**ABSTRACT**

Natural gas is the world's most leading fuel. Some machine learning models fits the dataset efficiently depending upon the type of data points provided. The main aim of this project is to find the different models that efficiently fit the data points and predict the price of fuel with the help of a machine learning model. This project aims to compare the different supervised learning models and bring a conclusion based on the efficiency. We have used 2 supervised learning models SVM, RandomForest to know which gives best in terms of accuracy and performance. We have tried these algorithms which are mostly adaptive to many environments. Now-a-days the gas price has been increasing in leaps and bounds due to certain reasons like inflation throughout the world. This has become a major problem in India where prices of LPG, Petroleum, Diesel have been increasing; India gets its source from neighboring countries like Dubai and Saudi-Arabia. To predict the values of the petroleum and Diesel in the mere future, we have decided to use the Machine Learning algorithms and after choosing a set of algorithms, which have given the most accurate results.

Keywords: Prediction, Natural gas Prices, Machine Learning Models.

**INTRODUCTION:**

As we are experiencing an unstable increase in Natural gas prices where the oil prices are dependent on the Natural gas prices of Dubai and Saudi-Arabia. The transportation will be affected for change in prices. The Natural gas price of India has also been taken into consideration for the accurate prediction of prices. I have used various algorithms for predicting the diesel price in India. The algorithms which we have used are Random Forest, SVM. The Prediction of Natural gas rates based on the previous datasets on the data and prices as the feature list are inputs and target list are predicted values. The implementation was on the model which is feasible to some extent for the prediction of the Natural gas prices. The implementation is on predicting the Natural gas prices for the days using Python machine learning Algorithm and getting results based on prediction.

**EXISTING SYSTEM:**

Price forecasting using statistical modeling methods and data mining has been a topic of great interest among data scientists around the world. In this paper, different machine learning approaches are applied to forecasting future yearly price trends in the natural gas Title Transfer Facility market in the Netherlands. The study compares two models: random forest and support vector classifiers. The identification of potential natural gas price drivers that improve the model’s classification is crucial. The forecast horizon was set in a range from 10 to 60 trading days, considering that shorter time horizons have greater importance for trading. The results reflect values up to 85% of the area-under-the-curve score as a reaction of the models to the four different feature combinations used. This invites continued research on the multiple opportunities that these new technologies could create.

**PROPOSED SYSTEM:**

Using the data set of prices provided from the 7th of January 1997 until 11th of august 2020, we will be trying to predict the prices of natural gas by testing through various machine learning models and providing a real-time web based GUI to ask the user to enter the desired date to predict the rate of the natural gas on that particular day. This study builds upon the existing literature by investigating the accuracy of various forecast methods until the best fit horizon is reached.

**Application**

* Natural gas accounts for 1/4 of the global demand and roughly 1/3 of the US energy demand. After oil, Natural gas is the most dominant sort of energy. So, being about to improve natural gas demand prediction is extremely valuable.
* The accurate prediction of energy price is critical to the energy market orientation, and it can provide a reference for policymakers and market participants.
* In practice, energy prices are affected by external factors, and their accurate prediction is challenging.Being able to forecast natural gas prices benefits various stakeholders and has become a very valuable tool for all market participants in competitive natural gas markets. Machine learning algorithms have gradually become popular tools for natural gas price forecasting.

**Literature survey:**

Energy in general is vital for sustainable development of any nation: Be it social, economic or environment. Consumption of clean energies like natural gas is also an important criterion to evaluate the performance of the economy of any nation, for it is a crucial factor of production in all aspects of every economy. Interest in forecasting natural gas has led to a tremendous surge of research activities in the last decade (Soldo, 2012; Aydin, 2014; Voudouris et al., 2014;

Fagiani et al., 2015; Khan, 2015; Suganthi and Samuel, 2012). World energy demand has increased sharply as primary energy sources are required for sustainable development (Fan and Xia, 2012; Panella et al., 2012; Gracias et al., 2012; Aydin, 2015; Azadeh et al., 2015; Ozturk and Al-Mulali, 2015; Szoplik, 2015; Rafindadi and Ozturk, 2015; Zaman, 2016; Karacaer-Ulusoy and Kapusuzoglu, 2017). Energy is linked to industrial production, agricultural output, health, access to water, population, education, quality of life, cooking, transport (Xiong et al., 2014; Dilaver et al., 2014). Around one-fifth of the globe’s primary energy is met from natural gas which is considered as the cleanest-burning fossil fuel (Selehnia et al., 2013; BP, 2009; Imam et al., 2004;

Xu and Wang, 2010; Agbonifo, 2016; Solarin and Ozturk, 2016). Aside from natural gas demand, the question of its availability quickly arises to know whether gas resources are large enough to support projected plans. For this, it is of crucial importance to be able to predict natural gas consumption with an acceptable degree of accuracy in order to improve operational efficiency, save energy, reduce costs at different levels, manage supply contracts, indigenous production and infrastructures planning. Failing one of these goals would certainly cause instabilities in a nation’s energy system (Bianco et al., 2014).

