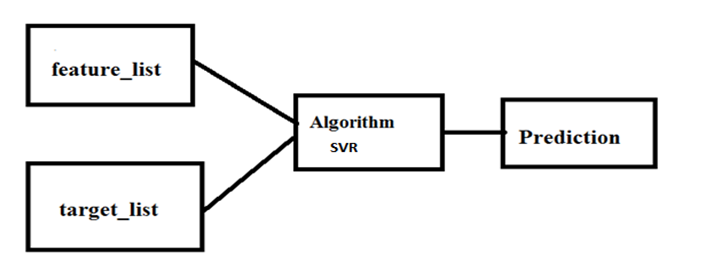
**INTRODUCTION**

**1. INTRODUCTION**

As we are experiencing an unstable increase in prices of oil prices where the oil prices are dependent on the crude oil prices of Dubai and Saudi-Arabia. The transportation will be affected for change in prices. The crude oil price of India has also been taken into consideration for the accurate prediction of prices. We have used various algorithms for predicting the diesel price in India. The algorithms which we have used are Random Forest, Support Vector Regression (RBF model, polynomial model, and linear model), and Linear Regression. The Prediction of crude oil 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 logistic regression model which is feasible to some extend for the prediction of the crude oil prices. The implementation is on predicting the crude oil prices for the days using linear regression Python machine learning Algorithm and

plotting the graph based on prediction[12].



**Figure 1.1: Architecture**

**1.1 HARDWARE SPECIFICATIONS**

**System :** Pentium IV 2.4 GHz.

**Hard Disk :** 40 GB.

**Monitor :** 14’ Colour Monitor.

**Mouse :** Optical Mouse.

**Ram :** 4 GB

**1.2 SOFTWARE SPECIFICATIONS**

**Operating system :** Windows 10.

**Coding Language :** Python 3.

**1.2.1 NumPy**

NumPy is a Python package which stands for ‘Numerical Python’. It is the core library for scientific computing, which contains a powerful n-dimensional array object, and provides tools for integrating C, C++ etc. It is also useful in linear algebra, random number capability etc. We can simply use a statement “import numpy as np” to use numpy.

**1.2.2 Pandas**

Pandas is used for data manipulation, analysis and cleaning. Python pandas is well suited for different of data such as:

* Tabular data with heterogeneously-typed columns.
* Ordered and unordered time series data.
* Arbitrary matrix data with row & column labels.
* Unlabelled data.
* Any other form of observational or statistical data sets.​

We can simply use a statement “import pandas as pd” to use pandas.

**1.2.3 Matplotlib**

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram. We can simply use a statement

“import matplotlib as mp”.

**1.2.4 Sci-kit Learn**

Scikit-learn is probably the most useful library for machine learning in Python. It is on Numpy, SciPy and matplotlib, this library contains a lot of efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction. Note that scikit-learn is used to build models. It should not be used for reading the data, manipulating and summarizing it. There are better libraries for that (e.g NumPy, Pandas etc.)

**1.2.5 PyCharm**

**PyCharm** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), specifically for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) language. PyCharm is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), with [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) versions. PyCharm is available in three editions: **Professional**, **Community**, and **Edu**. The **Community** and **Edu** editions are open-source projects and they are free, but they have less features. PyCharm Edu provides courses and helps you learn programming with Python. The **Professional** edition is commercial, and provides an outstanding set of tools and features.

**Steps to install:**

1.Download the installer .exe from the [Toolbox App web page](https://www.jetbrains.com/toolbox/app/).

2.Run the installer and follow the wizard steps.

After you run the Toolbox App, click its icon in the notification area and select which product and version you want to install.

**1.2.6 Keras :**

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

Keras contains numerous implementations of commonly used neural network building blocks

such as layers, objectives, activation functions, optimizers, and a host of tools to make working

with image and text data easier.

The code is hosted on GitHub, and community support forums include the GitHub issues

page, and a Slack channel. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU).

**1.2.7 TensorFlow :**

TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

**1.2.8 h5py**

The h5py package is a Pythonic interface to the HDF5 binary data format. It lets you store huge amounts of numerical data, and easily manipulate that data from NumPy. For example, you can slice into multi-terabyte data sets stored on disk, as if they were real NumPy arrays.

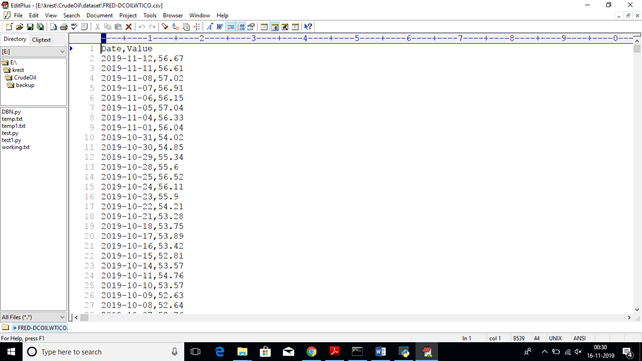
**1.2.9 Tkinter**

**Tkinter** is a [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) [binding](https://en.wikipedia.org/wiki/Language_binding) to the [Tk](https://en.wikipedia.org/wiki/Tk_(software)) [GUI](https://en.wikipedia.org/wiki/Graphical_user_interface) toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's [*de facto* standard](https://en.wikipedia.org/wiki/De_facto_standard) GUI. Tkinter is included with standard [Linux](https://en.wikipedia.org/wiki/Linux), [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows) and [Mac OS X](https://en.wikipedia.org/wiki/Mac_OS_X) installs of Python.

**DATASET DESCRIPTION**

**2.DATASET DESCRIPTION**

In this project we are using historical crude oil prices from the QUANDL website as dataset and this dataset saved inside ‘dataset’ folder. This dataset contains two values such as DATE and price value. See below screen of dataset values.



**Figure 2.1 Image of data set**

Date : displays the date

Value : shows the cost of crude oil on particular date

**SYSTEM ANALYSIS**

**3.SYSTEM ANALYSIS**

**3.1 EXISITING SYSTEM:**

They had been used various algorithms for predicting the diesel price in India. The algorithms which have been used are random forest, support vector regression (rbf) model, polynomial model, and linear model, and linear regression [7].

**3.1.1 Disadvantages of Existing System**

* Using Linear Regression algorithm gives less approximate prediction compared to SVR Algorithm in the proposed model in the project.
* As well the feature\_list and target\_list fitted into the algorithm gives less predicting prices compared to the SVR.
* Comparatively Linear regression performs poorly when there are non-linear relationships.
* They are not naturally flexible enough to capture more complex patterns, and adding the right interaction terms or polynomials can be tricky and time-consuming.

**3.2 PROPOSED SYSTEM**

We are using 4 deep learning algorithms such as LSTM (long short term memory network), DBN(deep belief network), ARMA and RW(random walk) to forecast crude oil prices. We are using above those four algorithms but in 2019 DBN and random walk for prediction is deprecated in python. Now only LSTM and ARMA is available in python so we are using those 2 algorithms to build training model with crude oil data set and then forecasting/predicting prices.

**3.2.1 Advantages of proposed System**

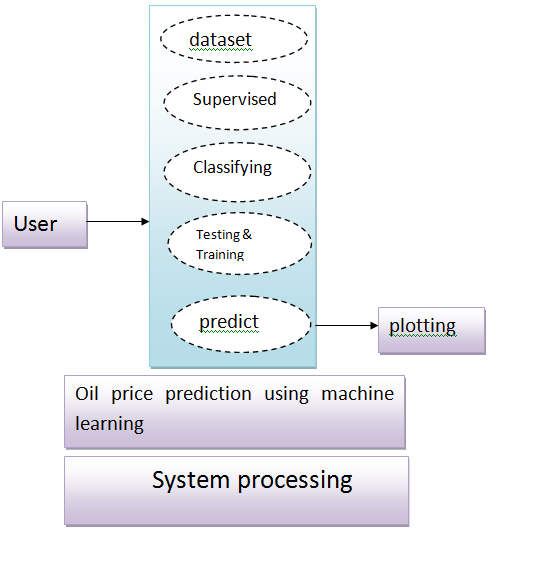
To predict the values of petroleum and diesel in the mere future, we have decided to use the machine learning algorithm after choosing the set of algorithms we have chosen the linear regression algorithm, which has given the most accurate results.

