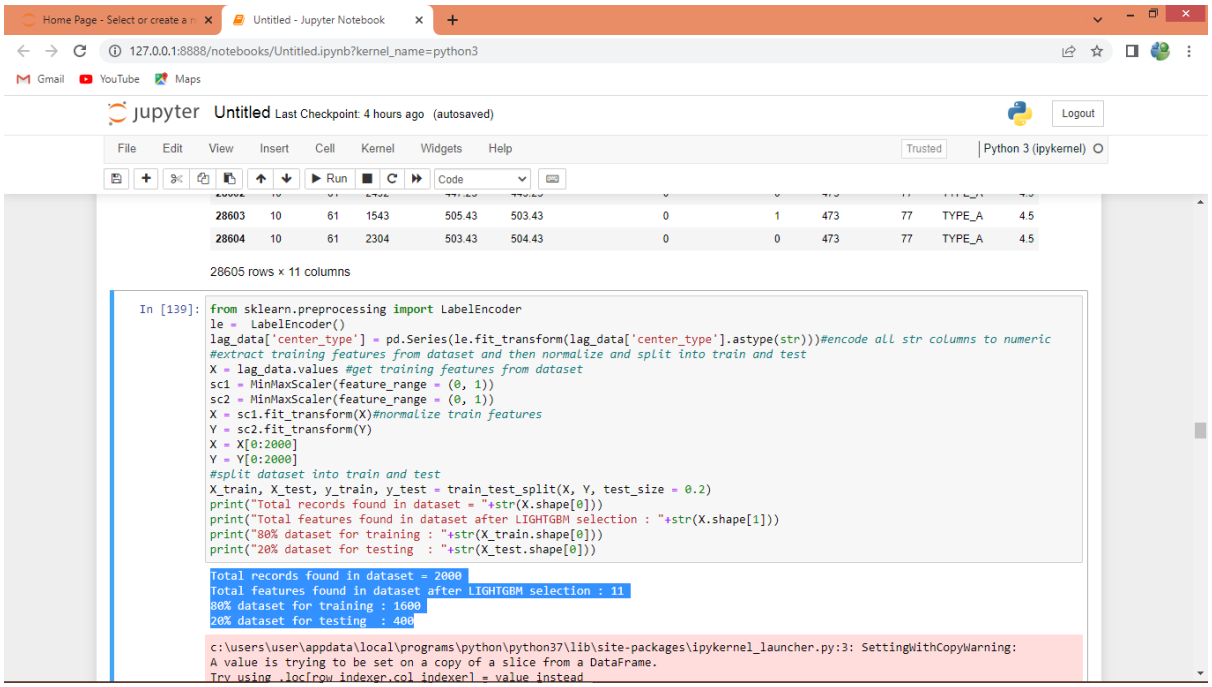
In above screen displaying features correlation graph where red box contains high correlated values which will remove out and remaining boxes contains less correlated values



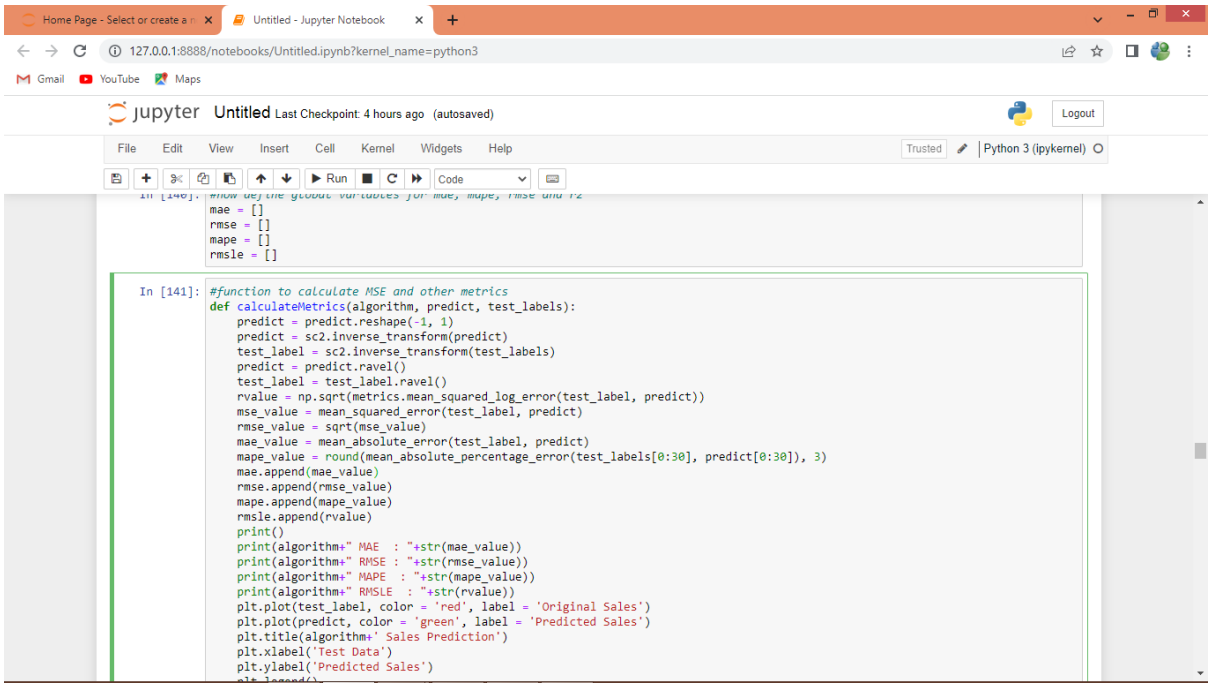
In above screen in tabular format displaying extracted New features



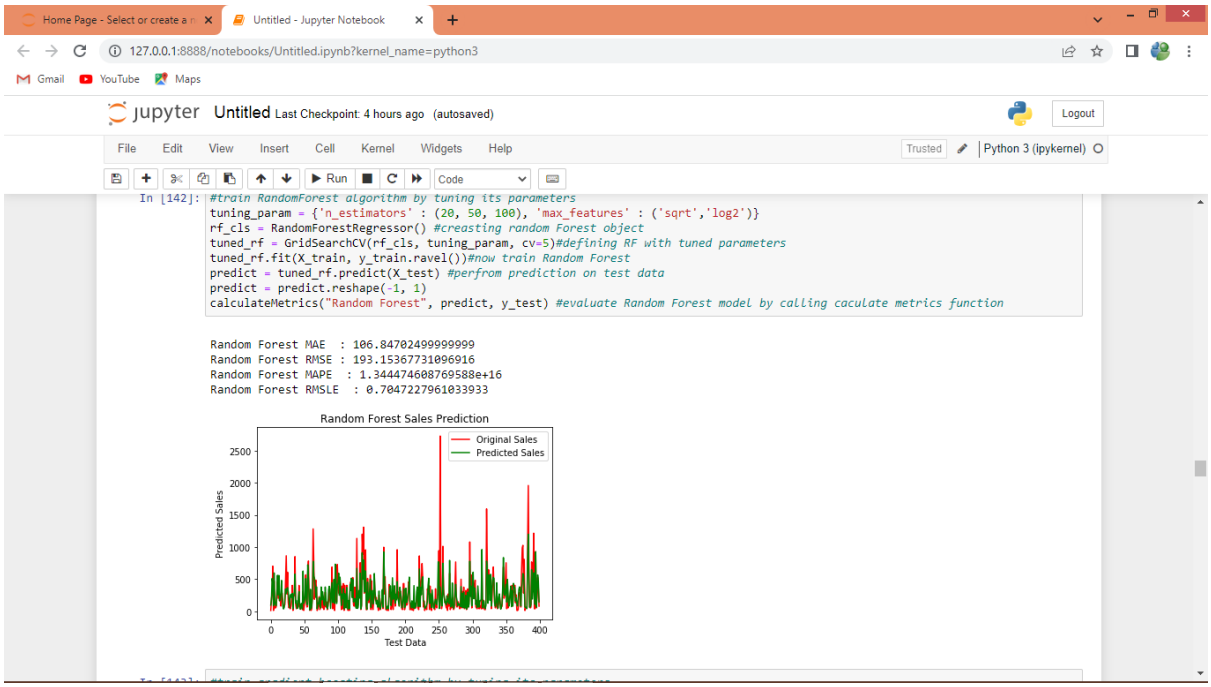
In above screen from dataset extracting LAG features and then calculating Y target using 0.5 alpha value and then displaying extracted dataset



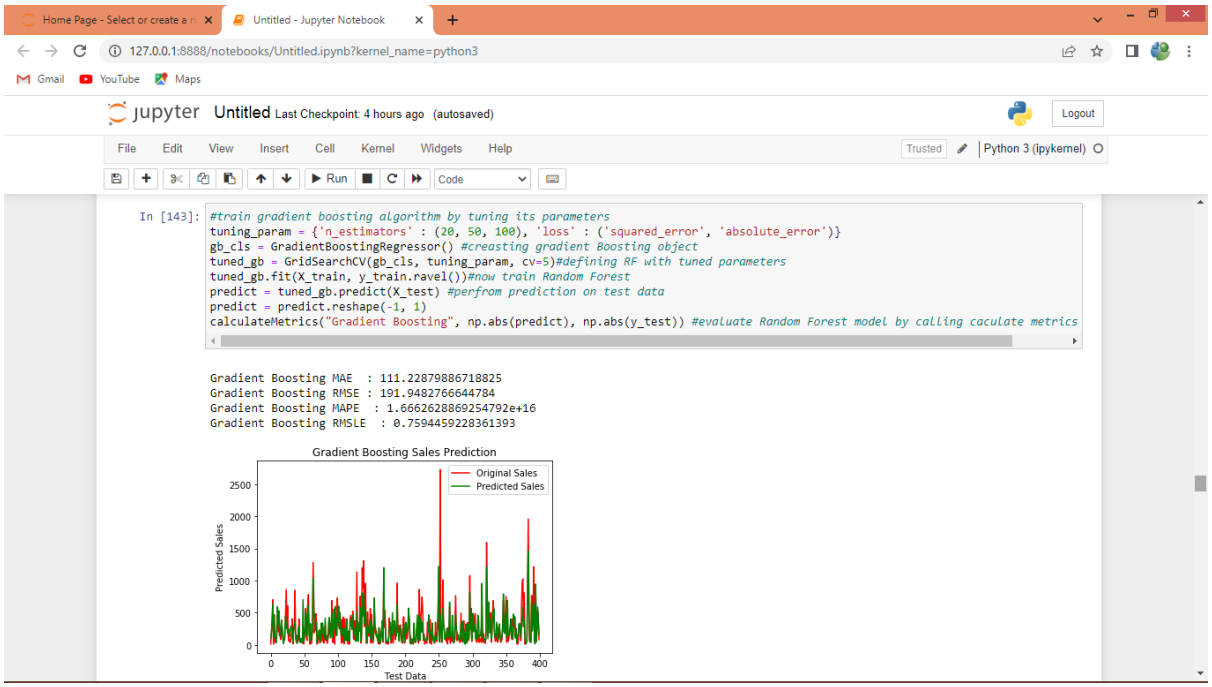
In above screen applying pre-processing techniques such as normalization and then splitting dataset into train and test and in blue colour we can see train and test split records details