Over the years, energy demand, particularly natural gas demand has attracted interest as much significance is attached to the issue(Erdogdu, 2010; Gorucu, 2004; Gorucu and Gumrah, 2004). Many studies have forecasted natural gas production, consumption or demand, prices and income elasticity, market volatility and hike in prices in several different areas, on world level, regional level, national level, city level, industrial and residential sectors, and individual costumer level (Hubbert, 1949; 1956; Al-Jarri and Startzman, 1997; Sanchez-Ubeda and and Berzosa, 2007; Behrouznia et al., 2010; Tonkovic et al. (2009); Azadeh et al., 2010). These studies used various data which could be classified into three main categories: Meteorological data, historical data and economic data. The first experimental study on natural gas demand prediction was conducted in 1966 by Balestra and Nerlove (1966) while the first theoretical study was that of Hubbert (1949).

In 2015, Izadya et al. (2015) carried out a systematic approach to create the extreme learning machine monthly overall natural gas consumption as a residential demand side of the district heating system predictive model, Gomez et al. (2015) elaborated a detailed model of the Uzbek energy system in other to analyze in a quantitative way the options in Uzbekistan. Khan (2015) developed a sectoral (industrial, transport, residential and commercial sectors) natural gas demand model and estimated short- and long-run income, price and cross price elasticities in Pakistan over the period 2012–2015. Using trend analysis, Aydin (2015) modeled the consumption of energy sources for Turkey and presented future planning. Still in Turkey, Boran (2015) proposed a grey prediction with rolling mechanism approach to predict natural gas consumption and assist policymakers. Azadeh et al. (2015) presented an integrated forecasting algorithm based on ANFIS and computer simulation (SC) for long-term gas forecasting with economic, environmental and IT/IS (number of computer users divided by population in each year) as standard indicators. Szoplik (2015) discussed and presented the results of forecasts of cumulative gas demand for residents of Szczecin; Fagiani et al. (2015) presented experiments concerning the prediction of water and natural gas consumption, focusing on how to exploit data heterogeneity to get a reliable outcome; Askari et al. (2015) predicted the behavior of an abstract-semi-dynamic natural gas distribution network and nodal gas consumption; Nai-ming et al. (2015) applied new models to forecast the developing trends of China’s energy production and consumption under the influence of China’s energy saving policy; Zhang et al. (2015) analyzed the impacts of gas supply costs on interregional gas flow and gas infrastructure development in China out to 2035, Zhu et al. (2015) also presented a novel approach, named support vector regression (SVR) machine based support vector regression local predictor (SVRLP) with false neighbors filtered-support vector regression local predictor (FNF-SVRLP), to predict short-term natural gas demand. Darda et al. (2015) investigated the natural gas production of four South Asian states: Bangladesh, India, Maynmar, and Pakistan. Using a comprehensive framework, Lin and Li (2015) explored the spillover effects between crude oil markets and natural gas markets, including price spillover and volatility spillover. Lebed’ko and Lebed’ko (2015) analyzed the key indicators of the provision of economically recoverable hydrocarbons of oil and gas industry in Russia, and Clarkson et al. (2015) demonstrated that analytical/semi-analytical methods, developed previously for single-phase flow analysis, can be extended to tight gas and shale cases that exhibit significant condensate production.

**2. ANALYSIS**

The analysis is defined as a detailed examination of the elements or structure of something.

**2.1 REQUIREMENT ANALYSIS**

The process to gather the software requirements from clients, analyze and document them is known as requirements engineering or requirements analysis. The goal of requirement engineering is to develop and maintain sophisticated and descriptive ‘System/Software Requirements Specification’ documents. It is a four step process generally, which includes –

• Feasibility Study four-step

• Requirements Gathering

• Software Requirements Specification

• Software Requirements Validation

The basic requirements of our project are:

• Python installed

• Research Papers

• Datasets

• Accuracy

**2.1.1 FUNCTIONAL REQUIREMENT ANALYSIS**

Functional requirements explain what has to be done by identifying the necessary task, action, or activity that must be accomplished. Functional requirements analysis will be used as the top-level functions for functional analysis.

**2.1.2 USER REQUIREMENTS ANALYSIS**

User Requirements Analysis is the process of determining user expectations for a new or modified product. These features must be quantifiable, relevant, and detailed. The main user requirements of our project are as follows:

* Internet Facility/ LAN Connection
* CPU i5+
* RAM 8 or 16 GB
* Memory 1GB

**2.1.3 NON-FUNCTIONAL REQUIREMENTS ANALYSIS**

Non-functional requirements describe the general characteristics of a system. They are also known as quality attributes. Some typical non-functional requirements are Performance, Response Time, Throughput, Utilization, and Scalability.

**Performance:**

The performance of a device is essentially estimated in terms of efficiency, effectiveness, and speed.

* Short response time for a given piece of work.
* High throughput (rate of processing work)
* Short data transmission time.

Response Time: Response time is the time a system or functional unit takes to react to a given input.