**3.3 OBJECTIVE**

The purpose of our project is to predict future crude oil prices by giving current price of the crude oil.

**3.4 SCOPE**

Based on our assessment of the applications in the field today, a majority of applications are used in the field of predicting oil prices. The scope of this is to achieve predicted price with best accuracy.



**Figure 3.2 Data Flow Diagram**

**SYSTEM STUDY**

**4. SYSTEM STUDY**

**4.1 FEASIBILITY STUDY**

The feasibility of the project is analysed 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.

**4.1.1 Economical Feasibility**

This study is carried out to check the economic impact that the system will have on the organization. The amount of funds that the company can pour into the research and development of the system is limited. The expenditures must be justified. Only the customized products had to be purchased. Our project requires less cost to done and maintain, so it is economically feasible.

**4.1.2 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. 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 for implementing this system. Our project contains all the technology needed to develop.

**4.1.3 Social 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. Our project gives more accurate result as expecting, so it’s an efficient one.

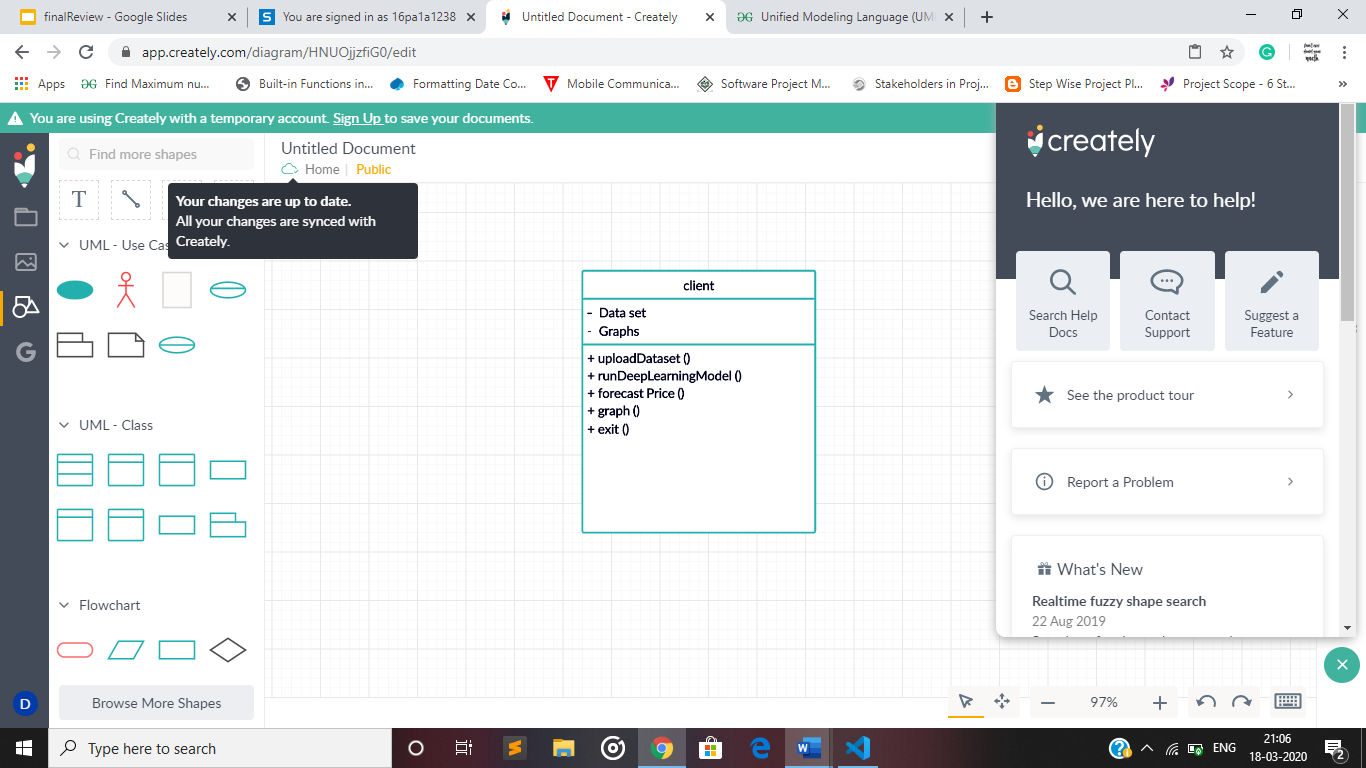
**SYSTEM DESIGN**

**5.SYSTEM DESIGN**

UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artefacts of software systems, as well as for business modelling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**5.1 Class Diagram**

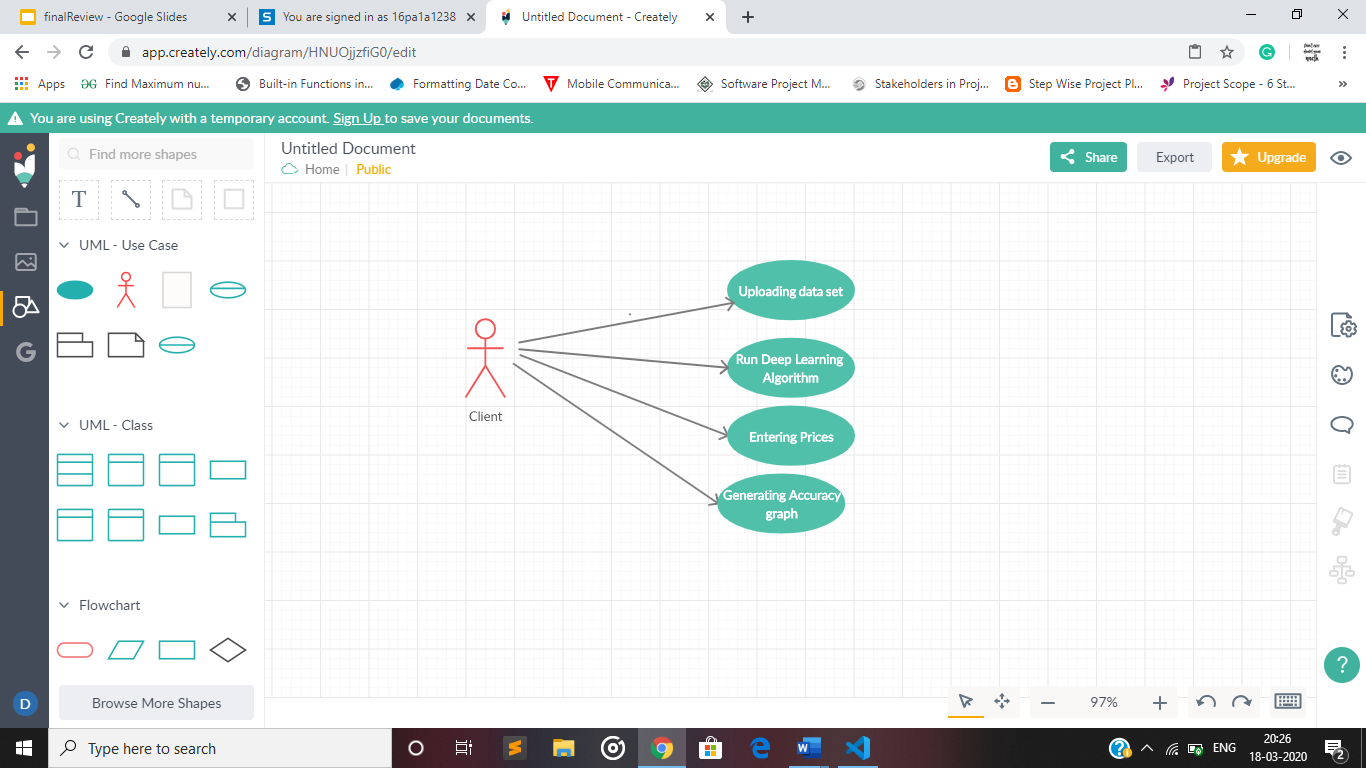
In software engineering, a class diagram in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class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