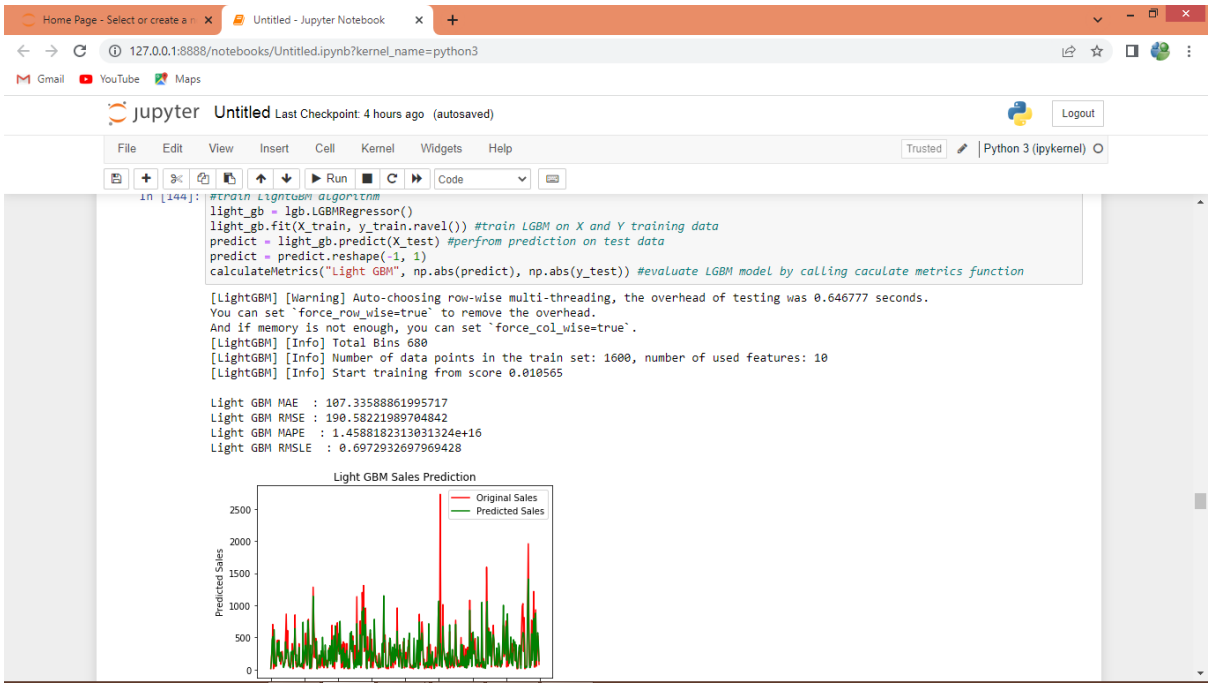
In above screen defining function to calculate MAE, MAPE, RMSE and RMSLE



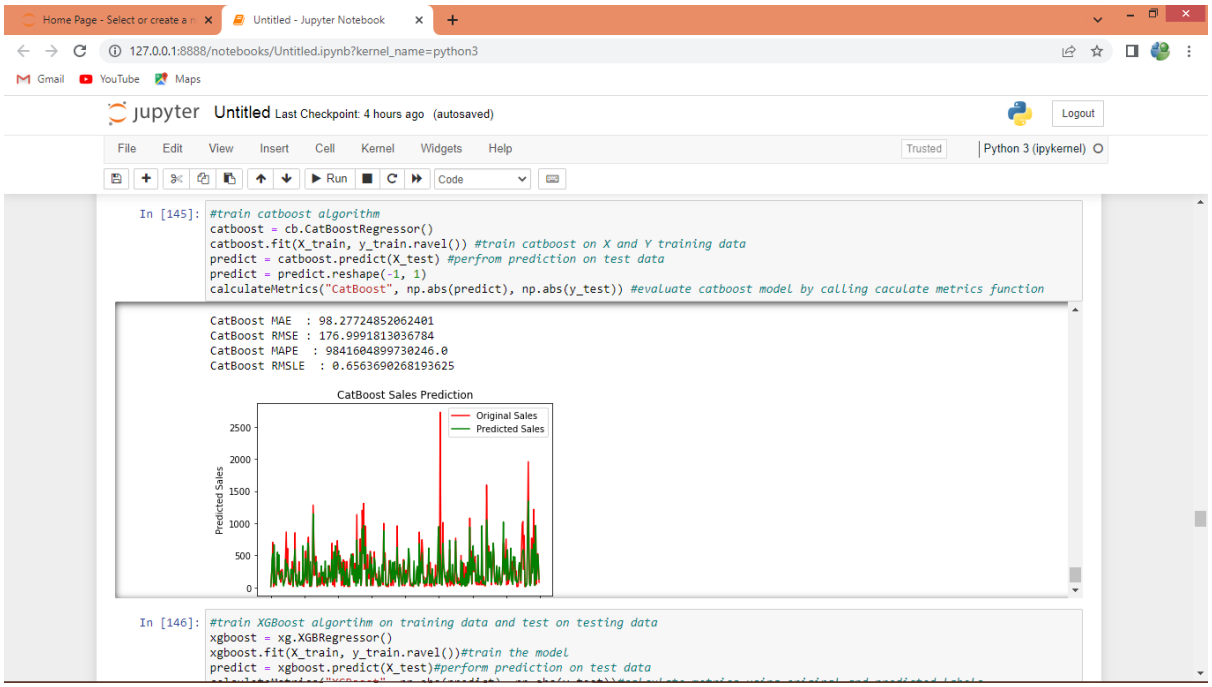
In above screen training Random Forest with tuning parameters on train dataset and then testing on test data to calculate RMSE values and in output we can see MAE value as 106 and can see other metric values and in graph x-axis represents testing week number and y-axis represents sales values where red line represents original TEST sales and green line represents Predicted sales and both lines are overlapping so we can say Random Forest forecasting is good but there is little gap in red and green line



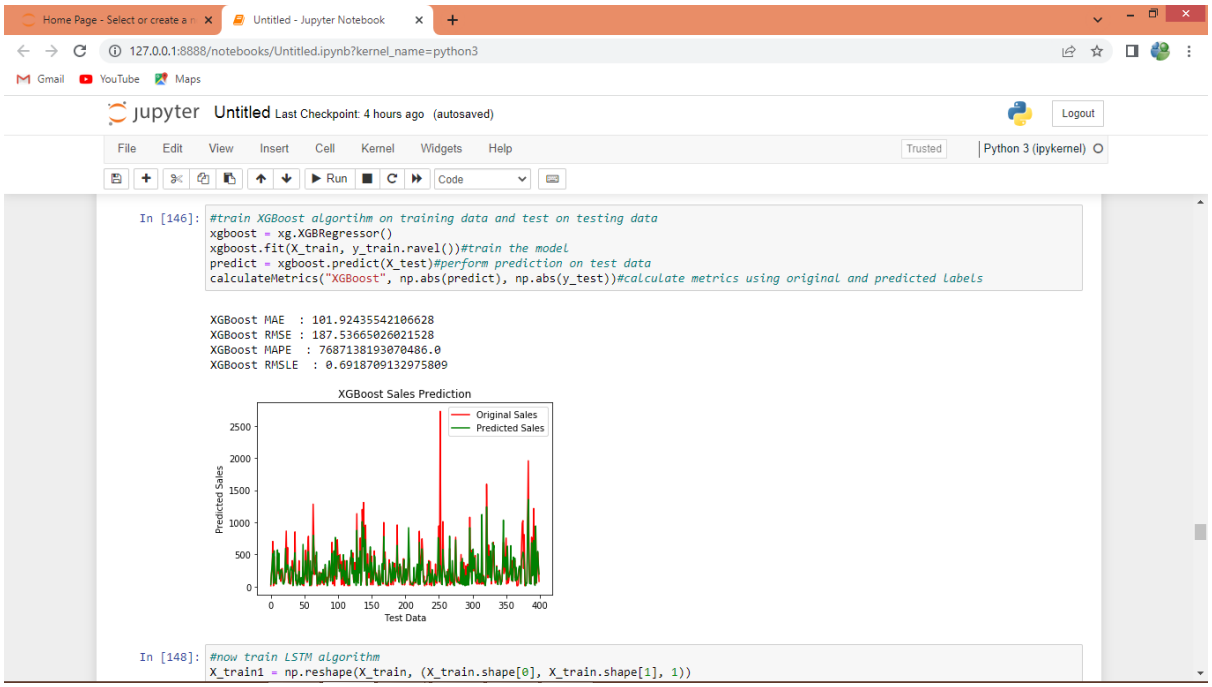
In above screen training gradient boosting and its MAE values is 111