**2.2 FEASIBILITY STUDY:**

Feasibility Study is a high-level capsule version of the entire process intended to answer a number of questions like What is the problem? Is there any feasible solution to the given problem? Is the problem even worth solving? A feasibility study is conducted once the problem is clearly understood. A feasibility study is necessary to determine that the proposed system is Feasible by considering the technical, Operational, and Economical factors. By having a detailed feasibility study the management will have a clear-cut view of the proposed system. A well-designed feasibility study should provide a historical background of the business or project, the operations and management, marketing research and policies, financial data, legal requirements, and tax obligations. The following possibilities are considered for the project in order to ensure that the project is variable and it does not have any major obstructions. Feasibility study encompasses the following things:

* Technical Feasibility
* Operational Feasibility
* Behavioral feasibility

In this phase, we study the feasibility of all proposed systems And pick the best feasible solution for the problem. The feasibility is studied based on three main factors as follows:

**2.2.1 Technical feasibility:**

In this step, we verify whether the proposed systems are technically feasible or not. i.e., all the technologies required to develop the system are available readily or not. Technical Feasibility determines whether the organization has the technology and skills necessary to carry out the project and how this should be obtained. The system can be feasible because of the following grounds.

* All necessary technology exists to develop the system
* This system is flexible and can be expanded further
* This system can give a guarantee of accuracy, ease of use, and reliability
* Our project is technically feasible because all the technology needed for our project is readily available.

**2.2.2 Operational feasibility:**

In this step, we verify different operational factors of the proposed systems like manpower, time, etc., whichever solution uses less operational resources, is the best operationally feasible solution. The solution should also be operationally possible to implement. Operational Feasibility determines if the proposed system satisfied user objectives could be fitted into the current system operation. The present system Insurance predicting can be justified as operationally feasible based on the following grounds.

* The methods of processing and presentation are completely accepted by the clients since they can meet all user requirements.
* The clients have been involved in the planning and development of the system.
* The proposed system will not cause any problem under any circumstances.

Our project is operationally feasible because the time requirements and personal requirements are satisfied. We are a team of four members and we worked on this project for three working months.

**2.2.3 Behavioral feasibility:**

This device will help people to save time. As there will be no wastage of time, the user will be satisfied.

**2.3 Process model used:**

The model that is basically being followed is the WATERFALL MODEL, which states that the phases are organized in a linear order. First of all the feasibility study is done. Once that part is over the requirement analysis and project planning begins. If a system exists and modification and the addition of a new module are needed, analysis of the present system can be used as a basic model. The design starts after the requirement analysis is complete and the coding begins after the design is complete. Once the programming is completed, the testing is done. In this model, the sequence of activities performed in a software development project is Requirement Analysis, Project Planning, System Design, Detail design, Coding, Unit testing, System integration & testing. Here the linear ordering of these activities is critical. The end of the phase and the output of one phase is the input of the other phase.

The output of each phase is to be consistent with the overall requirement of the system. Some of the qualities of the spiral model are also incorporated after the people concerned with the project review completion of each of the phases of the work done. WATERFALL MODEL was chosen because all requirements were known beforehand and the objective of our software development is the computerization/automation of an already existing manual working system.

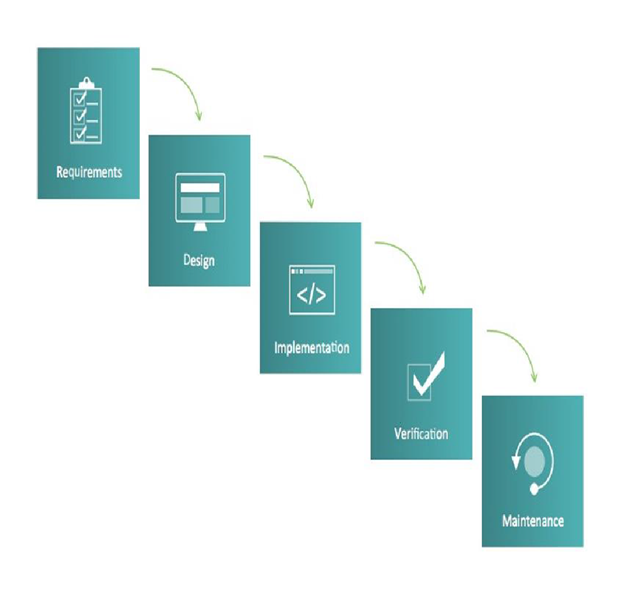
****

Fig. Waterfall model

**2.5 Software and hardware requirements:**

**2.5.1 Software requirements:**

Operating system: Windows 7.

Coding language: Python.

IDE: Python IDLE.

**2.5.2 Hardware requirements:**

System: Intel Core i3 2.4 GHz.

Hard Disk: 500 GB.

RAM: 4GB.

**2.6 SRS Specification:**

Software Requirements Specification (SRS) – a requirements specification for a software system- is a complete description of the behavior of a system to be developed. It includes a set of cases that describe all the interactions users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements that impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System Requirements Specification It is a collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Projects are subject to three sorts of required elements. Business requirements describe in business terms what must be delivered or accomplished to provide value.

* Product requirements describe properties of a system or product (which could be one of several ways to accomplish a set of business requirements.)
* Process requirements describe activities performed by the developing organization. For instance, process requirements could specify methodologies that must be followed, and constraints that the organization must obey.

Product and process requirements are closely linked. Process requirements often specify the activities that will be performed to satisfy a product requirement. For example, a maximum development cost requirement (a process requirement) may be imposed to help achieve a maximum sales price requirement (a product requirement) a requirement that the product is maintainable (a product requirement) often is addressed by imposing requirements to follow particular development styles. A system engineering requirement can be a description of what a system must do, referred to as a Functional Requirement. This type of requirement specifies something that the delivered system must be able to do. Another type of requirement specifies something about the system itself, and how well it performs its functions. Such requirements are often called Nonfunctional requirements, or ‘Performance requirements’ or ‘Quality of service requirements’. Examples of such requirements include usability, availability, reliability, supportability, testability, and maintainability.