**Figure 5.1: Class Diagram**

**5.2 Use case Diagram**

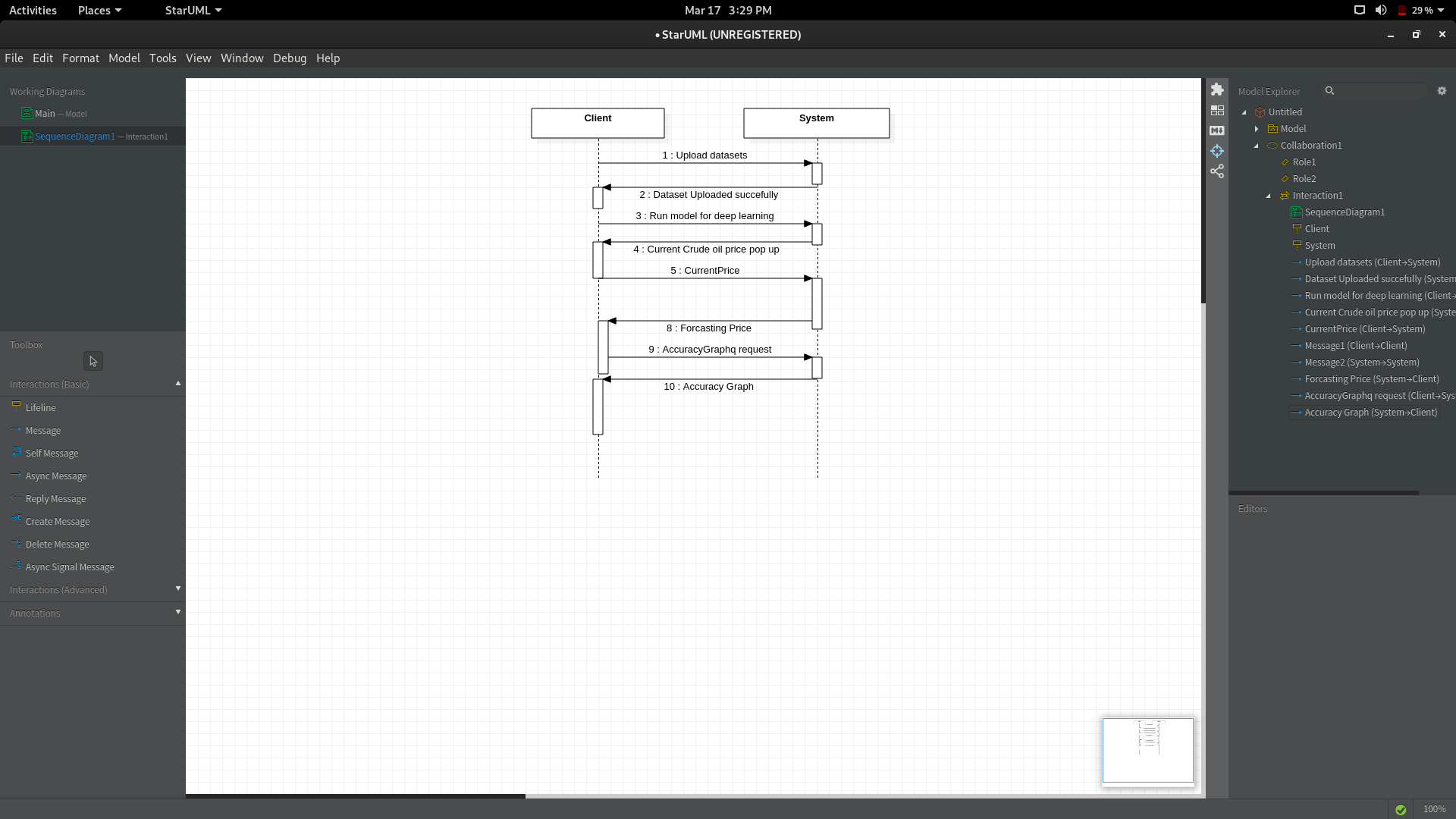
A use case diagram in the Unified Modelling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies 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.



**Figure 5.2: Use case Diagram**

**5.3 Sequence Diagram**

A sequence diagram in Unified Modelling 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.

****

**Figure 5.3: Sequence Diagram**

**MODELS USED**

**6. MODELS USED**

**6.1 INTRODUCTION TO ARTIFICIAL INTELLIGENCE**

“The science and engineering of making intelligent machines, especially intelligent computer programs”. -John McCarthy.

Artificial Intelligence is an approach to make a computer, a robot, or a product to think how smart human think. AI is a study of how human brain think, learn, decide and work, when it tries to solve problems. And finally this study outputs intelligent software systems. The aim of AI is to improve computer functions which are related to human knowledge, for example, reasoning, learning, and problem-solving[21].

The intelligence is intangible. It is composed of

* Reasoning
* Learning
* Problem Solving
* Perception
* Linguistic Intelligence

The objectives of AI research are reasoning, knowledge representation, planning, learning, natural language processing, realization, and ability to move and manipulate objects. There are long-term goals in the general intelligence sector. Approaches include statistical methods, computational intelligence, and traditional coding AI. During the AI research related to search and mathematical optimization, artificial neural networks and methods based on statistics, probability, and economics, we use many tools. Computer science attracts AI in the field of science, mathematics, psychology, linguistics, philosophy and so on.

**6.2 MACHINE LEARNING**

**6.2.1 Introduction**

Machine learning is a subfield of artificial intelligence (AI) [22] . The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve.

Machine Learning algorithms instead allow for computers to train on data inputs and use statically analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate

Decision - making processes based on data inputs.

Any technology user today has benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type[11].

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyse the impact of machine learning processes[10].

Let us look into the common machine learning methods of supervised and unsupervised learning, and common algorithmic approaches in machine learning, including the k-nearest neighbour algorithm, decision tree learning, and deep learning. We’ll explore which programming languages are most used in machine learning, providing you with some of the positive and negative attributes of each. Additionally, we’ll discuss biases that are perpetuated by machine learning algorithms, and consider what can be kept in mind to prevent these biases when building algorithms[16][6].

## 6.2.2 Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labeled by humans, and **unsupervised learning** which provides the algorithm with no labeled data in order to allow it to find structure within its input data. Let’s explore these methods in more detail.

### 6.2.2.1 Supervised Learning

In supervised learning, the computer is provided with example inputs that are labeled with their desired outputs. The purpose of this method is for the algorithm to be able to “learn” by comparing its actual output with the “taught” outputs to find errors, and modify the model accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabelled data.

By being trained on this data, the supervised learning algorithm should be able to later identify unlabelled shark images as fish and unlabelled ocean images as water. A common use case of supervised learning is to use historical data to predict statistically likely future events[19].

**6.2.2.2 Unsupervised Learning**

### In unsupervised learning, data is unlabelled, so the learning algorithm is left to find commonalities among its input data. As unlabelled data are more abundant than labeled data, machine learning methods that facilitate unsupervised learning are particularly valuable.

The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

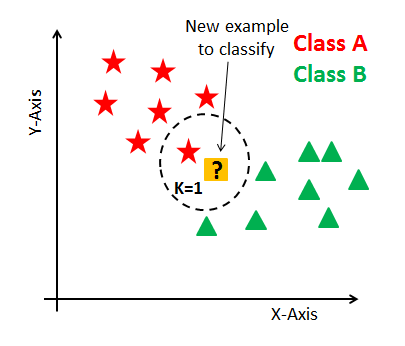
Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases.

Without being told a “correct” answer, unsupervised learning methods can look at complex data that is more expansive and seemingly unrelated in order to organize it in potentially meaningful ways. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together [19].

### 6.3 K-NEAREST NEIGHBOUR

The k-nearest neighbour algorithm is a pattern recognition model that can be used for classification as well as regression. Often abbreviated as k-NN, the **k** in k-nearest neighbour is a positive integer, which is typically small. In either classification or regression, the input will consist of the k closest training examples within a space. We will focus on k-NN classification. In this method, the output is class membership. This will assign a new object to the class most common among its k nearest neighbours. In the case of k =1, the object is assigned to the class of the single nearest neighbour.