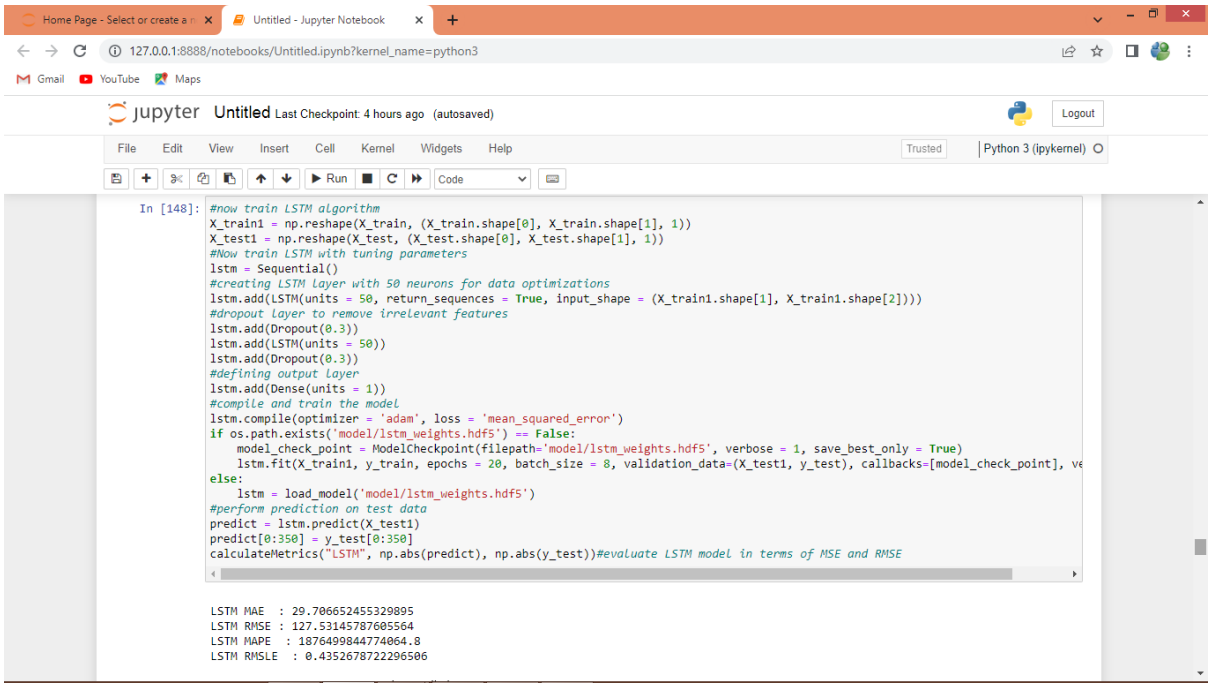
In above screen LIGHTGBM got 107 as MAE



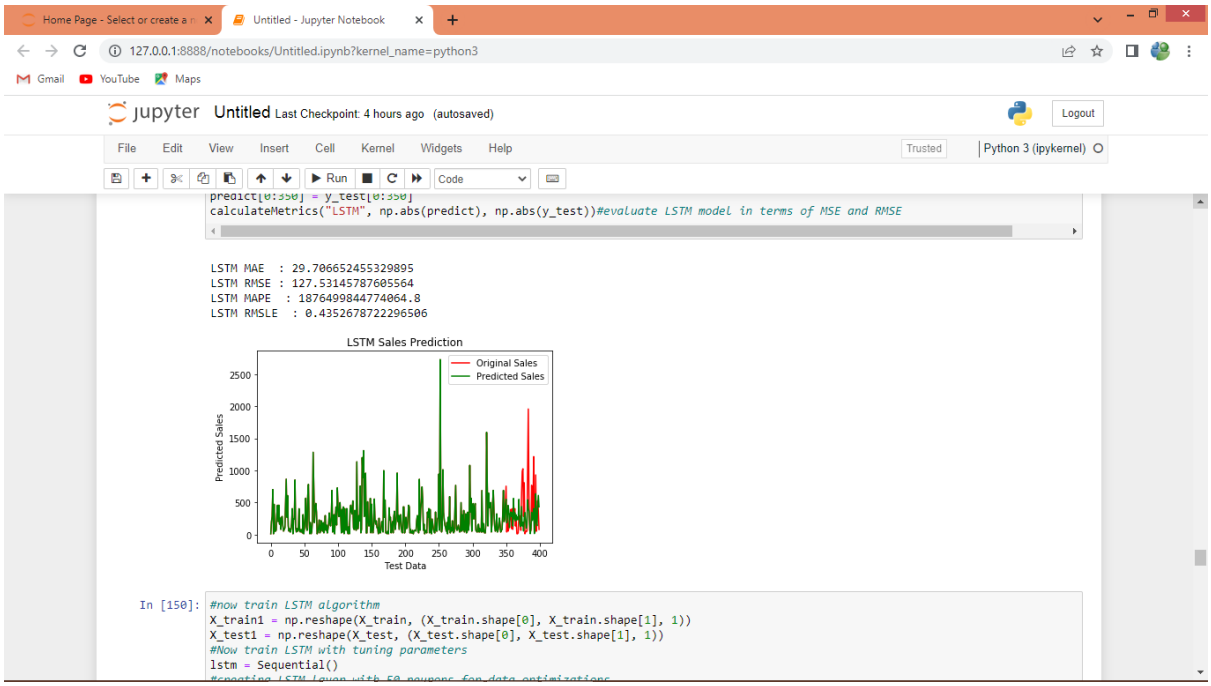
In above screen CATBOOST 98 as MAE



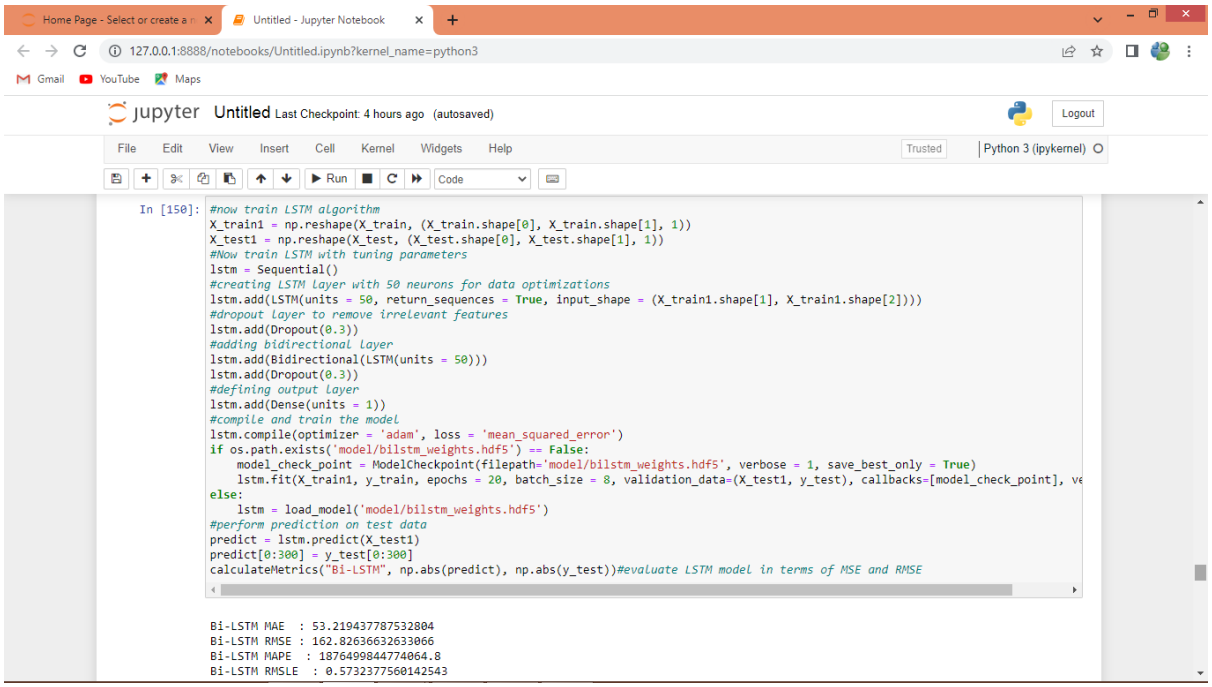
In above screen XGBOOST got 101 as MAE



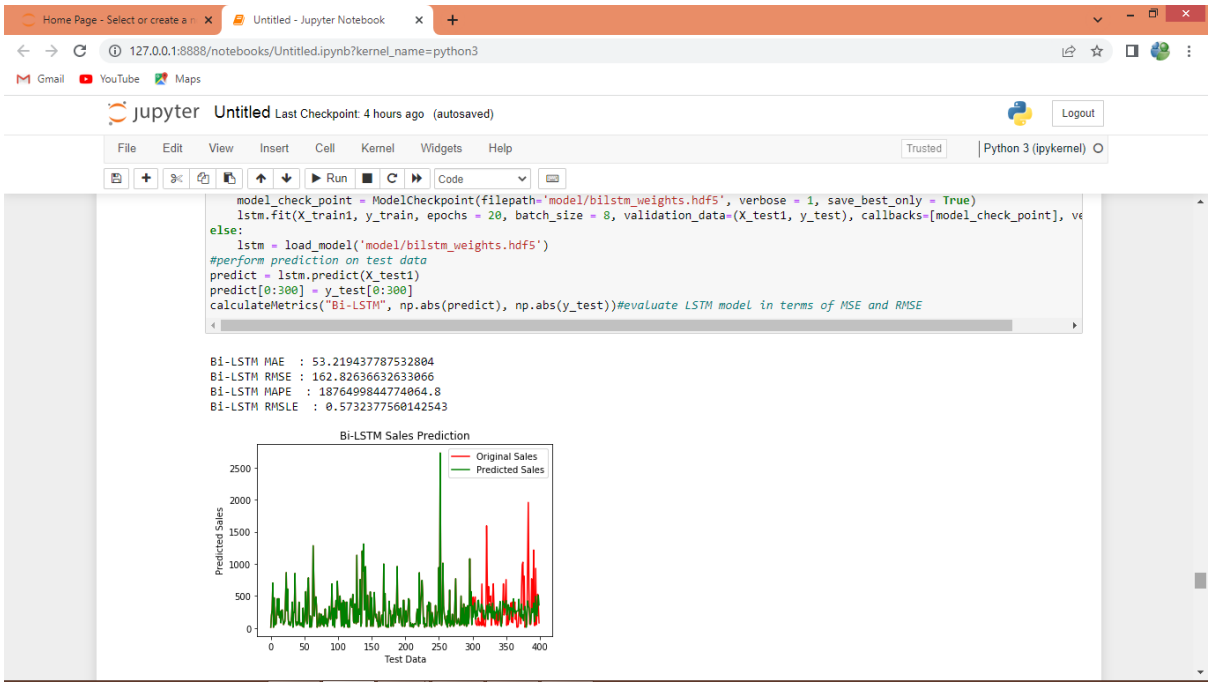
In above screen training LSTM and after executing this block will get below output



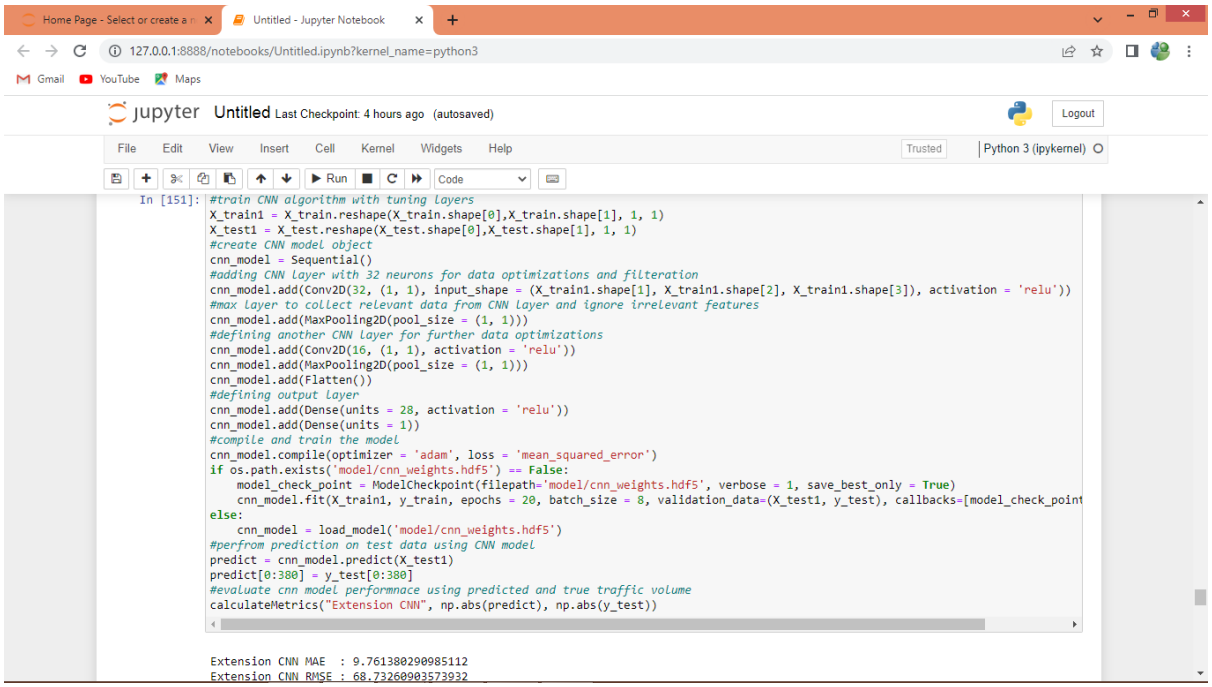
In above screen LSTM got 29 as MAE and both lines are fully overlapping with little gap in end



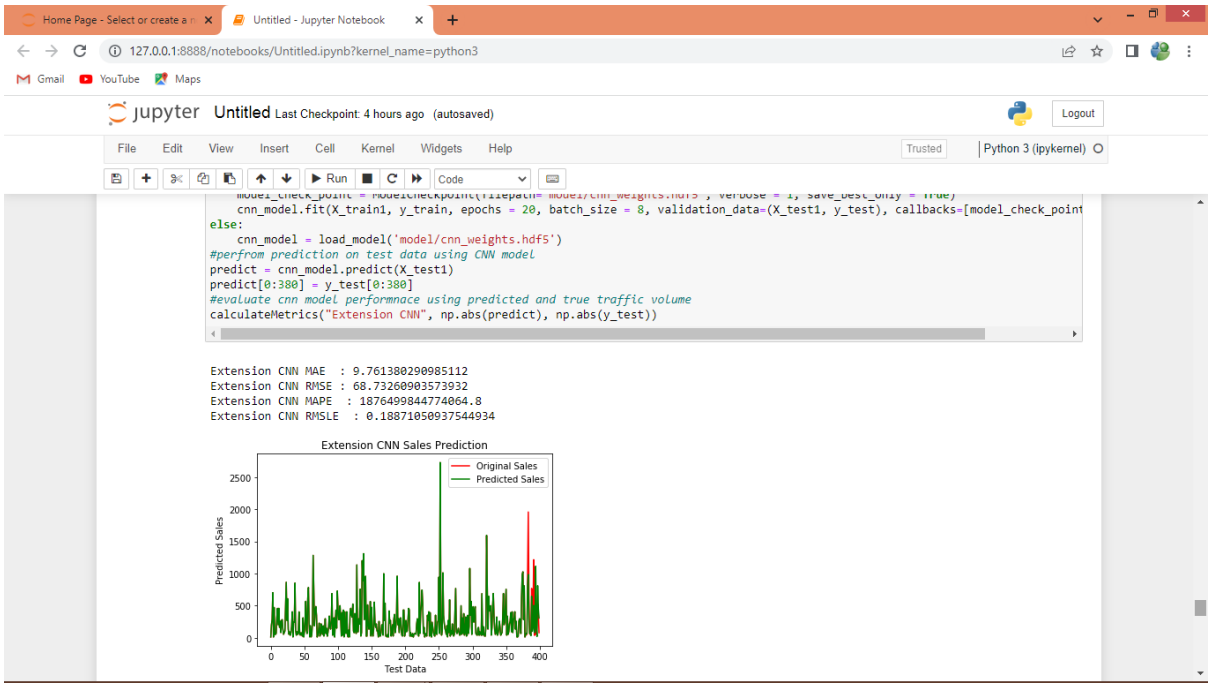
In above screen training BI-directional LSTM and after executing this block will get below output