A collection of requirements define the characteristics or features of the desired system. A ‘good’ list of requirements as far as possible avoids saying how the system should implement the requirements, leaving such decisions to the system designer. Specifying how the system should be implemented is called “implementation bias” or “solution engineering”. However, implementation constraints on the solution may validly be expressed by the future owner, for example for required interfaces to external systems; for interoperability with other systems; and for commonality with other owned products.

**Functional requirements:**

The Functional Requirements Specification gives the operations and activities that a system must be able to perform. Functional requirements should include functions performed by specific screens, outlines of workflows performed by the system, and other business or compliance requirements the system must meet. It also depends upon the type of software, expected users, and the type of system where the software is used. Some constraints was given as input. This was given as input to code which takes some random information. The output was in the form of predicted output.

1.Data Collection

2.Data Preprocessing

3.Training And Testing

4.Modeling

5.Predicting

**Non-functional requirements:**

In systems engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions. The nonfunctional requirements can be considered as quality attributes of a system.

Performance: The time required to calculate the cost.

Reliability: The system should be 90% reliable. Since it may need some maintenance or preparation for some particular day, the system does not need to be reliable every time. so, 80% reliability is enough.

Efficiency: Based upon the density of given values or input to calculate.

Availability: It is available in the present industries .

Maintainability: The system should be optimized for supportability, or ease of maintenance as far as possible.

**CHAPTER 3**

**DESIGN PHASE**

**Design phase:**

Design is a multi-step process that focuses on data structure, Software architecture, procedural details, and interface between modules. The design process also translates the requirements into the presentation of software that can be accessed for quality before coding begins. Computer software design changes continuously as new methods; better analysis and broader understanding evolved. Software design is at a relatively early stage in its revolution. Therefore, software design methodology lacks the depth, flexibility, and quantitative nature that are normally associated with more classical engineering disciplines. However, the techniques for software design do exist, criteria for design qualities are available and design notation can be applied.

The purpose of the design phase is to plan a solution to the problem specified by the requirements document. The design of a system is perhaps the most critical factor affecting the quality of the software. It has a major impact on the project during later phases, particularly during testing and maintenance.

**Design phase purpose:**

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer’s goal is to produce a model or representation of an entity that will later be built. Beginning, once the system requirements have been specified and analyzed, system design is the first of the three technical activities design, code, and test that is required to build and verify software.

The importance can be stated with the single word “Quality”. Design is the place where quality is fostered in software development. The design provides a representation of software that can be accessed for quality. Design is the only way that can accurately translate a customer’s view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design, we risk building an unstable system that will be difficult to test one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed, reviewed, and documented. System design can be viewed from either a technical or project management perspective. From the technical point of view, design is comprised of four activities- architectural design, data structure design, interface design, and procedural design.

The design model is an abstraction of the implementation of the system. It is used to conceive as well as document the design of the software system. It is a comprehensive, composite artifact encompassing all design classes, subsystems, packages, collaborations, and the relationships between them.

**3.1 Design Concepts:**

The set of fundamental software design concepts are as follows:

1. Abstraction:

The lower level of abstraction provides a more detailed description of the solution. A sequence of instructions that contains a specific and limited function refers to a procedural abstraction. A collection of data that describes a data object is data abstraction.

2. Architecture:

The complete structure of the software is known as software architecture. The structure provides conceptual integrity for a system in a number of ways. The architecture is the structure of program modules where they interact with each other in a specialized way. The aim of the software design is to obtain an architectural framework of a system.

3. Patterns:

A design pattern describes a design structure and that structure solves a particular design problem in a specified content.

4. Modularity:

Modularity is the single attribute of software that permits a program to be managed easily.

5. Information hiding:

Modules must be specified and designed so that the information like algorithm and data presented in a module is not accessible for other modules not requiring that information.

6. Functional independence:

Functional independence is the concept of separation and is related to the concept of modularity, abstraction, and information hiding. The functional independence is accessed using two criteria i.e. Cohesion and coupling. Cohesion is an extension of the information hiding concept. A cohesive module performs a single task and it requires a small interaction with the other components in other parts of the program. Coupling is an indication of interconnection between modules in a structure of software.

7. Refinement:

Refinement is a top-down design approach. It is a process of elaboration. A program is established for refining levels of procedural details.

8. Refactoring:

Refactoring is the process of changing the software system in a way that it does not change the external behavior of the code and still improves its internal structure.

9. Design classes:

The model of software is defined as a set of design classes. Every class describes the elements of the problem domain and that focus on features of the problem which are user-visible.

**3.2 Design Constraints:**

Design Constraints are generally the limitations on a design. They include imposed limitations that you don't control and limitations that are self-imposed as a way to improve a design. The following are common types of design constraints. 9 Types of Design Constraints:

Commercial Constraints:

Basic commercial constraints such as time and budget come under commercial constraints

Requirements:

Requirements specify the basic needs of a project. Ex: Functional requirements.

Non-Functional Requirements:

Non-Functional Requirements are the requirements that specify intangible elements of a design.