When a new object is added to the space — in this case a green heart — we will want the machine learning. Among the most basic of machine learning algorithms, k-nearest neighbour is considered to be a type of “lazy learning” as generalization beyond the training data does not occur until a query is made to the system [8].



**Figure 6.1 KNN Example**

**6.4 LINEAR REGRESSION**

In [statistics](https://en.wikipedia.org/wiki/Statistics), **linear regression** is a [linear](https://en.wikipedia.org/wiki/Linearity) approach to modeling the relationship between a scalar response (or [dependent variable](https://en.wikipedia.org/wiki/Dependent_variable)) and one or more [explanatory variables](https://en.wikipedia.org/wiki/Explanatory_variable) (or [independent variables](https://en.wikipedia.org/wiki/Independent_variable)). The case of one explanatory variable is called [simple linear regression](https://en.wikipedia.org/wiki/Simple_linear_regression). For more than one explanatory variable, the process is called **multiple linear regression.**

In linear regression, the relationships are modelled using [linear predictor functions](https://en.wikipedia.org/wiki/Linear_predictor_function) whose unknown model [parameters](https://en.wikipedia.org/wiki/Parameters) are [estimated](https://en.wikipedia.org/wiki/Estimation_theory) from the [data](https://en.wikipedia.org/wiki/Data). Such models are called [linear models](https://en.wikipedia.org/wiki/Linear_model)[4].

**6.5 Deep Learning**

**6.5.1 What is deep learning**

Deep learning is a branch of [machine learning](https://www.geeksforgeeks.org/introduction-machine-learning/) which is completely based on [artificial neural networks](https://www.geeksforgeeks.org/tag/neural-network/), as neural network is going to mimic the human brain so deep learning is also a kind of mimic of human brain. In deep learning, we don’t need to explicitly program everything. The concept of deep learning is not new. It has been around for a couple of years now. It’s on hype nowadays because earlier we did not have that much processing power and a lot of data. As in the last 20 years, the processing power increases exponentially, deep learning and machine learning came in the picture [6].

A formal definition of deep learning is- neurons Deep learning is a particular kind of

machine learning that achieves great power and flexibility by learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones.

In human brain approximately 100 billion neurons all together this is a picture of an individual neuron and each neuron is connected through thousand of their neighbours.

**6.5.2 Long Short Term Memory Network**

LSTM algorithm is a deep learning famous algorithms used to train and predict any kind of data such as voice recognition, image classification or data classification, this algorithm consist of three layers called input, output, forget layer and at input layer algorithm will read features from dataset and find out best features and saved that best features in output layer and this process continue to filter features between input and output layer and if best output found then assign it current output and old output will be assigned to forgot layer. Final filtered features will be used to train model. While prediction algorithm accept test data and then apply that test data on train model to predict accurate class to which this test data belongs [5].

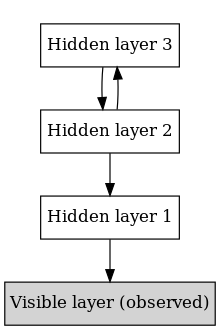
**6.5.3 Deep Belief Network**

In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), a deep belief network (DBN) [20]. is a [generative](https://en.wikipedia.org/wiki/Generative_model) [graphical model](https://en.wikipedia.org/wiki/Graphical_model), or alternatively a class of [deep](https://en.wikipedia.org/wiki/Deep_learning) [neural network](https://en.wikipedia.org/wiki/Artificial_neural_network), composed of multiple layers of [latent variables](https://en.wikipedia.org/wiki/Latent_variables) ("hidden units"), with connections between the layers but not between units within each layer.

When trained on a [set of examples](https://en.wikipedia.org/wiki/Training_set) [without supervision](https://en.wikipedia.org/wiki/Unsupervised_learning), a DBN can learn to probabilistically reconstruct its inputs. The layers then act as [feature detectors](https://en.wikipedia.org/wiki/Feature_learning). After this learning step, a DBN can be further trained with [supervision](https://en.wikipedia.org/wiki/Supervised_learning) to perform [classification](https://en.wikipedia.org/wiki/Statistical_classification)[6].

DBNs can be viewed as a composition of simple, unsupervised networks such as [restricted Boltzmann machines](https://en.wikipedia.org/wiki/Restricted_Boltzmann_machine) (RBMs) or [autoencoders](https://en.wikipedia.org/wiki/Autoencoder), where each sub-network's hidden layer serves as the visible layer for the next. An RBM is an [undirected](https://en.wikipedia.org/wiki/Undirected_graph), generative energy-based model with a "visible" input layer and a hidden layer and connections between but not within layers.

This composition leads to a fast, layer-by-layer unsupervised training procedure, where [contrastive divergence](https://en.wikipedia.org/wiki/Contrastive_divergence) is applied to each sub-network in turn, starting from the "lowest" pair of layers (the lowest visible layer is a [training set](https://en.wikipedia.org/wiki/Training_set))[17].

****

**Figure 6.2 DBN**

**6.5.4 Auto Regressive Moving Average**

ARMA algorithm will also use for same training and testing model but its accuracy of prediction is less compare to LSTM [2].

**6.5.5 Random Walk**

A random walk, mathematically, is something that can be described in several different technical ways. Some describe it as a randomized collection of variables; others might call it a "[stochastic](https://www.techopedia.com/definition/16448/stochastic) process." Regardless, the random walk contemplates a scenario where a variable set takes a path that is a pattern based on random increments [3].

**EXPERIMENTAL SETUP USED**

**7.EXPERIMENTAL SETUP USED**

The steps we followed while developing this project are-:

1. Analysis of the problem statement.
2. Gathering of the requirement specification.
3. Analysation of the feasibility of the project.
4. Development of a general layout.
5. Going by the journals regarding the previous related works in this field.
6. Choosing the method for developing the algorithm.
7. Analyzing the various pros and cons.
8. Starting the development of a project.
9. Installation of software like PYCHARM.
10. Developing an algorithm.
11. Analysis of algorithm by guide.
12. Coding as per the developed algorithm in PYTHON[18].

**7.1 Training & Testing Dataset :**

In machine learning, the study and construction of algorithms that can learn from and make predictions on data is a common task. Such algorithms work by making data driven predictions or decisions, through building a mathematical model from input data. The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model.

The model is initially fit on a ​**training dataset**​, that is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model. The model (e.g. a neural net or a DBN) is trained on the training dataset using a supervised learning method (e.g. gradient descent or stochastic gradient descent). In practice, the training dataset often consists of pairs of an input vector and the corresponding answer vector or scalar, which is commonly denoted as the target. The current model is run with the training dataset and produces a result, which is then compared with the target, for each input vector in the training dataset. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

Successively, the fitted model is used to predict the responses for the observations in a second dataset called the validation dataset. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters (e.g. the number of hidden units in a neural network). ​**Validation datasets** can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset.

This simple procedure is complicated in practice by the fact that the validation dataset's error may fluctuate during training, producing multiple local minima. This complication has led to the creation of many ad-hoc rules for deciding when overfitting has truly begun.

Finally, the test dataset is a dataset used to provide an unbiased evaluation of a final model fit on the training dataset.

**EVALUATING MODEL**

**8. EVALUATING MODEL**

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future. Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and overfitted models. There are two methods of evaluating models in data science, Hold- Out and Cross-Validation. To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance. We have used many evaluation metrics to evaluate our model i.e how accurately the model is working[13][14].

**8.1 Evaluation Metrics**

Model valuation metrics are required to quantify model performance. The choice of evaluation metrics depends on a given machine learning task (such as classification, regression, ranking, clustering, topic modelling, among others). Some metrics, such as precision-recall, are useful for multiple tasks. Supervised learning tasks such as classification and regression constitute a majority of machine learning applications. In this article, we focus on metrics for these two supervised learning models.

**8.1.1 Accuracy**

Accuracy is one metric for evaluating classification models. Informally, accuracy is the fraction of predictions our model got right. Formally, accuracy has the following definition: Accuracy = Number of correct predictions/Total number of predictions For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:

Accuracy = TP+TN/ TP+TN+FP+FN

Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives

We find the accuracy of a model by using the command “accuracy score” in sklearn metrics module of python.