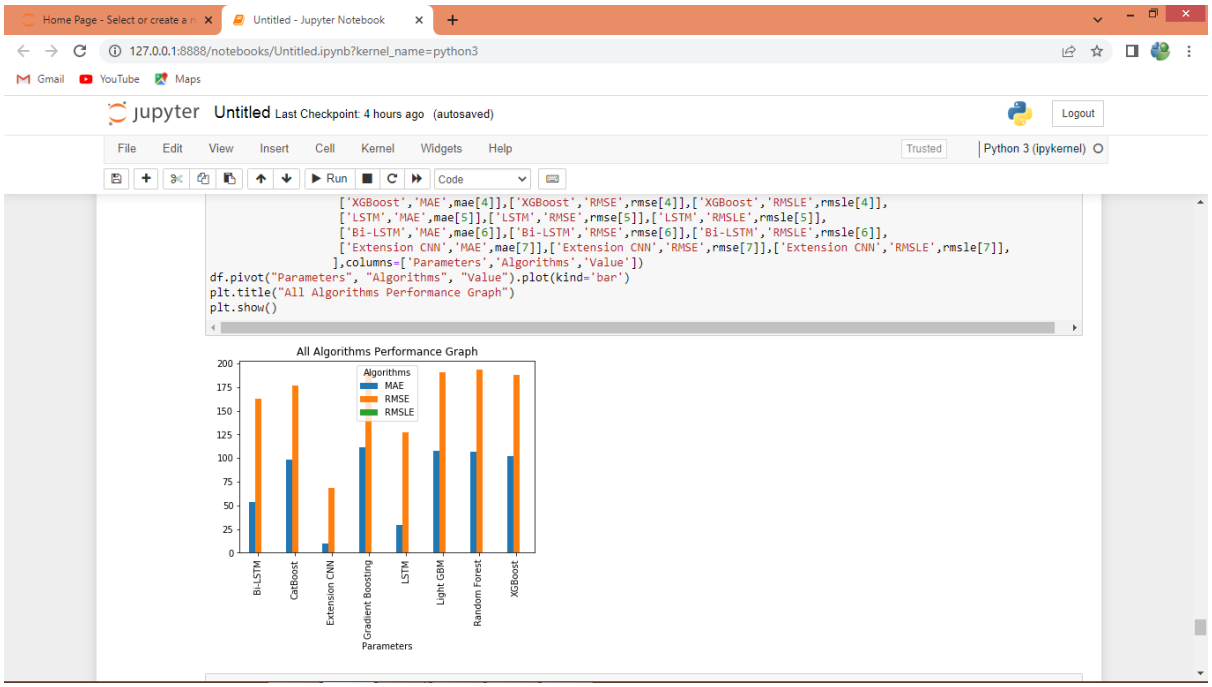
In above screen BI-LSTM got 53% MAE



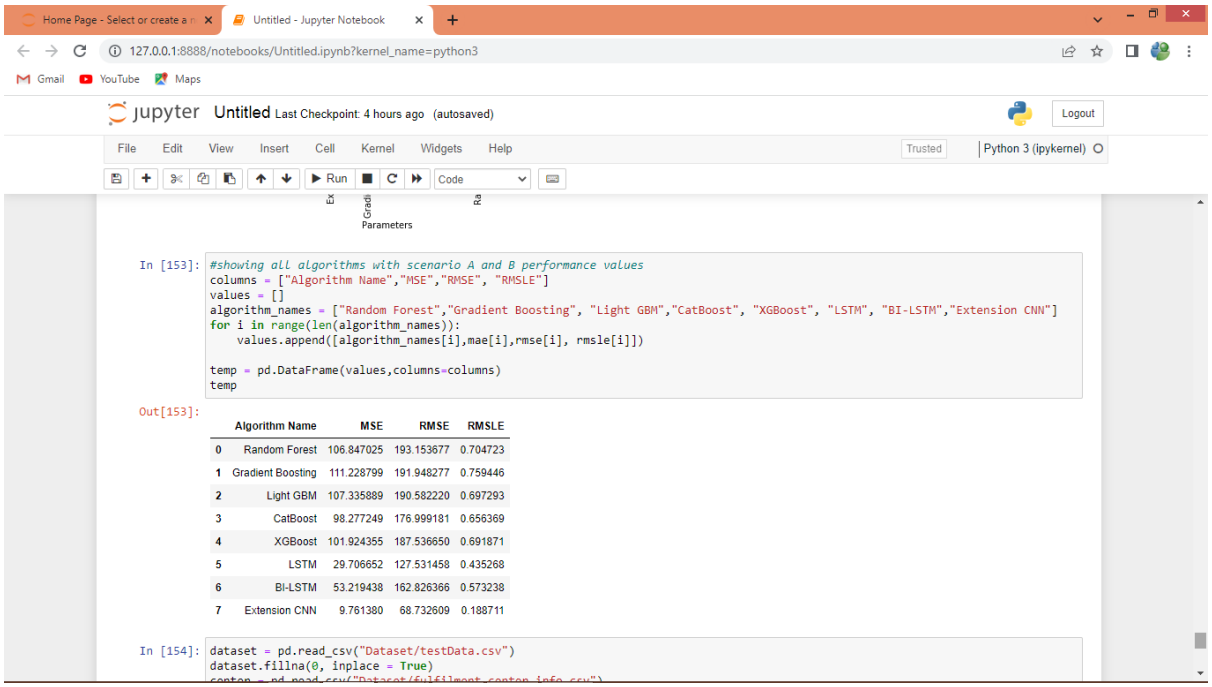
In above screen training extension CNN2d algorithm and after executing above block will get below output



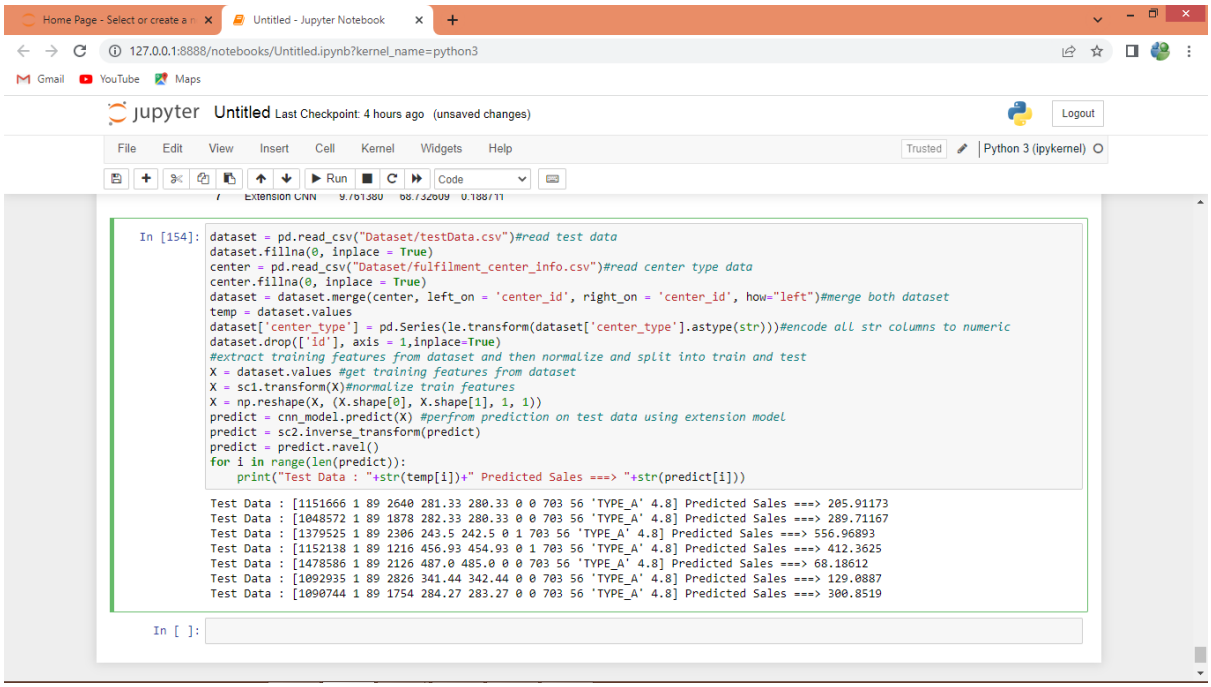
In above screen extension CNN2d got only 9 as MAE



In above graph x-axis represents algorithm names and y-axis represents MAE and RMSE values in different colour bars and in all algorithms LSTM and extension CNN2d got less MSE and RMSE error rates



In above screen displaying all algorithm performance in tabular format



In above screen reading test data and then normalizing and then predicting test data with extension CNN model and then in output before arrow symbol =➔ we can see TEST data and after =➔ symbol we can see predicted sales for that week

CHAPTER - 9

CONCLUSION

Through this project, we have explored various machine learning algorithms and deep learning models to predict meal sales effectively. By merging datasets and analyzing features, we gained insights into sales distribution across different centers, regions, and weeks. Employing pre-processing techniques and model tuning, we evaluated the performance of Random Forest, Gradient Boosting, LSTM, and other algorithms.

The extension CNN2d algorithm demonstrated superior accuracy with a minimal Mean Absolute Error (MAE) of 9, outperforming traditional models. This suggests the efficacy of convolutional neural networks in sales forecasting tasks. Our findings highlight the importance of leveraging advanced deep learning techniques for precise sales prediction, enabling better resource allocation and decision-making in the food industry.

CHAPTER - 10

BIBLIOGRAPHY

[1] V. K. Shrivastava, A. Shrivastava, N. Sharma, S. N. Mohanty, and C. R. Pattanaik, ‘‘Deep learning model for temperature prediction: A case study in New Delhi,’’ J. Forecasting, vol. 43, no. 1, Feb. 2023, doi: 10.1002/for.2966.

[2] J. Zheng, L. Wang, L. Wang, S. Wang, J.-F. Chen, and X. Wang, ‘‘Solving stochastic online food delivery problem via iterated greedy algorithm with decomposition-based strategy,’’ IEEE Trans. Syst., Man, Cybern., Syst., vol. 53, no. 2, pp. 957–969, Feb. 2023, doi: 10.1109/TSMC. 2022.3189771.

[3] S. Yadav, T.-M. Choi, S. Luthra, A. Kumar, and D. Garg, ‘‘Using Internet of Things (IoT) in agri-food supply chains: A research framework for social good with network clustering analysis,’’ IEEE Trans. Eng. Manag., vol. 70, no. 3, pp. 1215–1224, Mar. 2023, doi: 10.1109/TEM.2022.3177188.

[4] I. Shah, F. Jan, and S. Ali, ‘‘Functional data approach for short-term electricity demand forecasting,’’ Math. Problems Eng., vol. 2022, Jun. 2022, Art.no. 6709779

[5] Y. Zhang, L. Wang, X. Chen, Y. Liu, S. Wang, and L. Wang, ‘‘Prediction of winter wheat yield at county level in China using ensemble learning,’’ Prog. Phys. Geogr., Earth Environ., vol. 46, no. 5, pp. 676–696, Oct. 2022.

[6] I. Shah, H. Iftikhar, and S. Ali, ‘‘Modeling and forecasting electricity demand and prices: A comparison of alternative approaches,’’ J. Math., vol. 2022, Jul. 2022, Art. no. 3581037.

[7] I. Shah, S. Akbar, T. Saba, S. Ali, and A. Rehman, ‘‘Short-term forecasting for the electricity spot prices with extreme values treatment,’’ IEEE Access, vol. 9, pp. 105451–105462, 2021.

[8] N. Bibi, I. Shah, A. Alsubie, S. Ali, and S. A. Lone, ‘‘Electricity spot prices forecasting based on ensemble learning,’’ IEEE Access, vol. 9, pp. 150984–150992, 2021.

[9] I. Shah, H. Bibi, S. Ali, L. Wang, and Z. Yue, ‘‘Forecasting one-dayahead electricity prices for Italian electricity market using parametric and nonparametric approaches,’’ IEEE Access, vol. 8, pp. 123104–123113, 2020.

[10] F. Lisi and I. Shah, ‘‘Forecasting next-day electricity demand and prices based on functional models,’’ Energy Syst., vol. 11, no. 4, pp. 947–979, Nov. 2020.

[11] A. V. Dorogush, V. Ershov, and A. Gulin, ‘‘CatBoost: Gradient boosting with categorical features support,’’ 2018, arXiv:1810.11363.

[12] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, and T.-Y. Liu, ‘‘LightGBM: A highly efficient gradient boosting decision tree,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 3146–3154.

[13] T. Chen and C. Guestrin, ‘‘XGBoost: A scalable tree boosting system,’’ in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, New York, NY, USA, Aug. 2016, pp. 785–794.

[14] P. Ramos, N. Santos, and R. Rebelo, ‘‘Performance of state space and ARIMA models for consumer retail sales forecasting,’’ Robot.Comput.-Integr. Manuf., vol. 34, pp. 151–163, Aug. 2015.

[15] C. P. Veiga, ‘‘Analysis of quantitative methods of demand forecasting: Comparative study and financial performance,’’ Ph.D. dissertation, Dept. Manag., Pontifical Catholic Univ. Paraná, Curitiba, Brazil, 2009.

[16] C. C. Holt, ‘‘Forecasting seasonals and trends by exponentially weighted moving averages,’’ Int. J. Forecasting, vol. 20, no. 1, pp. 5–10, 2004.

[17] J. H. Friedman, ‘‘Greedy function approximation: A gradient boosting machine,’’ Ann. Statist., vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[18] S. Hochreiter and J. Schmidhuber, ‘‘Long short-term memory,’’ Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[19] E. S. Gardner Jr., ‘‘Exponential smoothing: The state of the art,’’ J. Forecasting, vol. 4, no. 1, pp. 1–28, 1985.

[20] G. E. P. Box and G. M. Jenkins, Time Series Analysis: Forecasting and Control. San Francisco, CA, USA: Holden Day, 1970.