Compliance:

Compliance refers to applicable laws, regulations, and standards.

Style:

A style guide or multiple style guides related to an organization, brand, product, service, environment or project. For example, a product development team may follow a style guide for a brand family that constrains the colors and layout of package designs.

Sensory Design:

Beyond visual design, constraints may apply to taste, touch, sound and smell. For example, a brand identity that calls for products to smell fruity.

Usability:

Usability principles imply frameworks and standards. Ex: The principle of least astonishment.

Principles:

Principles include the design principles of an organization, team, or individual. For example, a designer who uses form follows function to constrain designs.

Integration:

A design that needs to work with other things such as products, services, systems, processes, controls, partners, and information.

**3.3 Conceptual Design:**

Conceptual Design is an early phase of the design process, in which the broad outlines of function and form of something are articulated. It includes the design of interactions, experiences, processes, and strategies. It involves an understanding of people's needs - and how to meet them with products, services, & processes. Common artifacts of conceptual design are concept sketches and models.

The unified modeling language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic, and pragmatic rules.

A UML system is represented using five different views that describe the system from a distinctly different perspective. Each view can be defined by a set of diagrams.

UML is specifically constructed through two different domains. They are:

* UML analysis modeling focuses on the user model and structural model views of the system.
* UML design modeling focuses on behavioral modeling, implementation modeling, and environment model views.

**Use case diagram:**

Use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. Actors are the external entities that interact with the system. The use cases are represented by either circles or ellipses.

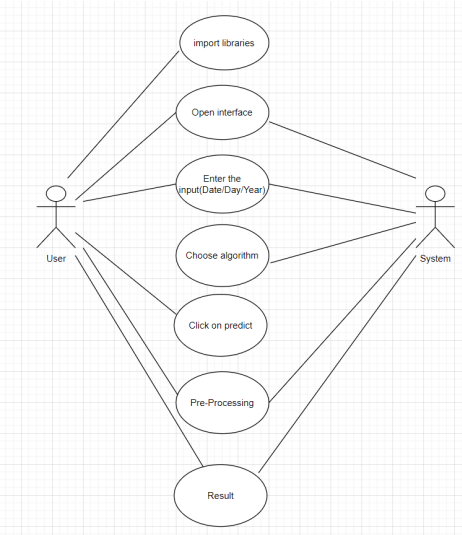


Fig. Use case diagram

**Class diagram:**

Class diagrams give an overview of a system by showing its classes and the relationships among them. Class diagrams are static – they display what interacts but not what happens when they do interact. In general a class diagram consists of some set of attributes and operations. Operations will be performed on the data values of attributes.

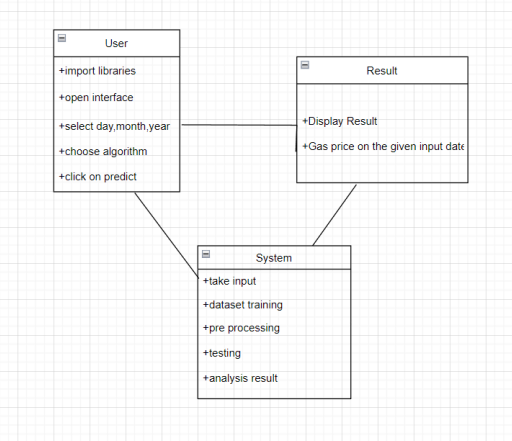


Fig. Class diagram

**3.4 Logical design:**

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract and sometimes graphical model of the actual system.

**Sequence diagram:**

A Sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

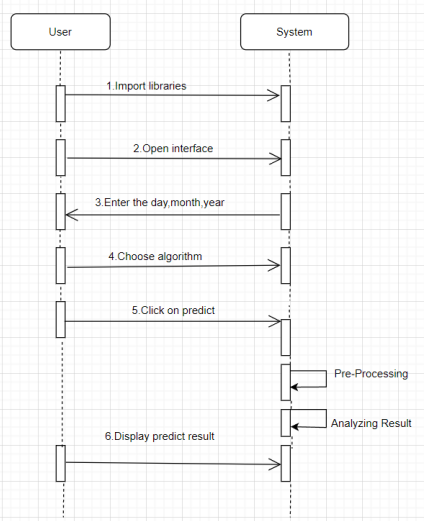


Fig. Sequence diagram

**Activity diagram:**

Activity diagram is essentially a fancy flowchart: Activity and state diagrams are related. Statechart diagram focuses on objects undergoing a process. An activity diagram focuses on the flow of activities involved in a single process. The activity diagram shows the activities depend on one another.

An activity represents the performance of the task or duty in a workflow. It may also represent the execution of a statement in a procedure. You can share activities between state machines. However, transitions cannot be shared.

Activity diagrams provide a way to model the workflow of a business process, code specific information such as a class operation. The transitions are implicitly triggered by the completion of the actions in the source activities.

The main difference between activity and statechart diagrams is activities are activity-centric, while state chart diagrams are state-centric.