**8.1.2 Precision and Recall**

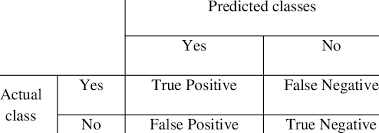
In pattern recognition, information retrieval and classification (machine learning), precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of the total amount of relevant instances that were actually retrieved. Both precision and recall are therefore based on an understanding and measure of relevance.

Precision = TP/ TP+FP

Recall = TP/ TP+FN

**8.1.3 Confusion Matrix**

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes prediction. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

****

**Figure 8.1 Actual and predicted classifier**

**8.1.4 True Positive Rate (Sensitivity)**

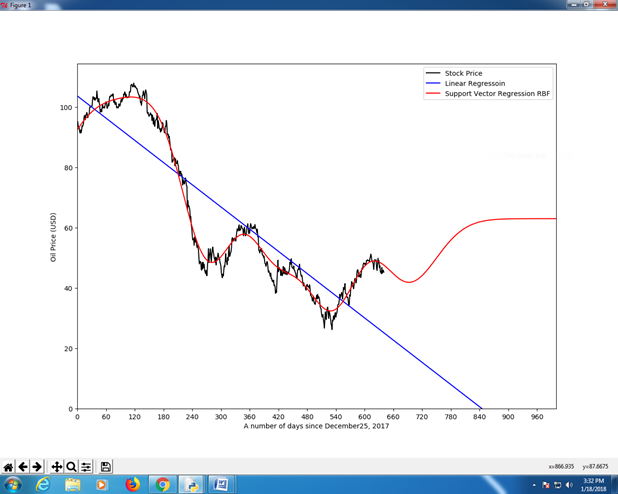
Sensitivity (also called the true positive rate, the recall, or probability of detection in some fields) measures the proportion of actual positives that are correctly identified as such (e.g., the percentage of sick people who are correctly identified as having the condition).

**8.1.5 False Positive Rate (Specificity)**

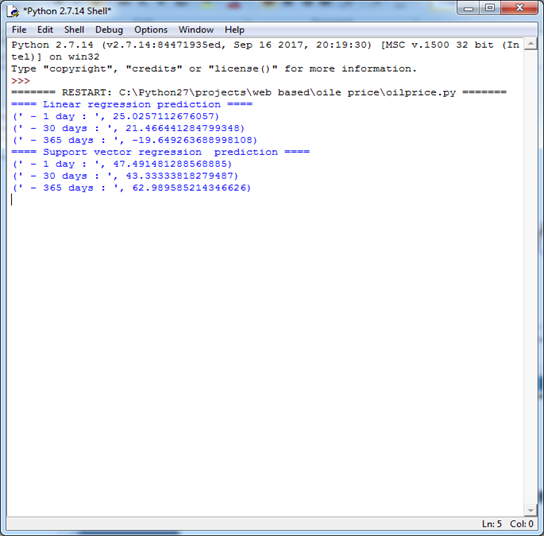
Specificity (also called the true negative rate) measures the proportion of actual negatives that are correctly identified as such (e.g., the percentage of healthy people who are correctly identified as not having the condition).

**8.2 Results**

The following are the results of different algorithms which have different metrics which helps in predicting which model will be best suitable and also to how much extent a particular model is accurate and so on.

****

**Figure 8.2 plot of oil price and dates**

****

**Figure 8.3 Example Output for crude oil price prediction**

**INPUT DESIGN AND OUTPUT DESIGN**

**9.INPUT AND OUTPUT DESIGN**

**9.1 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

What data should be given as input?

How the data should be arranged or coded?

The dialog to guide the operating personnel in providing input.

Methods for preparing input validations and steps to follow when error occur [9].

**9.1.1 OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

**9.2 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right

output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following.

**9.2.1 Objectives:**

• Convey information about past activities, current status or projections of the Future.

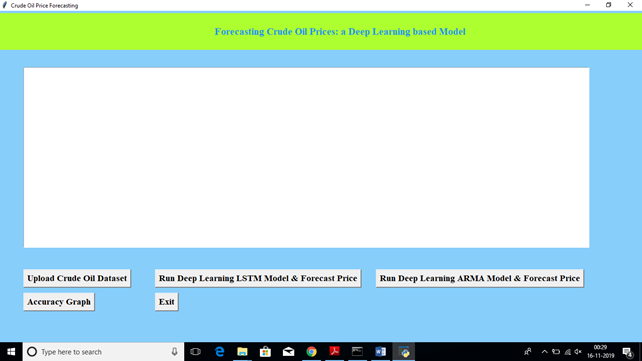
• Signal important events, opportunities, problems, or warnings.

• Trigger an action.

• Confirm an action.

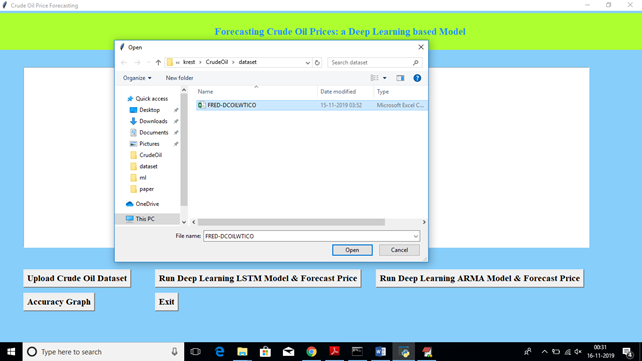
**SCREENSHOTS**

**10.FRONT SCREENS**

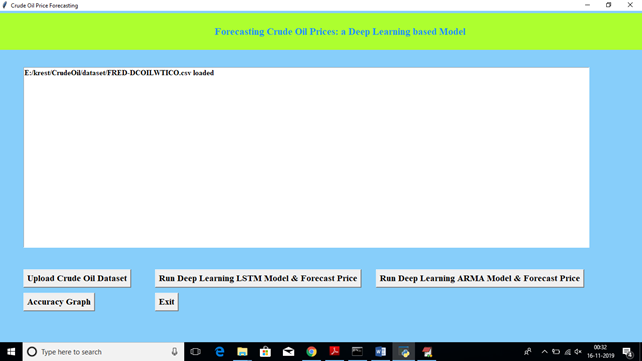


**Figure 10.1 Uploading dataset**

In the above screen click on “upload crude oil dataset” to upload the dataset.

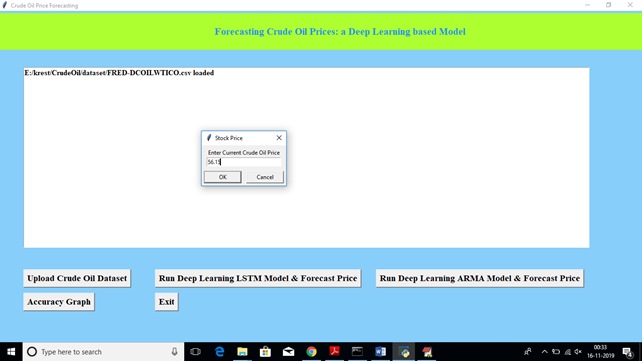


**Figure 10.2 Selecting dataset**



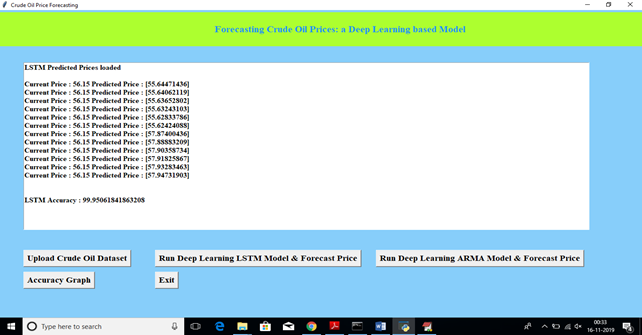
**Figure 10.3 Uploaded dataset image**

After uploading dataset click on ‘Run Deep Learning LSTM Model & Forecast Price’ button to allow LSTM to read dataset and then generate model. Here we need to enter current oil price in next dialog box to predict future prices.