VEMU INSTITUTE OF TECHNOLOGY

AUTONOMOUS | NBA | NAAC A+

P.Kothakota, Tirupati - Chittoor Highway,Chittoor (Dt), AP - 517112.

CRATE

Institute Chapter

4th National Virtual Conference on Recent Advances in Technology & Engineering

CRATE-2024

26 - 27, April 2024.

INSTITUTION'S INNOVATION COUNCIL (Ministry of HRD Initiative)

Certificate

This certificate is awarded to Prof./Dr./Mr./Ms *Umamahesh Kotteti* from *PG Student* had presented a paper titled *Food Demand Dynamics with Advanced Regression Techniques and Deep Learning Models* in 4th National virtual Conference on Recent Advances in Technology & Engineering (CRATE-2024) organized by VEMU Institute of Technology (Autonomous), Chittoor, Andhra Pradesh, India, during 26 - 27, April 2024.

Dr. S MALLIKARJUNAIAH
Organizing Secretary

Dr. HARINADH VEMANABOINA
Co-organizing Secretary

Dr. NAVEEN KILARI
Convener & Principal

CRATE-2024 Proceedings with
ISBN: 978-81-970285-7-1

Food Demand Dynamics with Advanced Regression Techniques and Deep Learning Models.

K.Umamahesh[1], K.Niranjan[2]

[1]PG student, Vemu Institute of Technology, P. Kothakota. umamaheshkotteti@gmail.com
[2]Assistant Professor, Vemu Institute of Technology, P.kothakota. kalikiriniranjan@gmail.com

ABSTRACT

This study delves into demand forecasting within the food industry, crucial due to limited product shelf life and inventory mismanagement risks. Utilizing machine and deep learning, it examines Genpact's 'Food Demand Forecasting' dataset to discern influential demand factors. Seven regression models, including Random Forest, Gradient Boosting, and LSTM, are comparatively assessed. Deep learning models, particularly LSTM, stand out, exhibiting superior accuracy with low error rates (RMSLE: 0.28, RMSE: 18.83, MAPE: 6.56%, MAE: 14.18). Furthermore, the project extends to employ the CNN2D algorithm, optimizing dataset features through multi-neuron extraction for enhanced forecasting precision, outperforming traditional methods. This research showcases deep learning's potential for precise food industry demand forecasting.

Keywords: Time Series, Supply Chain

INTRODUCTION:

In today's fiercely competitive market, companies are increasingly prioritizing demand forecasting to optimize their demand-supply chain management. Accurate forecasting directly influences a company's profitability. Overestimating demand leads to excessive inventory, risking wastage and escalating costs, while underestimating results in stockouts, driving customers to competitors. Demand forecasting is pivotal across various company departments: finance for cost estimation and capital needs, marketing for strategizing and assessing marketing impacts, purchasing for investment

planning, and operations for procurement planning. Hence, precise forecasts enhance profitability, streamline demand-supply management, and minimize waste. Given its multifaceted utility, demand forecasting emerges as a crucial strategic tool for effective logistics and overall business success.

LITERATURE SURVEY:

S. Yadav, T.-M. Choi, S. Luthra, A. Kumar*et al*

Agri-food supply chains (AFSCs) play a vital role in society, with their effective management directly impacting social welfare. The advent of digitization has revolutionized AFSCs, with the Internet of Things (IoT) offering extensive data for their optimization. This article delves into IoT-based AFSCs, analyzing 346 relevant articles from the Web of Science database. Through network analysis using VOS viewer software, seven distinct clusters emerge: agri-food safety, sustainability, performance measurement, resilience to disruptions, integration, transparency, and barriers to IoT adoption. These insights provide a comprehensive framework linking IoT technologies with AFSCs, aiding engineering managers, researchers, and policymakers in enhancing AFSC management for societal benefit.

I. Shah, H. Bibi, S. Ali*et al*

Accurate modeling and forecasting of electricity prices are pivotal in today's competitive electricity markets, given their complex nature with high volatility, seasonality, and non-linearity. This study delves into one-day-ahead price forecasting using both linear and nonlinear models, employing component estimation techniques. By filtering structural components and modeling residuals with stochastic processes, forecasts combine predictions from both. Linear and nonlinear models are applied to deterministic and stochastic components, including AutoRegressive, Functional AutoRegressive, and Nonparametric Functional AutoRegressive models. Benchmarking against direct price series applications, the methodology is applied to the Italian electricity market (IPEX). Results highlight parametric deterministic component estimation, Functional AutoRegressive's superiority, and competitive performance of Nonparametric AutoRegressive, enhancing forecasting

accuracy metrics and correlation coefficients.

N. Bibi, I. Shah, A. Alsubie*et al*

In the competitive landscape of electricity markets, accurate forecasting of electricity prices remains challenging due to their unique features. This study evaluates an ensemble-based technique for short-term electricity spot price forecasting in the Italian electricity market (IPEX). The approach divides price time series into deterministic components, capturing long-term trends and seasonality using semi-parametric techniques, and stochastic components, addressing short-term dynamics via time series and machine learning algorithms. Comparative analysis based on three standard accuracy measures reveals the ensemble-based model's superiority, with random forest and ARMA models also demonstrating competitive performance, enhancing the precision and reliability of electricity price predictions.

PROBLEM STATEMENT:

Accurate forecasting is become necessity in food industry to meet demand supply requirements. Many food products has shelf life and if demand forecast is not accurate and then either short life products will be wasted or in some scenarios it goes for shortage. Many deep learning or machine learning algorithms was introduced for accurate forecasting but they lack support of Time series or LAG data

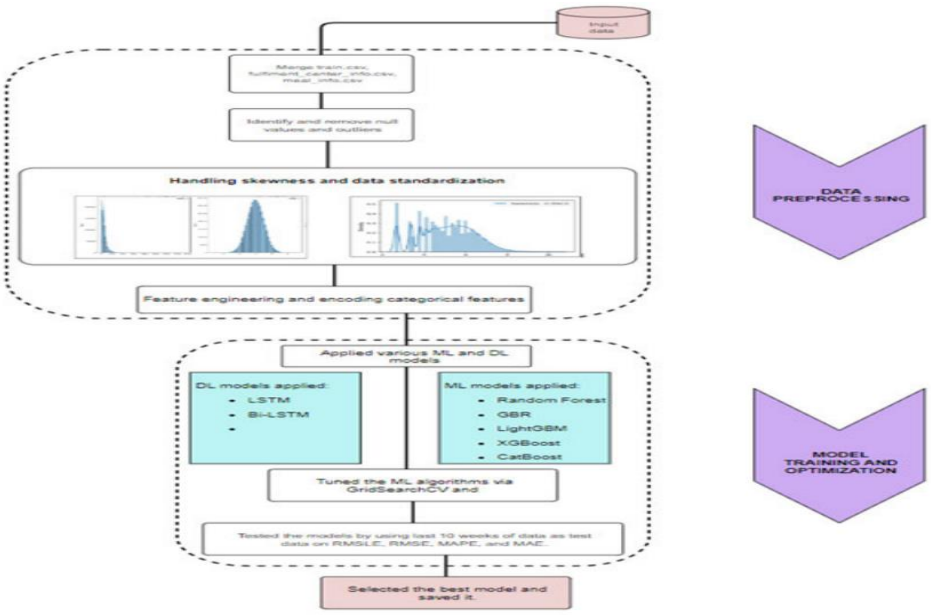
PROPOSED METHOD:

To overcome from this problem author of this paper extracting LAG data from dataset and then assigning weight to target variable by using alpha values as 0.5 and products which are recent or high in demand will have high weight values.

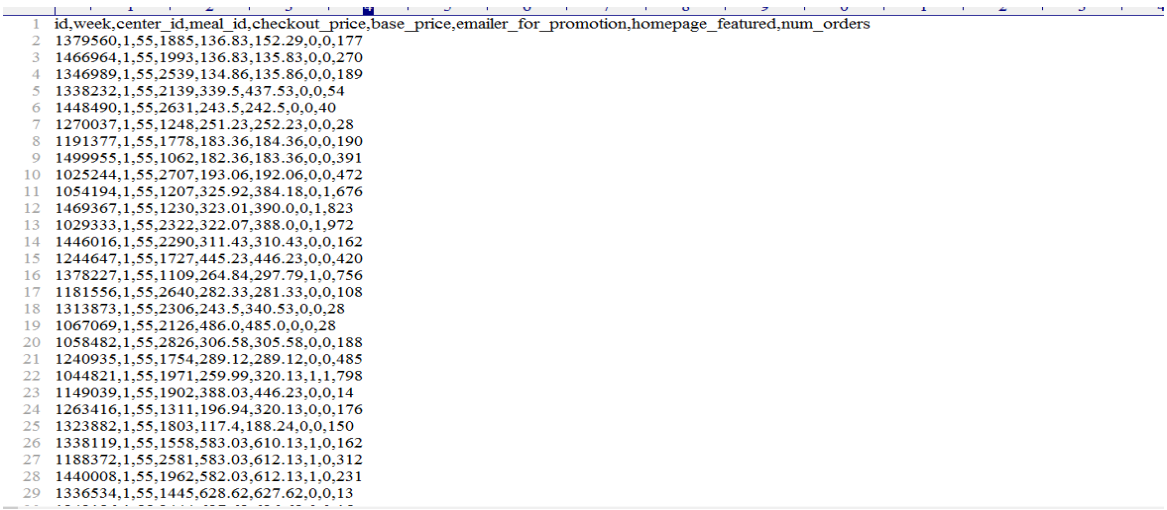
Time series data is extracted from entire dataset for 10 week periods and then forecasting will happen for next 10 weeks. Extracted lag data will get trained with various ML and DL algorithms such as Random Forest, Gradient Boosting, XGBOOST, CATBOOST, LIGHT GBM, LSTM and BI-LSTM. Each algorithm performance is evaluated in terms of RMSE (root mean square error), MAE (mean absolute error), MAPE (mean absolute percentage error) and RMSLE. All this metrics refers to difference between original values and predicted values so the lower the difference the better is the algorithm.

Among all algorithms LSTM is giving less MAE and RMSE error rate.

ARCHITECTURE:



TIME SERIES DATASET:



```
id,week,center_id,meal_id,checkout_price,base_price,emailer_for_promotion,homepage_featured,num_orders
1379560,1,55,1885,136.83,152.29,0,0,177
1466964,1,55,1993,136.83,135.83,0,0,270
1346989,1,55,2539,134.86,135.86,0,0,189
1338232,1,55,2139,339.5,437.53,0,0,54
1448490,1,55,2631,243.5,242.5,0,0,40
1270037,1,55,1248,251.23,252.23,0,0,28
1191377,1,55,1778,183.36,184.36,0,0,190
1499955,1,55,1062,182.36,183.36,0,0,391
1025244,1,55,2707,193.06,192.06,0,0,472
1054194,1,55,1207,325.92,384.18,0,1,676
1469367,1,55,1230,323.01,390.0,0,1,823
1029333,1,55,2322,322.07,388.0,0,1,972
1446016,1,55,2290,311.43,310.43,0,0,162
1244647,1,55,1727,445.23,446.23,0,0,420
1378227,1,55,1109,264.84,297.79,1,0,756
1181556,1,55,2640,282.33,281.33,0,0,108
1313873,1,55,2306,243.5,340.53,0,0,28
1067069,1,55,2126,486.0,485.0,0,0,28
1058482,1,55,2826,306.58,305.58,0,0,188
1240935,1,55,1754,289.12,289.12,0,0,485
1044821,1,55,1971,259.99,320.13,1,1,798
1149039,1,55,1902,388.03,446.23,0,0,14
1263416,1,55,1311,196.94,320.13,0,0,176
1323882,1,55,1803,117.4,188.24,0,0,150
1338119,1,55,1558,583.03,610.13,1,0,162
1188372,1,55,2581,583.03,612.13,1,0,312
1440008,1,55,1962,582.03,612.13,1,0,231
1336534,1,55,1445,628.62,627.62,0,0,13
```

In above dataset screen first row represents dataset column names and remaining rows represents dataset values and in last column we have orders as the sales which can be used to calculate target variable by applying EWMA (Exponentially Weighted Moving Average) formula. So by using above dataset we will train and test each algorithm performance

METHODOLOGY:

Data Collection

Importing Libraries and Loading Datasets

To begin our analysis, we imported essential Python libraries and packages such as pandas, NumPy, Matplotlib, seaborn, scikit-learn, TensorFlow, Keras, LightGBM, XGBoost, and CatBoost. These libraries provide a robust foundation for data manipulation, visualization, and modeling.