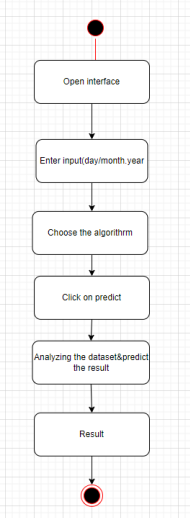


Fig. Activity diagram

**3.5 Algorithmic Design:**

Step 1: Start

Step 2: Open interface

Step 3: Enter input

Step 4: Select Algorithm

Step 5: Submit data

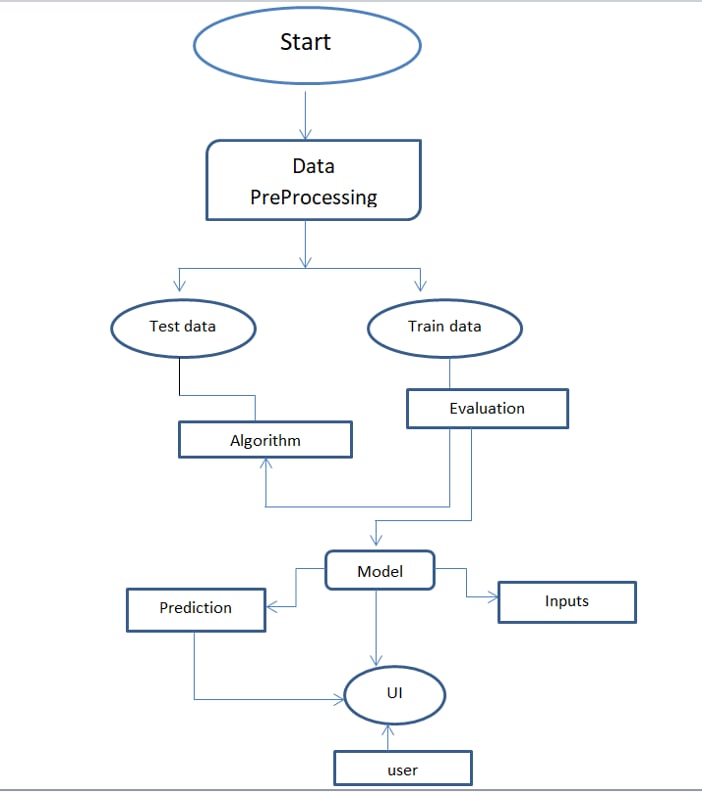
Step 6: Pre-processing

Step 7: Prediction of opening value

Step 8: Result generation

Step 9: Stop

**3.6 ARCHITECTURE:**



**Chapter 4**

**Coding & output screens**

**4.1Sample coding:**

**4.2 Output screens:**

**5. SYSTEM TESTING**

**5.1 TESTING STRATEGIES**

**5.1.1 UNIT TESTING**

Unit testing, a testing technique using which individual modules are tested to determine if there are issues by the developer himself.. it is concerned with functional correctness of the standalone modules. The main aim is to isolate each unit of the system to identify, analyze and fix the defects.

Unit Testing Techniques:

Black Box Testing - Using which the user interface, input and output are tested.

White Box Testing –Used to test each one of those functions behavior is tested.

**5.1.2 DATA FLOW TESTING**

Data flow testing is a family of testing strategies based on selecting paths through the program’s control flow in order to explore sequence of events related to the status of Variables or data object. Dataflow Testing focuses on the points at which variables receive and the points at which these values are used.

**5.1.3 INTEGRATION TESTING**

Integration Testing done upon completion of unit testing, the units or modules are to be integrated which gives raise too integration testing. The purpose of integration testing is to verify the functional, performance, and reliability between the modules that are integrated.

**5.1.4 BIG BANG INTEGRATION TESTING**

Big Bang Integration Testing is an integration testing Strategy wherein all units are linked at once, resulting in a complete system. When this type of testing strategy is adopted, it is difficult to isolate any errors found, because attention is not paid to verifying the interfaces across individual units.

# **5.1.5 USER INTERFACE TESTING**

User interface testing, a testing technique used to identify the presence of defects is a product/software under test by Graphical User interface [GUI].

**5.2 TEST CASES:**

| **S.NO** | **INPUT** | **OUTPUT** | **RESULT** |
| --- | --- | --- | --- |
| **Test Case 1** | The user gives the input in the form of a training with a dataset.  . | An output is predicted as the training is successful . | The result is that the dataset is trained. Therefore the test case1 is passed successfully. |
| **Test Case 2** | The user gives the input in the form of  Opening interface . | An output is predicted as the user opens the interface is successful. | The result is that the user opened the interface. Therefore the testcase2 passed successfully. |
| **Test Case 3** | The user gives the input in the form of entering input. | An output is predicted as the user enters the input is successful. | The result is that the user enters the input. Therefore the test case3 passed successfully. |
| **Test Case 4** | The user gives the input in the form of the click on predict. | An output is predicted as the user gets a result as the price of the natural gas. | The result is that the user gets the predicted price. Therefore the test case4 passed successfully. |

**TABLE: TEST CASES**

**6. Implementation**

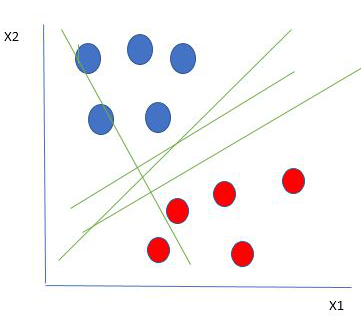
**6.1 Implementation:**

Here we predicted the price of the natural gas by using Machine learning technology. Also here we had mainly used the two algorithms, SVM, Random forest. Some basic information about these algorithms are given below:

**SVM(Support Vector Machine Algorithm):**

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

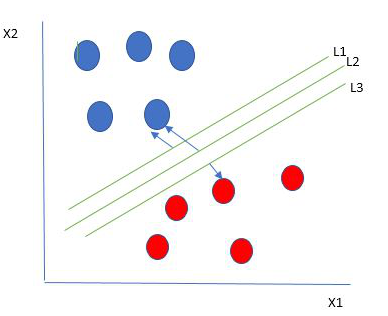
Let’s consider two independent variables x1, x2 and one dependent variable which is either a blue circle or a red circle.