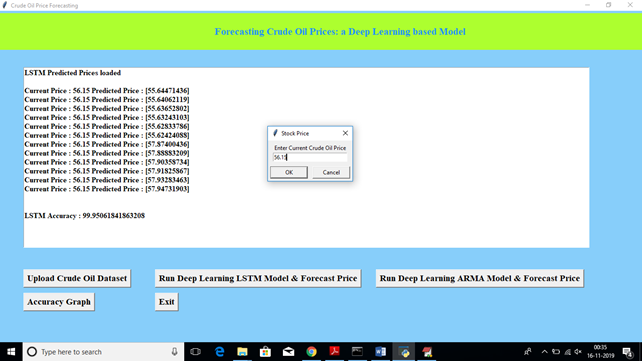


**Figure 10.4 Dialog box for entering current price of crude oil**

In above screen I entered current price as ’56.15’ and then we will get 10 new predicted prices.



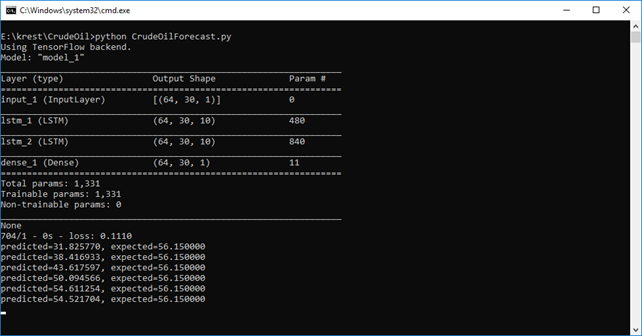
**Figure 10.5 Shows current and predicted price as output using LSTM**

In above screen we can see current price and new best possible forecast prices. We will get LSTM prediction accuracy also. Now click on ‘Run Deep Learning RAM Model & Forecast Price’ button to allow application to read dataset and to generate model. Here also we need to enter current price to get future prices.I

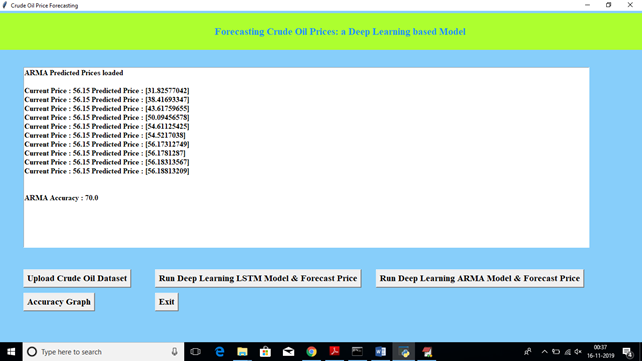
**Figure 10.6 Dialog box for entering current price**

In the above screen for ARMA algorithm also I gave current price as 56.15 and now below are the predicted prices.

In below black console also we can see algorithms process details.



**Figure 10.7 Predicted prices using ARMA in command prompt**



**Figure 10.8 Current and predicted prices using ARMA**

Above are the predicted prices with ARMA.

In above screen we can see predicted prices from ARMA and we can also see its prediction accuracy. Now click on “Accuracy Graph’ button to to see accuracy in graph.



**Figure 10.9 Image of accuracy graph**

In above graph x-axis represents algorithm name and y-axis represents accuracy. From above

graph we can say LSTM perform better than ARMA.

**SYSTEM TEST**

**11.SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### 11.1 TYPES OF TESTS

**11.1.1 Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# 11.1.2 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**11.1.3 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 centred 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 must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**11.1.4 Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**11.1.5 White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

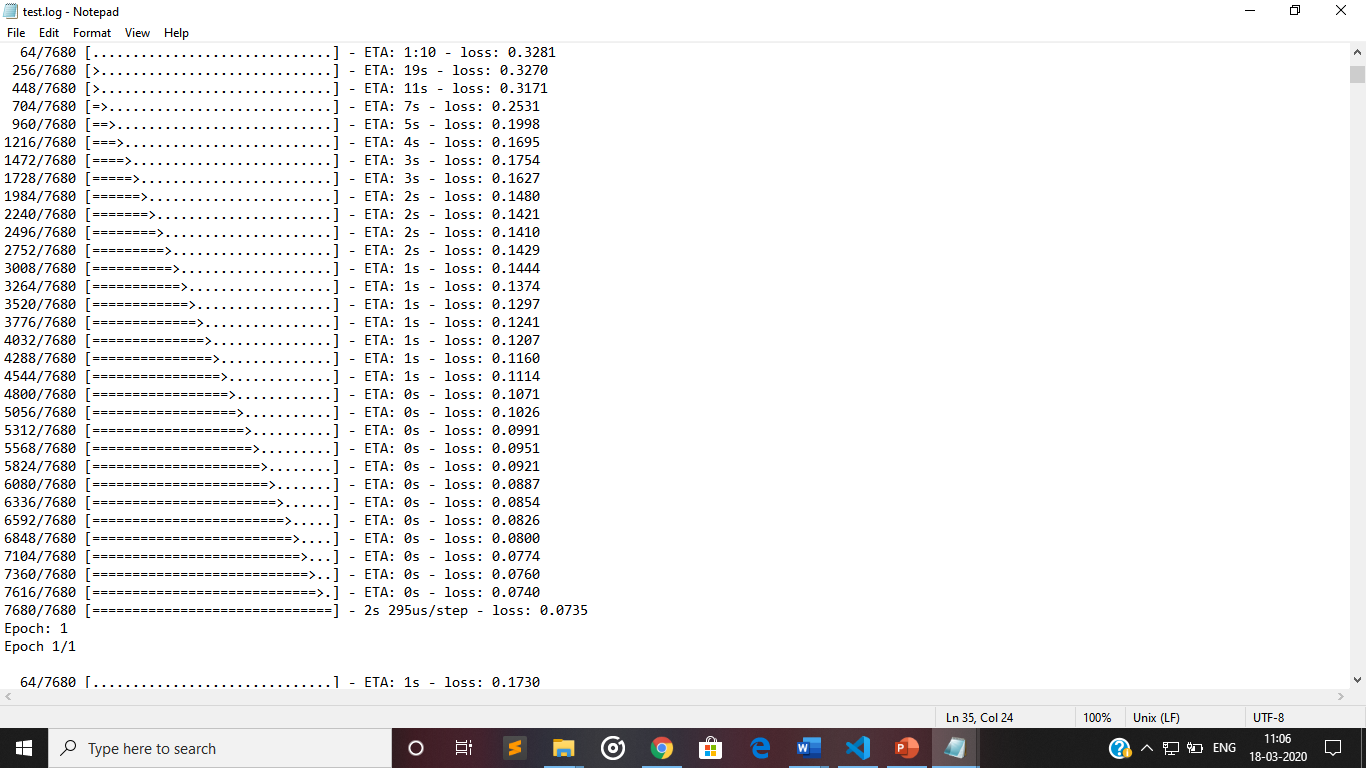
**11.1.6 Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document,