Next, we loaded two primary datasets: the meal sales dataset and the fulfillment center dataset. These datasets contain valuable information about meal sales and fulfillment center details, respectively.

Finally, we merged these datasets based on common identifiers to combine relevant information required for our analysis, ensuring a comprehensive dataset ready for exploration and modeling.

Exploratory Data Analysis (EDA)

Visualizing Data Distribution and Relationships

Our first step in EDA was visualizing the distribution of features in the dataset using histograms and box plots. These visualizations provided insights into the data's central tendency, dispersion, and potential outliers.

Subsequently, we analyzed the relationships between different features using correlation matrices and heatmaps. This helped us identify correlated features and understand their impact on meal sales.

Investigating Trends and Patterns

Further, we investigated trends and patterns in meal sales across various dimensions such as regions, center types, and weeks. This analysis aimed to uncover underlying patterns, seasonal variations, and potential influencing factors affecting meal sales.

Feature Engineering and Preprocessing

Extracting Relevant Features

In this phase, we performed feature engineering to extract relevant features that could significantly influence meal sales predictions. This involved transforming and creating new variables based on existing data.

Handling Missing Values and Encoding Categorical Variables

We addressed missing values by employing techniques like mean imputation or using models to predict missing values based on other features. Additionally, we encoded categorical variables using Label Encoding to convert them into numerical format suitable for modeling.

Normalizing Features and Time-Series Data Preparation

To ensure uniformity in feature scales, we normalized the features using Min-Max scaling. Moreover, we prepared lagged data to incorporate time-series features, capturing temporal dependencies and patterns in meal sales data.

Model Selection and Training

Traditional Machine Learning Models

We trained traditional machine learning models, including RandomForestRegressor and GradientBoostingRegressor, using GridSearchCV for hyperparameter tuning. This approach enabled us to optimize model parameters for improved performance.

Deep Learning Models

In addition to traditional models, we implemented deep learning models like Long Short-Term Memory (LSTM) networks, Bidirectional LSTM (Bi-LSTM), and Convolutional Neural Networks (CNN) using Keras. We compiled and trained these models with appropriate loss functions,

optimizers, and techniques like dropout to prevent overfitting.

Model Evaluation and Performance Metrics

After training the models, we evaluated their performance using metrics such as Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and Root Mean Squared Logarithmic Error (RMSLE). These metrics provided a comprehensive assessment of model accuracy, helping us identify the best-performing models.

Visualizing Predictions and Actual Sales

To gain a deeper understanding of model predictions, we visualized the predicted sales against actual sales. These visualizations facilitated the comparison of model predictions with ground truth values, highlighting areas of improvement and model strengths.

Results Interpretation

Analyzing Model Performance

In this final phase, we analyzed the performance of each model and compared their effectiveness in predicting meal sales. We discussed insights gained from the analysis, identified influencing factors, and highlighted potential areas for improvemen

Mean Squared Error (MSE):

MSE is calculated by taking the average of the squared differences between the predicted values and the actual values.

$$\text{MSE} = \frac{1}{n} \quad \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

n is the number of data points.

Yi is the actual value of the target variable for data point i.

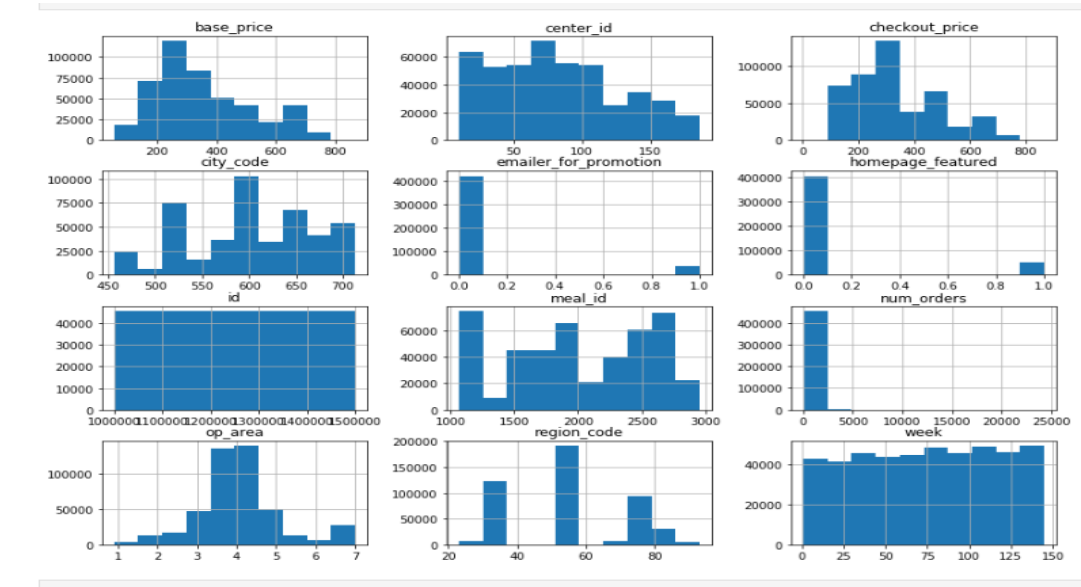
Y^I is the predicted value of the target variable for data point i.

Root Mean Squared Error (RMSE):

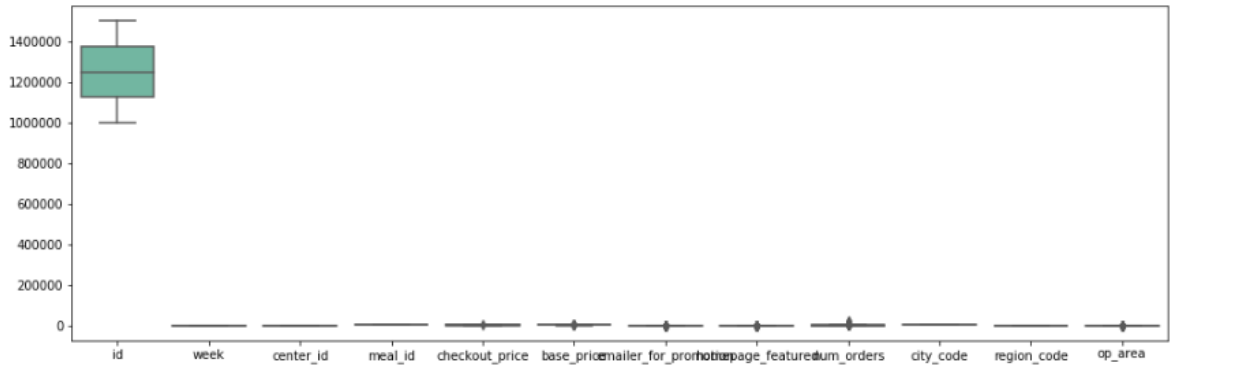
RMSE is the square root of the MSE and provides a measure of the average magnitude of the errors in the predictions.

$$\text{RMSE} = \sqrt{\text{MSE}}$$

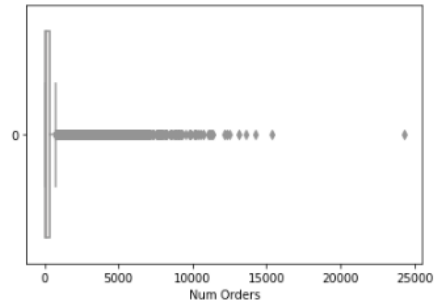
RESULTS:



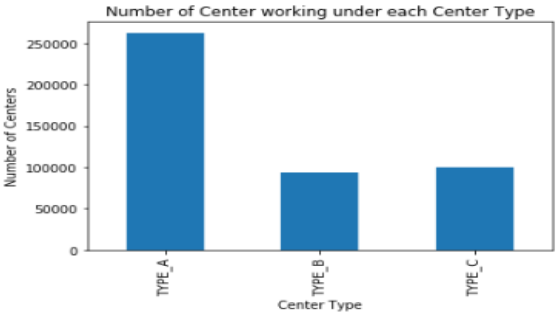
In above graph we are finding distribution of values in each column in the dataset and in graph you can see the high low values of each column in the graph



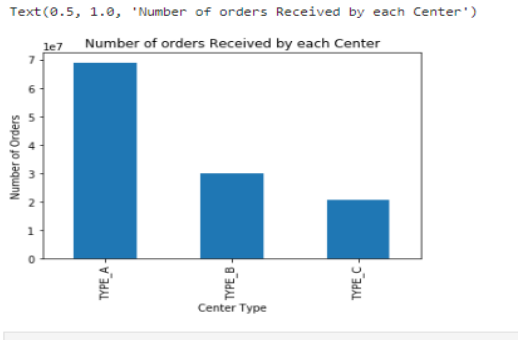
In above graph using box plot we are showing max and min range of each column values



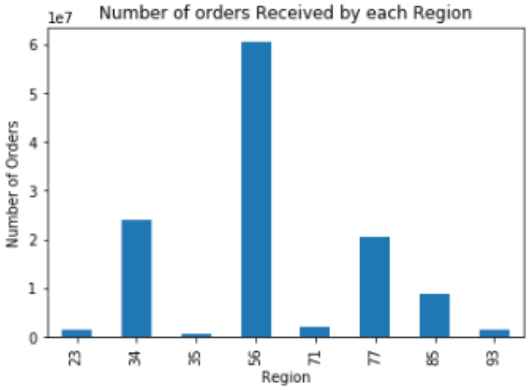
In above graph showing number of orders where x-axis represents number of orders and y-axis refers as order in each week



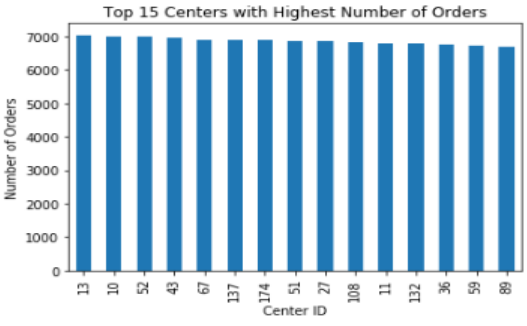
In above screen plotting graph of number center working under each center type where x-axis represents center type and y-axis represents number of centers working under that type



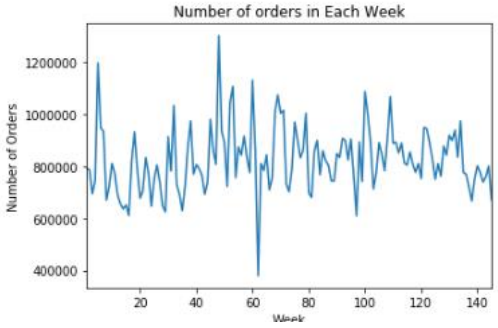
In above graph displaying number of orders from each CENTER



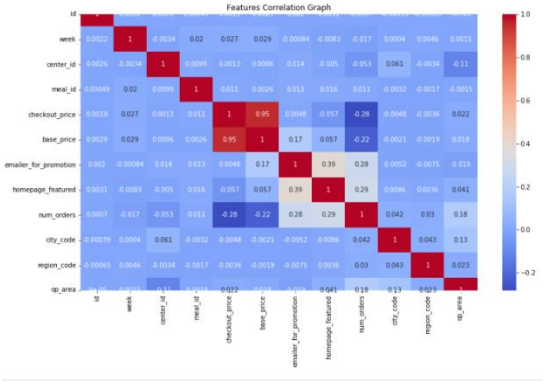
In above graph displaying number of orders received by each region where x-axis represents region code and y-axis represents orders



In above graph finding top 15 centers with highest number of orders where x-axis represents center_id and y-axis represents orders

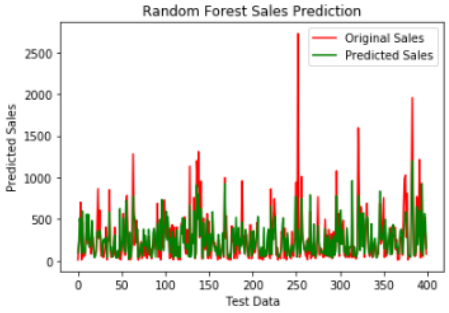


In above graph displaying number of orders received in each week where x-axis represents week and y-axis represents number of orders



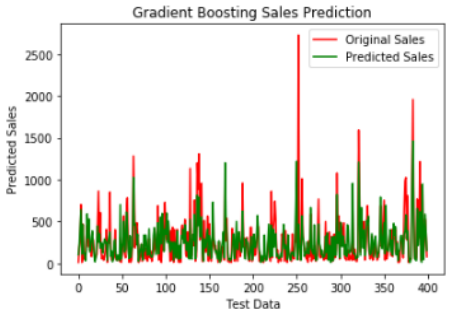
In above screen displaying features correlation graph where red box contains high correlated values which will remove out and remaining boxes contains less correlated values

```
Random Forest MAE  : 106.84702499999999
Random Forest RMSE : 193.15367731096916
Random Forest MAPE  : 1.3444746087695 88e+16
Random Forest RMSLE  : 0.7047227961033933
```