From the figure above its very clear that there are multiple lines (our hyperplane here is a line because we are considering only two input features x1, x2) that segregates our data points or does a classification between red and blue circles. So how do we choose the best line or in general the best hyperplane that segregates our data points.

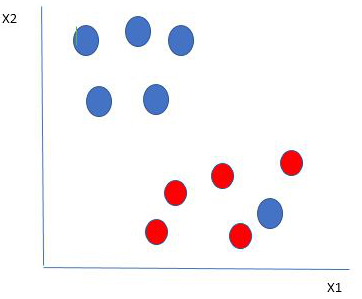
**Selecting the best hyper-plane:**

One reasonable choice as the best hyperplane is the one that represents the largest separation or margin between the two classes.

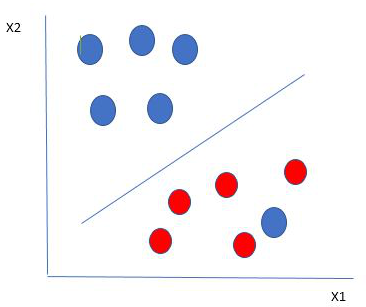


So we choose the hyperplane whose distance from it to the nearest data point on each side is maximized. If such a hyperplane exists it is known as the maximum-margin hyperplane/hard margin. So from the above figure, we choose L2.

Let’s consider a scenario like shown below



Here we have one blue ball in the boundary of the red ball. So how does SVM classify the data? It’s simple! The blue ball in the boundary of red ones is an outlier of blue balls. The SVM algorithm has the characteristics to ignore the outlier and finds the best hyperplane that maximizes the margin. SVM is robust to outliers.



So in this type of data points what SVM does is, it finds maximum margin as done with previous data sets along with that it adds a penalty each time a point crosses the margin. So the margins in these type of cases are called soft margin. When there is a soft margin to the data set, the SVM tries to minimize (1/margin+∧(∑penalty)). Hinge loss is a commonly used penalty. If no violations no hinge loss.If violations hinge loss proportional to the distance of violation.

**SVM Kernel:**

The SVM kernel is a function that takes low dimensional input space and transforms it into higher-dimensional space, ie it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put the kernel does some extremely complex data transformations then finds out the process to separate the data based on the labels or outputs defined.

**Advantages of SVM:**

* Effective in high dimensional cases.
* It is memory efficient as it uses a subset of training points in the decision function called support vectors.
* Different kernel functions can be specified for the decision functions and its possible to specify custom kernels

**RANDOM FOREST:**

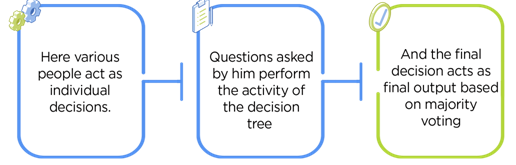
**Introduction**

Random forest is a Supervised Machine Learning Algorithm that is used widely in Classification and Regression problems. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression.

One of the most important features of the Random Forest Algorithm is that it can handle the data set containing continuous variables as in the case of regression and categorical variables as in the case of classification. It performs better results for classification problems.

**Real Life Analogy**

Let’s dive into a real-life analogy to understand this concept further. A student named X wants to choose a course after his 10+2, and he is confused about the choice of course based on his skill set. So he decides to consult various people like his cousins, teachers, parents, degree students, and working people. He asks them varied questions like why he should choose, job opportunities with that course, course fee, etc. Finally, after consulting various people about the course he decides to take the course suggested by most of the people.



random forest 1

**Working of Random Forest Algorithm**

Before understanding the working of the random forest we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus a collection of models is used to make predictions rather than an individual model.

Ensemble uses two types of methods:

1. Bagging– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest.

2. Boosting– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST

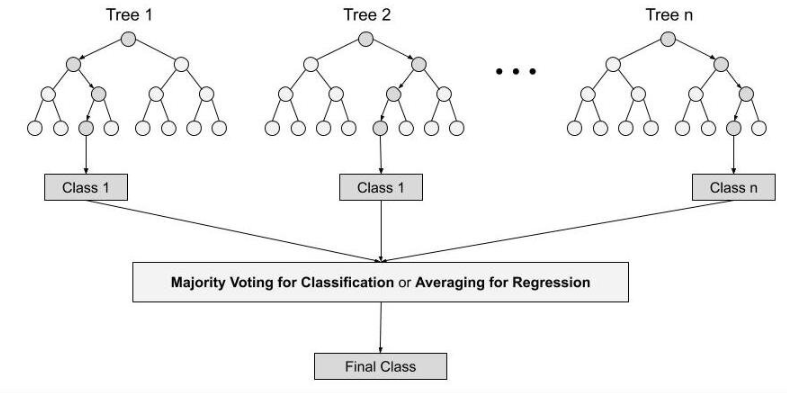
**Steps involved in random forest algorithm:**

Step 1: In Random forest n number of random records are taken from the data set having k number of records.