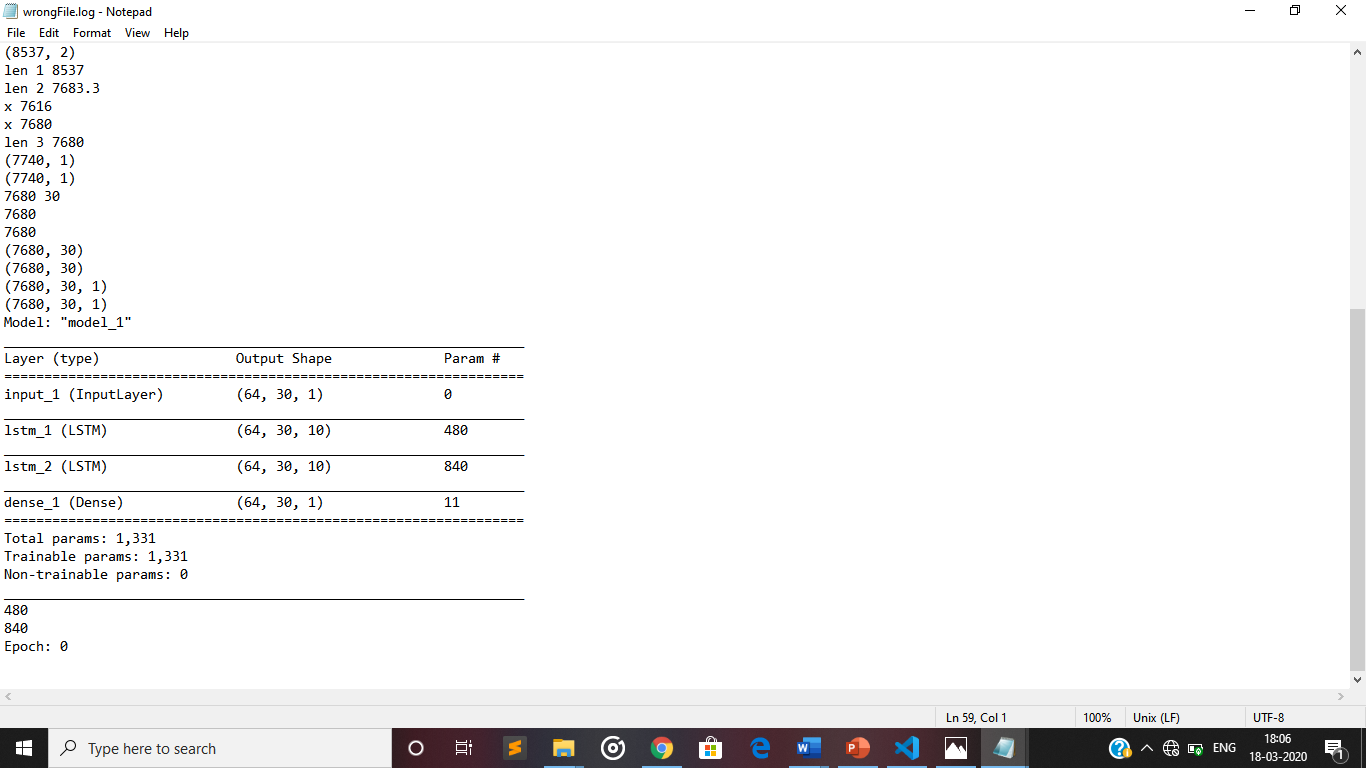
such as specification or requirements document. It is a testing in which the software under test is treated, as a black box. You cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### 11.1 TEST CASES

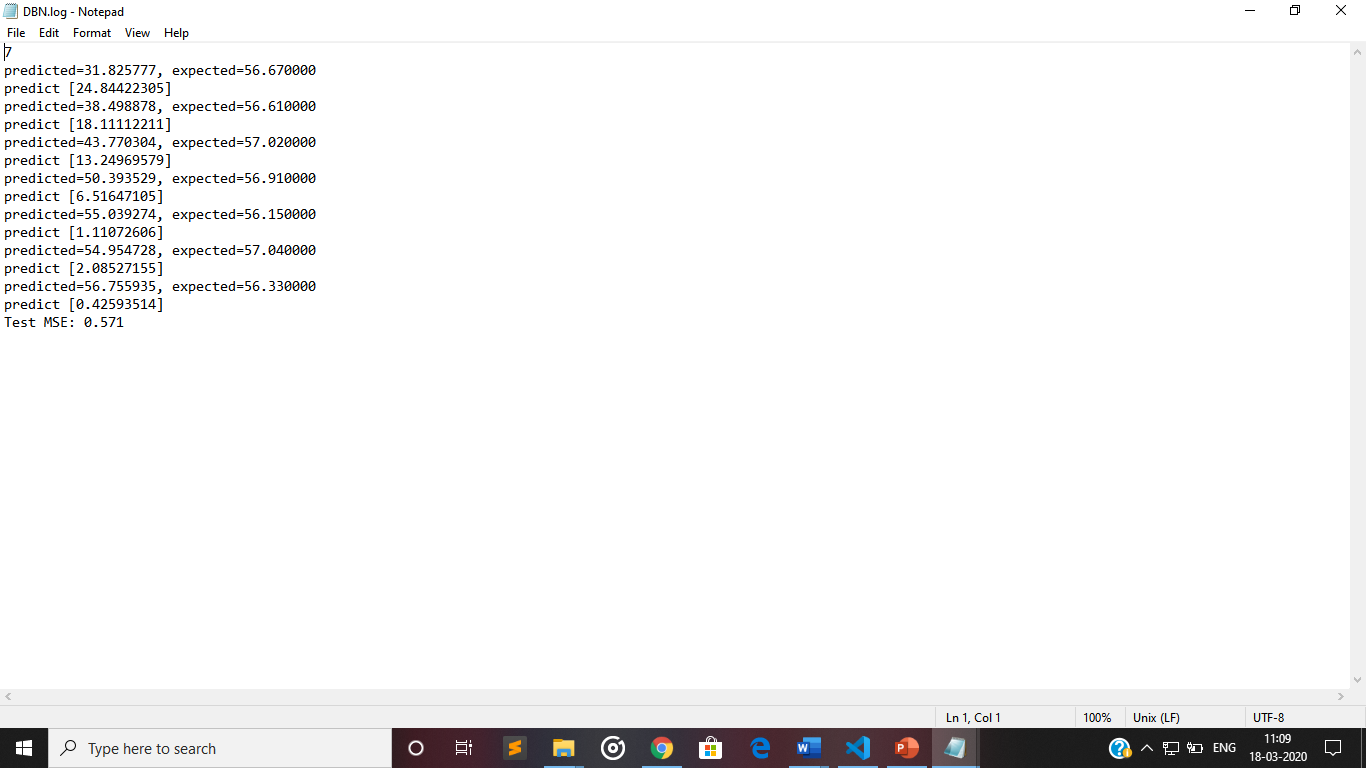
**Test Case 1: Prediction of crude oil price for the trained data set**



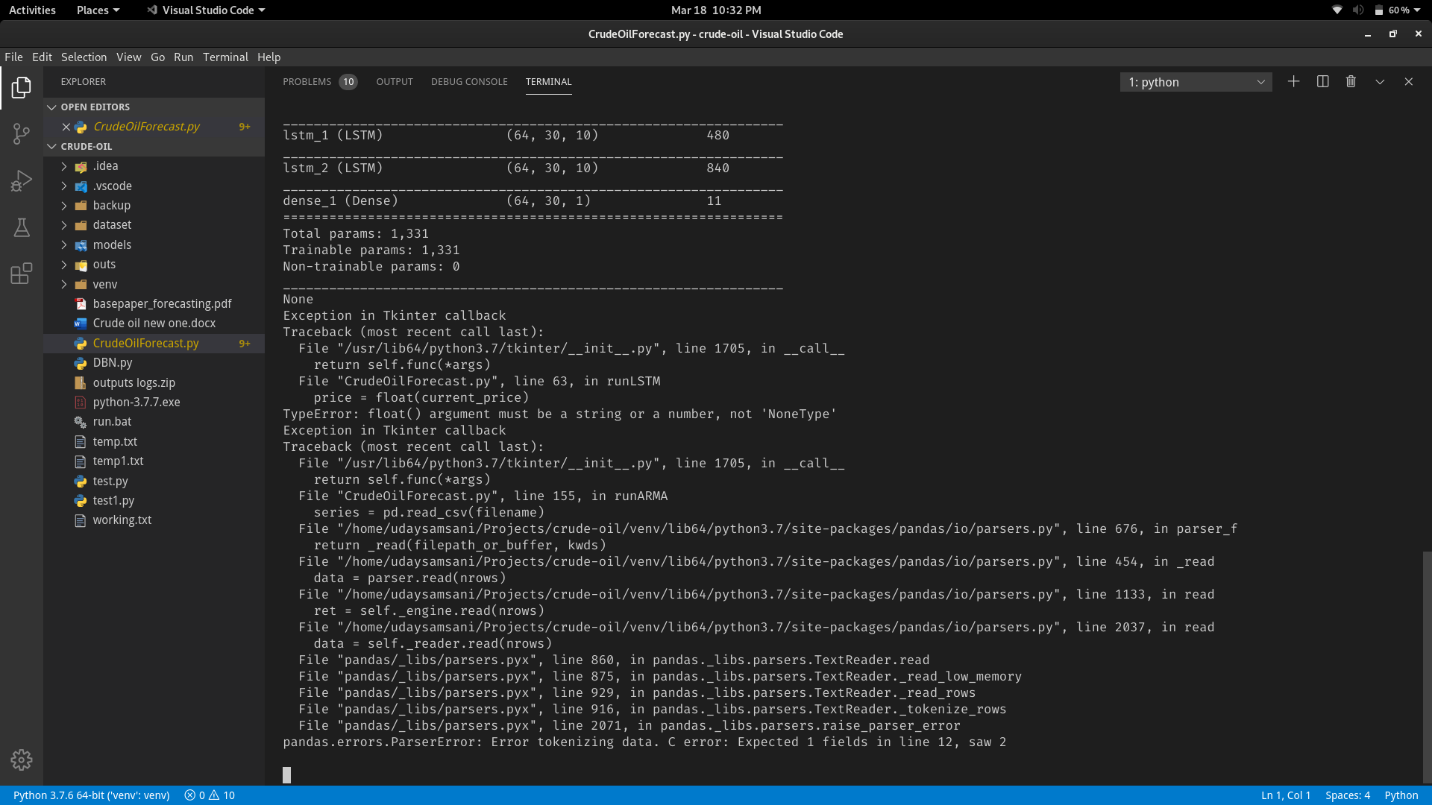
**Test Case 2 : Prediction unsuccessful if we give unrelated .csv file**