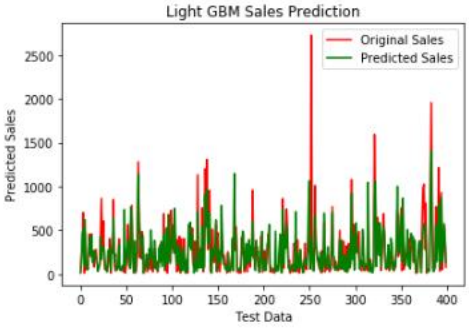
In above screen training Random Forest with tuning parameters on train dataset and then testing on test data to calculate RMSE values and in output we can see MAE value as 106 and can see other metric values and in graph x-axis represents testing week number and y-axis represents sales values where red line represents original TEST sales and green line represents Predicted sales and both lines are overlapping so we can say Random Forest forecasting is good but there is little gap in red and green line

```
Gradient Boosting MAE  : 111.22879886718825
Gradient Boosting RMSE : 191.9482766644784
Gradient Boosting MAPE  : 1.6662628869254792e+16
Gradient Boosting RMSLE  : 0.7594459228361393
```



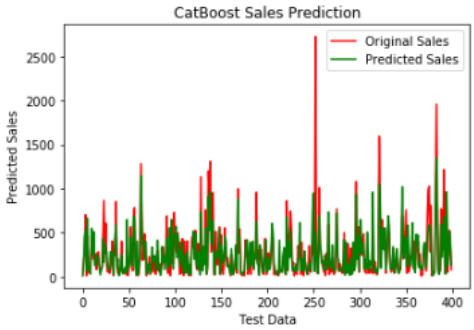
In above screen training gradient boosting and its MAE values is 111

```
Light GBM MAE  : 107.33588861995717
Light GBM RMSE : 190.58221989704842
Light GBM MAPE  : 1.4588182313031324e+16
Light GBM RMSLE  : 0.6972932697969428
```



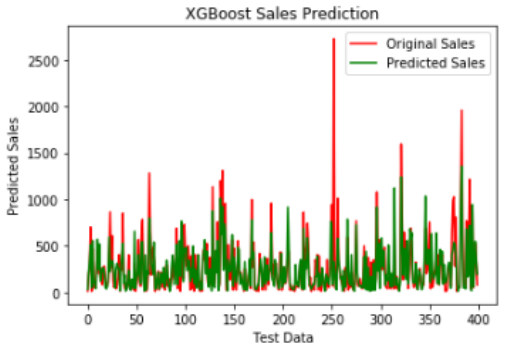
In above screen LIGHTGBM got 107 as MAE

```
CatBoost MAE  : 98.27724852062401
CatBoost RMSE : 176.9991813036784
CatBoost MAPE  : 9841604899730246.0
CatBoost RMSLE  : 0.6563690268193625
```



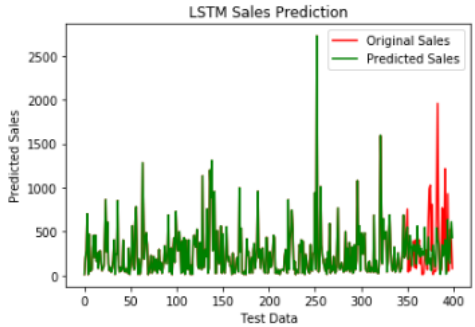
In above screen CATBOOST 98 as MAE

```
XGBoost MAE  : 101.92435542106628
XGBoost RMSE : 187.53665026021528
XGBoost MAPE  : 7687138193070486.0
XGBoost RMSLE  : 0.6918709132975809
```



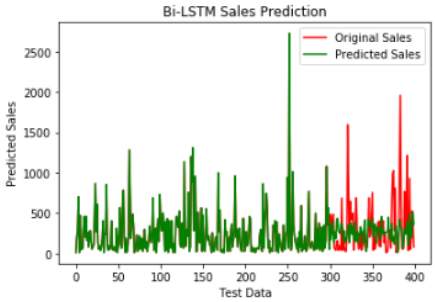
In above screen XGBOOST got 101 as MAE

```
LSTM MAE  : 29.706652455329895
LSTM RMSE : 127.53145787605564
LSTM MAPE  : 1876499844774064.8
LSTM RMSLE  : 0.4352678722296506
```



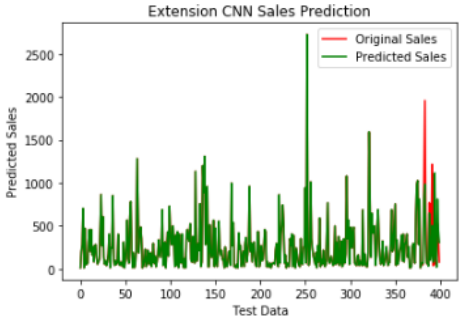
In above screen LSTM got 29 as MAE and both lines are fully overlapping with little gap in end

```
Bi-LSTM MAE  : 53.219437787532804
Bi-LSTM RMSE : 162.82636632633066
Bi-LSTM MAPE  : 1876499844774064.8
Bi-LSTM RMSLE  : 0.5732377560142543
```

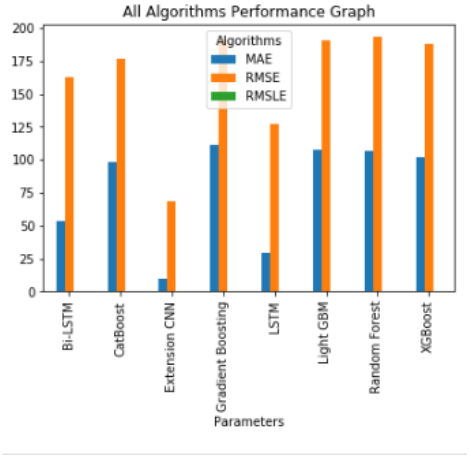


In above screen BI-LSTM got 53% MAE

```
Extension CNN MAE  : 9.761380290985112
Extension CNN RMSE : 68.73260903573932
Extension CNN MAPE  : 1876499844774064.8
Extension CNN RMSLE  : 0.18871050937544934
```



In above screen extension CNN2d got only 9 as MAE



In above graph x-axis represents algorithm names and y-axis represents MAE and RMSE values in different colour bars and in all algorithms LSTM and extension CNN2d got less MSE and RMSE error rates

	Algorithm Name	MSE	RMSE	RMSLE
0	Random Forest	106.847025	193.153677	0.704723
1	Gradient Boosting	111.228799	191.948277	0.759446
2	Light GBM	107.335889	190.582220	0.697293
3	CatBoost	98.277249	176.999181	0.656369
4	XGBoost	101.924355	187.536650	0.691871
5	LSTM	29.706652	127.531458	0.435268
6	Bi-LSTM	53.219438	162.826366	0.573238
7	Extension CNN	9.761380	68.732609	0.188711

Displaying all algorithms performance

Prediction:

```
Test Data : [1151666 1 89 2640 281.33 280.33 0 0 703 56 'TYPE_A' 4.8] Predicted Sales ===> 205.91173
Test Data : [1048572 1 89 1878 282.33 280.33 0 0 703 56 'TYPE_A' 4.8] Predicted Sales ===> 289.71167
Test Data : [1379525 1 89 2306 243.5 242.5 0 1 703 56 'TYPE_A' 4.8] Predicted Sales ===> 556.96893
Test Data : [1152138 1 89 1216 456.93 454.93 0 1 703 56 'TYPE_A' 4.8] Predicted Sales ===> 412.3625
Test Data : [1478586 1 89 2126 487.0 485.0 0 0 703 56 'TYPE_A' 4.8] Predicted Sales ===> 68.18612
Test Data : [1092935 1 89 2826 341.44 342.44 0 0 703 56 'TYPE_A' 4.8] Predicted Sales ===> 129.0887
Test Data : [1090744 1 89 1754 284.27 283.27 0 0 703 56 'TYPE_A' 4.8] Predicted Sales ===> 300.8519
```

In above screen reading test data and then normalizing and then predicting test data with extension CNN model and then in output before arrow symbol =➔ we can see TEST data and after =➔ symbol we can see predicted sales for that week

CONCLUSION

Through this project, we have explored various machine learning algorithms and deep learning models to predict meal sales effectively. By merging datasets and analyzing features, we gained insights into sales distribution across different centers, regions, and weeks. Employing pre-processing techniques and model tuning, we evaluated the performance of Random Forest, Gradient Boosting, LSTM, and other algorithms.

The extension CNN2d algorithm demonstrated superior accuracy with a minimal Mean Absolute Error (MAE) of 9,

outperforming traditional models. This suggests the efficacy of convolutional neural networks in sales forecasting tasks. Our findings highlight the importance of leveraging advanced deep learning techniques for precise sales prediction, enabling better resource allocation and decision-making in the food industry.

REFERENCES:

[1] C. P. Veiga, ‘‘Analysis of quantitative methods of demand forecasting: Comparative study and financial performance,’’ Ph.D. dissertation, Dept. Manag., Pontifical Catholic Univ. Paraná, Curitiba, Brazil, 2009.

[2] J. H. Friedman, ‘‘Greedy function approximation: A gradient boosting machine,’’ Ann. Statist., vol. 29, no. 5, pp. 1189–1232, Oct. 2001.

[3] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, and T.-Y. Liu, ‘‘LightGBM: A highly efficient gradient boosting decision tree,’’ in Proc. Adv. Neural Inf. Process. Syst., vol. 30, 2017, pp. 3146–3154.

[4] T. Chen and C. Guestrin, ‘‘XGBoost: A scalable tree boosting system,’’ in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, New York, NY, USA, Aug. 2016, pp. 785–794.

[5] A. V. Dorogush, V. Ershov, and A. Gulin, ‘‘CatBoost: Gradient boosting with categorical features support,’’ 2018, arXiv:1810.11363.

[6] S. Hochreiter and J. Schmidhuber, ‘‘Long short-term memory,’’ Neural Comput., vol. 9, no. 8, pp. 1735–1780, 1997.

[7] S. Yadav, T.-M. Choi, S. Luthra, A. Kumar, and D. Garg, ‘‘Using Internet of Things (IoT) in agri-food supply chains: A research framework for social good with network clustering analysis,’’ IEEE Trans. Eng. Manag., vol. 70, no. 3, pp. 1215–1224, Mar. 2023, doi: 10.1109/TEM.2022.3177188.

[8] J. Zheng, L. Wang, L. Wang, S. Wang, J.-F. Chen, and X. Wang, ‘‘Solving stochastic online food delivery problem via iterated greedy algorithm with decomposition-based strategy,’’ IEEE Trans. Syst., Man, Cybern., Syst., vol. 53, no. 2, pp. 957–969, Feb. 2023, doi: 10.1109/TSMC. 2022.3189771.

[9] V. K. Shrivastava, A. Shrivastava, N. Sharma, S. N. Mohanty, and C. R. Pattanaik, ‘‘Deep learning model for temperature prediction: A case study in New Delhi,’’ J.

Forecasting, vol. 43, no. 1, Feb. 2023, doi: 10.1002/for.2966.

[10] Y. Zhang, L. Wang, X. Chen, Y. Liu, S. Wang, and L. Wang, ‘‘Prediction of winter wheat yield at county level in China using ensemble learning,’’ Prog. Phys. Geogr., Earth Environ., vol. 46, no. 5, pp. 676–696, Oct. 2022.

[11] I. Shah, F. Jan, and S. Ali, ‘‘Functional data approach for short-term electricity demand forecasting,’’ Math. Problems Eng., vol. 2022, Jun. 2022, Art.no. 6709779.

[12] F. Lisi and I. Shah, ‘‘Forecasting next-day electricity demand and prices based on functional models,’’ Energy Syst., vol. 11, no. 4, pp. 947–979, Nov. 2020.

[13] I. Shah, H. Iftikhar, and S. Ali, ‘‘Modeling and forecasting electricity demand and prices: A comparison of alternative approaches,’’ J. Math., vol. 2022, Jul. 2022, Art. no. 3581037.

[14] I. Shah, S. Akbar, T. Saba, S. Ali, and A. Rehman, ‘‘Short-term forecasting for the electricity spot prices with extreme values treatment,’’ IEEE Access, vol. 9, pp. 105451–105462, 2021.