Step 2: Individual decision trees are constructed for each sample.

Step 3: Each decision tree will generate an output.

Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

****

### **Important Features of Random Forest**

**1. Diversity-** Not all attributes/variables/features are considered while making an individual tree, each tree is different.

**2. Immune to the curse of dimensionality-** Since each tree does not consider all the features, the feature space is reduced.

**3. Parallelization-** Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.

**4. Train-Test split-** In a random forest we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.

**5. Stability-** Stability arises because the result is based on majority voting/ averaging.

**6.2 Implementation steps:**

Step 1: Start

Step 2: Open interface

Step 3: Give input (date, month, year)

Step 4: Select Algorithm

Step 5: Submit data

Step 6: Pre-processing

Step 7: Prediction of opening value

Step 8: Result generation

Step 9: Stop

**6.3 Implementation procedure:**

User has to start the python server, for that the user has to run the ‘FLASK’ command in the prompt. Then the user has to open the interface. After opening the interface the user has to enter the input i.e., Date, Month, Year. Here we used the dataset for training the model. Now the user has to choose the algorithm, and the user can select only one algorithm at a time. After choosing the algorithm the user has to click on the predict/ Submit option. Then the result will be generated as the price of the natural gas on the particular given input date .

**7. Conclusion and future enhancements**

It has always been a difficult task to predict the exact daily price of natural gas price. Many factors such as political events, general economic conditions, and traders’ expectations may have an influence on it. But here, based on the past and present traits, we were able to achieve up to 97% accuracy in predicting the price of any given date. Albeit, its impossible to predict unexpected scenarios such as acts of warfare or terrorism. But, the benefits of having reliable information of what the price of natural gas could be at any given time is paramount, it could make or break economies. And in this case, as this project points out data-driven machine learning models deserve all the attention it could ever garner and even more.

The project has been built using 2 models of prediction namely the Decision Tree method and Random Forest method with the accuracy score of over 97% on both the models (97.4% on Decision Tree and 97.74% on Random Forest Method). By doing some further research and learning the accuracy can be uplifted upto 100% which would be an ideal prediction real- time application which would be much more helpful in the trading sector.

**8. References**

1. Ying, C.; Luh, P. B.; Che, G.; Yige, Z.; Michel, L. D.; Coolbeth, M. A.; Friedland, P. B.; Rourke, S. J. Short-Term

Load Forecasting: Similar Day-Based Wavelet Neural Networks. IEEE Transactions on Power Systems, 2010, vol. 25,

no. 1, pp. 322-330.

2. Feinberg, E. A.; Genethliou, D. Load Forecasting. in Applied Mathematics for Restructured Electric Power Systems,

Chow, J.H., Wu, F.F., Momoh, J. Eds., Springer, US, 2005. pp 269-285

3. Moser, M., Jordan, M., Petsche, T. Eds., MIT Press, Cambridge, MA, 1996, pp. 281-287.

4. Felice, M. D.; Xin, Y. Short-Term Load Forecasting with Neural Network Ensembles: A Comparative Study

[Application Notes]. Computational Intelligence Magazine: Springer Science, 2011, vol. 6, no. 3, pp. 47-56.

5. Park, D. C.; El-Sharkawi, M. A.; Marks, R. J.; Atlas, L. E.; Damborg, M. J. Electric load forecasting using an artificial

neural network. IEEE Transactions on Power Systems, 1991, vol. 6, no. 2, pp. 442-449.

6. Liu, K.; Subbarayan, S.; Shoults, R. R.; Manry, M. T.; Kwan, C.; Lewis, F. I.; Naccarino, J. Comparison of very short-

term load forecasting techniques. IEEE Transactions on Power Systems, 1996, vol. 19, no. 1, pp. 164-173.

7. Abdel-Aal, R. E. Short-term hourly load forecasting using abductive networks. IEEE Transactions on Power Systems,

2004, vol. 19, no. 1, pp. 164-173.

8. Fan, S.; Chen, C. Short-term load forecasting based on an adaptive hybrid method. IEEE Transactions on Power

Systems, 2006, vol. 21, no. 1, pp. 392-401.

9. Buchananan, W.; Hodges, P.; Theis, J. Which way the natural gas price: an attempt to predict the direction of the

natural gas spot price movements using trader positions. Energy Economics, 2001, vol. 23, pp. 279-293.

10. Nguyen, H. T.; Nabney, I. T.; Short-term electricity demand and gas price forecasts using wavelet transforms and

adaptive models. Energy, 2010, vol. 35, no. 9, pp. 3674-3685.

11. Salehnia N. et al. Forecasting natural gas spot prices with nonlinear modeling using Gamma test analysis. J. of Nat.

Gas Sci. and Eng., 2013, vol. 14, pp. 238-249.

12. Serletis, A.; Shahmoradi, A. Returns and volatility in the NYMEX Henry Hub natural gas futures market. OPEC

Review: Energy Economics and related issues, 2006, pp. 171-186.