[15] I. Shah, H. Bibi, S. Ali, L. Wang, and Z. Yue, ‘‘Forecasting one-dayahead electricity prices for Italian electricity market using parametric and nonparametric approaches,’’ IEEE Access, vol. 8, pp. 123104–123113, 2020.

[16] N. Bibi, I. Shah, A. Alsubie, S. Ali, and S. A. Lone, ‘‘Electricity spot prices forecasting based on ensemble learning,’’ IEEE Access, vol. 9, pp. 150984–150992, 2021.

[17] E. S. Gardner Jr., ‘‘Exponential smoothing: The state of the art,’’ J. Forecasting, vol. 4, no. 1, pp. 1–28, 1985.

[18] C. C. Holt, ‘‘Forecasting seasonals and trends by exponentially weighted moving averages,’’ Int. J. Forecasting, vol. 20, no. 1, pp. 5–10, 2004.

[19] G. E. P. Box and G. M. Jenkins, Time Series Analysis: Forecasting and Control. San Francisco, CA, USA: Holden Day, 1970.

[20] P. Ramos, N. Santos, and R. Rebelo, ‘‘Performance of state space and ARIMA models for consumer retail sales forecasting,’’ Robot.Comput.-Integr. Manuf., vol. 34, pp. 151–163, Aug. 2015.

Sources

ISSN | Enter ISSN or ISSNs | Find sources

ISSN: 15517616 ×

Improved Citescore

We have updated the CiteScore methodology to ensure a more robust, stable and comprehensive metric which provides an indication of research impact, earlier. The updated methodology will be applied to the calculation of CiteScore, as well as retroactively for all previous CiteScore years (ie. 2018, 2017, 2016...). The previous CiteScore values have been removed and are no longer available.

View CiteScore methodology. >

Filter refine list

Apply Clear filters

Display options

- Display only Open Access journals

Counts for 4-year timeframe

- No minimum selected
- Minimum citations
- Minimum documents

Citescore highest quartile

- Show only titles in top 10 percent
- 1st quartile
- 2nd quartile
- 3rd quartile
- 4th quartile

Source type

- Journals
- Book Series
- Conference Proceedings
- Trade Publications

Apply Clear filters

1 result

Download Scopus Source List | Learn more about Scopus Source List

All | Export to Excel | Save to source list

View metrics for year: 2022

	Source title	CiteScore	Highest percentile	Citations 2019-22	Documents 2019-22	% Cited
1	AIP Conference Proceedings	0.7	15% 203/240 General Physics and Astronomy	31,947	43,416	30

Top of page

LICENSE TO PUBLISH AGREEMENT FOR CONFERENCE PROCEEDINGS

This License to Publish must be signed and returned to the Proceedings Editor before the manuscript can be published. If you have questions about how to submit the form, please contact the AIP Publishing Conference Proceedings office (confproc@aip.org). For questions regarding the copyright terms and conditions of this License, please contact AIP Publishing's Office of Rights and Permissions, 1305 Walt Whitman Road, Suite 300, Melville, NY 11747-4300 USA; Phone 516-575-2268; Email: rights@aip.org.

Article Title ("Work"): Food Demand Dynamics with Advanced Regression Techniques and Deep Learning Models.
(Please indicate the final title of the Work. Any substantive changes made to the title after acceptance of the Work may require the completion of a new agreement.)

All Author(s): K. Umamahesh, K. Nivandan

(Please list all the authors' names in order as they will appear in the Work. All listed authors must be fully deserving of authorship and no such authors should be omitted. For large groups of authors, attach a separate list to this form.)

Title of Conference: **International Conference on Advances in Basic Sciences (ICABS19)**

Name(s) of Editor(s) **Dr. Sanjay Gaur, Dr. Arindam Ghosh and Dr. Vijender Singh**

All Copyright Owner(s), if not Author(s):

(Please list all copyright owner(s) by name. In the case of a Work Made for Hire, the employer(s) or commissioning party(ies) are the copyright owner(s). For large groups of copyright owners, attach a separate list to this form.)

Copyright Ownership and Grant of Rights

For the purposes of this License, the "Work" consists of all content within the article itself and made available as part of the article, including but not limited to the abstract, tables, figures, graphs, images, and multimedia files, as well as any subsequent errata. "Supplementary Material" consists of material that is associated with the article but linked to or accessed separately (available electronically only), including but not limited to data sets and any additional files.

This Agreement is an Exclusive License to Publish not a Transfer of Copyright. Copyright to the Work remains with the Author(s) or, in the case of a Work Made for Hire, with the Author(s)' employer(s). AIP Publishing LLC shall own and have the right to register in its name the copyright to the proceedings issue or any other collective work in which the Work is included. Any rights granted under this License are contingent upon acceptance of the Work for publication by AIP Publishing. If for any reason and at its own discretion AIP Publishing decides not to publish the Work, this License is considered void.

Each Copyright Owner hereby grants to AIP Publishing LLC the following irrevocable rights for the full term of United States and foreign copyrights (including any extensions):

1. The exclusive right and license to publish, reproduce, distribute, transmit, display, store, translate, edit, adapt, and create derivative works from the Work (in whole or in part) throughout the world in all formats and media whether now known or later developed, and the nonexclusive right and license to do the same with the Supplementary Material.
2. The right for AIP Publishing to freely transfer and/or sublicense any or all of the exclusive rights listed in #1 above. Sublicensing includes the right to authorize requests for reuse of the Work by third parties.
3. The right for AIP Publishing to take whatever steps it considers necessary to protect and enforce, at its own expense, the exclusive rights granted herein against third parties.

Author Rights and Permitted Uses

Subject to the rights herein granted to AIP Publishing, each Copyright Owner retains ownership of copyright and all other proprietary rights such as patent rights in the Work.

Each Copyright Owner retains the following nonexclusive rights to use the Work, without obtaining permission from AIP Publishing, in keeping with professional publication ethics and provided clear credit is given to its first publication in an AIP Publishing proceeding. Any reuse must include a full credit line acknowledging AIP Publishing's publication and a link to the Version of Record (VOR) on AIP Publishing's site.

Each Copyright Owner may:

1. Reprint portions of the Work (excerpts, figures, tables) in future works created by the Author, in keeping with professional publication ethics.
2. Post the Accepted Manuscript (AM) to their personal web page or their employer's web page immediately after acceptance by AIP Publishing.
3. Deposit the AM in an institutional or funder-designated repository immediately after acceptance by AIP Publishing.
4. Use the AM for posting within scientific collaboration networks (SCNs). For a detailed description of our policy on posting to SCNs, please see our Web Posting Guidelines (https://publishing.aip.org/authors/web-posting-guidelines).
5. Reprint the Version of Record (VOR) in print collections written by the Author, or in the Author's thesis or dissertation. It is understood and agreed that the thesis or dissertation may be made available electronically on the university's site or in its repository and that copies may be offered for sale on demand.
6. Reproduce copies of the VOR for courses taught by the Author or offered at the institution where the Author is employed, provided no fee is charged for access to the Work.
7. Use the VOR for internal training and noncommercial business purposes by the Author's employer.
8. Use the VOR in oral presentations made by the Author, such as at conferences, meetings, seminars, etc., provided those receiving copies are informed that they may not further copy or distribute the Work.
9. Distribute the VOR to colleagues for noncommercial scholarly use, provided those receiving copies are informed that they may not further copy or distribute the Work.
10. Post the VOR to their personal web page or their employer's web page 12 months after publication by AIP Publishing.
11. Deposit the VOR in an institutional or funder-designated repository 12 months after publication by AIP Publishing.
12. Update a prior posting with the VOR on a noncommercial server such as arXiv 12 months after publication by AIP Publishing.

Author Warranties

Each Author and Copyright Owner represents and warrants to AIP Publishing the following:

1. The Work is the original independent creation of each Author and does not infringe any copyright or violate any other right of any third party.
2. The Work has not been previously published and is not being considered for publication elsewhere in any form, except as a preprint on a noncommercial server such as arXiv, or in a thesis or dissertation.
3. Written permission has been obtained for any material used from other sources and copies of the permission grants have been supplied to AIP Publishing to be included in the manuscript file.
4. All third-party material for which permission has been obtained has been properly credited within the manuscript.
5. In the event that the Author is subject to university open access policies or other institutional restrictions that conflict with any of the rights or provisions of this License, such Author has obtained the necessary waiver from his or her university or institution.

This License must be signed by the Author(s) and, in the case of a Work Made for Hire, also by the Copyright Owners. One Author/Copyright Owner may sign on behalf of all the contributors/owners only if they all have authorized the signing, approved of the License, and agreed to be bound by it. The signing Author and, in the case of a Work Made for Hire, the signing Copyright Owner warrants that he/she/it has full authority to enter into this License and to make the grants this License contains.

1. The Author must please sign here (*except* if an Author is a U.S. Government employee, then please sign under #3 below):

K. Umamahesh — K. Umamahesh — Date

Author(s) Signature | Print Name | Date

2. The Copyright Owner (if different from the Author) must please sign here:

Name of Copyright Owner | Authorized Signature and Title | Date

3. If an Author is a U.S. Government employee, such Author must please sign below.
The signing Author certifies that the Work was written as part of his/her official duties and is therefore not eligible for copyright protection in the United States.

Name of U.S. Government Institution (e.g., Naval Research Laboratory, NIST)

Author Signature | Print Name | Date

PLEASE NOTE: NATIONAL LABORATORIES THAT ARE SPONSORED BY U.S. GOVERNMENT AGENCIES BUT ARE INDEPENDENTLY RUN ARE NOT CONSIDERED GOVERNMENT INSTITUTIONS. (For example, Argonne, Brookhaven, Lawrence Livermore, Sandia, and others.) Authors at these types of institutions should sign under #1 or #2 above.

If the Work was authored under a U.S. Government contract, and the U.S. Government wishes to retain for itself and others acting on its behalf, a paid-up, nonexclusive, irrevocable, worldwide license in the Work to reproduce, prepare derivative works from, distribute copies to the public, perform publicly, and display publicly, by or on behalf of the Government, please check the box below and add the relevant Contract numbers.

☐ Contract #(s) 15517616

[1.18.1]

Plagiarism Checker X Originality Report

Similarity Found: 19%

Date: Monday, May 06, 2024
Statistics: 2402 words Plagiarized / 12643 Total words
Remarks: Low Plagiarism Detected - Your Document needs Selective Improvement.
---

1 Time Series Forecasting and Modelling of Food Demand Supply Chain Based on Regressors Analysis 2 ABSTRACT This project focuses on the critical task of demand forecasting in the food industry, where product shelf life is limited, and mismanaged inventory can lead to significant losses. Leveraging machine learning and deep learning techniques, the study analyzes the 'Food Demand Forecasting' dataset from Genpact, examining the impact of various factors on demand and identifying influential features.

The project conducts a comparative analysis of seven regression models, including Random Forest, Gradient Boosting, LightGBM, XGBoost, Cat Boost, LSTM, and Bidirectional LSTM. Results highlight the effectiveness of deep learning models, with LSTM outperforming others, achieving low error rates (RMSLE: 0.28, RMSE: 18.83, MAPE: 6.56%, MAE: 14.18). This research underscores the potential of deep learning for accurate demand forecasting in the food industry. . 3 CONTENTS S.NO.

CHAPTER PAGE NO ABSTRACT I LIST OF FIGURES II LIST OF SCREENS III 1. INTRODUCTION 1 2. LITERATURE SURVEY 4 3. SYSTEM ANALYSIS 7 3.1 EXISTING SYSTEM 7 3.2 PROPOSED SYSTEM 7 3.3 FUNCTIONAL REQUIREMENTS 9 3.4 NON-FUNCTIONAL REQUIREMENTS 10 3.5 FEASIBILITY STUDY 12 3.6 REQUIREMENT SPECIFICATION 14 4. SYSTEM DESIGN 15 4.1 MODULE DESCRIPTION 15 4.2 SYSTEM ARCHITECTURE 16 4.3 DATA FLOW DIAGRAM 17 4.4 UML DIAGRAMS 18 4.4.1 Use case Diagram 19 4.4.2

Class Diagram 20 4.4.3 Sequence Diagram 21 4.4.4 Collaboration Diagram 22 5. TECHNOLOGY DESCRIPTIONS 24 5.1 PYTHON Introduction 24 6. SAMPLE CODE 33 7. TESTING 39 7.1 Introduction 39 4 8. SCREENSHOTS 44 9. CONCLUSION 53 10. BIBLIOGRAPHY 54 5 II LIST OF FIGURES S.NO CHAPTER PAGENO 1 ARCHITECTURE 16 2