**Test Case 3: Successful prediction for the test set:**



**Test Case 4: Error in Command Prompt if we give other than .csv file**



**CONCLUSION**

**12.CONCLUSION**

Forecasting Crude Oil prices is a very challenging problem due to the high volatility of oil prices. In this paper, we developed a new oil price prediction approach using ideas and tools from stream learning, a machine learning paradigm for analysis and inference of continuous flow of non-stationary data. Our stream learning model will be updated whenever new oil price data are available, and provided to model, so the model continuously evolves over time, and can capture the changing pattern of oil prices. In addition, updating the model requires only a small constant time per new data example, The experiment results show that our stream learning model outperformed four other popular oil price prediction models over a variety of forecast time horizons. This process is used to Predict the oil Prices. The prediction model predicts continuous valued functions. To generalize the linear regression model, when dependant variable is categorical and analyses relationship between multiple independent variables.

**BIBLIOGRAPHY**

**13.BIBLIOGRAPHY**

1. F. Shen, J. Chao, J. Zhao, Forecasting exchange rate using deep belief networks and conjugate gradient method, Neurocomputing 167 (2015) 243 – 253
2. <https://scikit-learn.org/stable/auto_examples/svm/plot_svm_regression.html>
3. H. Y. Zeng, M. D. Edwards, G. Liu, D. K. Giﬀord, Convolutional neural network architectures for predicting dna-protein binding, Bioinformatics 32 (12) (2016) 121–127
4. <https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html>
5. S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Computation 9 (8) (1997) 1735.
6. Stock price prediction using machine learning and deep learning techniques by AISHWARYA SINGH
7. F. A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: Continual prediction with lstm. neural computation 12(10): 2451-2471, Neural Computation 12 (10) (2000) 2451–2471.
8. [www.indexwiz.com](http://www.indexwiz.com)
9. J. Wang, J. Wang, Forecasting energy market indices with recurrent neural networks: Case study of crude oil price ﬂuctuations, Energy 102 (2016) 365–374.
10. Z. H. Ling, L. Deng, D. Yu, Modeling spectral envelopes using restricted boltzmann machines and deep belief networks for statistical parametric speech synthesis, IEEE Transactions on Audio, Speech, and Language Processing 21 (10) (2013) 2129–2139.

11. R. Gupta, M. Wohar, Forecasting oil and stock returns with a qual var using over 150 years oﬀ data, Energy Economics 62 (2017) 181–186.

12.D. M. Zhu, W. K. Ching, R. J. Elliott, T. K. Siu, L. M. Zhang, Hidden markov models with threshold eﬀects and their applications to oil price forecasting, Journal of Industrial and Management Optimization 13 (2) (2017) 757–773.

13.H. Chiroma, S. Abdulkareem, T. Herawan, Evolutionary neural network model for west texas intermediate crude oil price prediction, Applied Energy 142 (2015) 266 – 273.

14.H. Chiroma, S. Abdul-kareem, A. S. M. Noor, A. I. Abubakar, N. S. Safa, L. Shuib, M. F. Hamza, A. Y. Gital, T. Herawan, A review on artiﬁcial intelligence methodologies for the

forecasting of crude oil price, Intelligent Automation and Soft Computing 22 (3) (2016) 449–462.

15.J. B. Heaton, N. G. Polson, J. H. Witte, Deep learning for ﬁnance: deep portfolios, Applied Stochastic Models in Business and Industry 33 (1) (2017) 3–12.

16.M. H. Raﬁei, H. Adeli, A novel machine learning model for estimation of sale prices of real estate units, Journal of Construction Engineering and Management 142 (2) (2016) 04015066. 17. A. Dedinec, S. Filiposka, A. Dedinec, L. Kocarev, Deep belief network based electricity load forecasting: An analysis of macedonian case, Energy 115, Part 3 (2016) 1688 – 1700, sustainable Development of Energy, Water and Environment Systems.

18.G. E. Hinton, R. R. Salakhutdinov, Reducing the dimensionality of data with neural networks, Science 313 (5786) (2006) 504.

19. H. Lee, R. Grosse, R. Ranganath, A. Y. Ng, Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations, in: International Conference on Machine Learning, 2009, pp. 609 – 616.

20.G. E. Hinton, S. Osindero, Y. W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (7) (2014) 1527–1554.

21.I. Arel, D. C. Rose, T. P. Karnowski, Deep machine learning - a new frontier in artiﬁcial intelligence research [research frontier], IEEE Computational Intelligence Magazine 5 (4) (2010) 13–18.

22. Y. Bengio, Learning Deep Architectures for AI, Now Publishers, 2009.

**APPENDIX**

**14.CODE**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

import numpy as np

from tkinter.filedialog import askopenfilename

import pandas as pd

import os

from keras.preprocessing import sequence

from keras.models import load\_model

from sklearn.preprocessing import MinMaxScaler

import numpy as np

from keras.layers import Dense

from keras.layers import Input, LSTM

from keras.models import Model

import h5py

import tensorflow as tf

from statsmodels.tsa.arima\_model import ARIMA

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

main = tkinter.Tk()

main.title("Crude Oil Price Forecasting") #designing main screen

main.geometry("1300x1200")

global filename

batch\_size = 64

epochs = 120

timesteps = 30

global lstm\_accuracy

global arma\_accuracy

def upload(): #function to upload tweeter profile

global filename

filename = filedialog.askopenfilename(initialdir="dataset")

text.delete('1.0', END)

text.insert(END,filename+" loaded\n");

def get\_train\_length(dataset, batch\_size, test\_percent):

# substract test\_percent to be excluded from training, reserved for testset

length = len(dataset)

length \*= 1 - test\_percent

train\_length\_values = []

for x in range(int(length) - 100,int(length)):

modulo=x%batch\_size

if (modulo == 0):

train\_length\_values.append(x)

print("x "+str(x))

return (max(train\_length\_values))

def runLSTM():

global lstm\_accuracy

current\_price = simpledialog.askstring(title="Stock Price",prompt="Enter Current Crude Oil Price")

text.delete('1.0', END)

text.insert(END,"LSTM Predicted Prices loaded\n\n");

if os.path.exists('models/model.h5'):

model = tf.keras.models.load\_model('models/model.h5')

print(model.summary())

strs = 'Date,Value\n'

price = float(current\_price)

price1 = float(current\_price)

for i in range(0,760):

if i < 550:

strs=strs+'1994-10-24,'+str(price)+'\n'

price = price - 0.005

if i > 550 and i < 760:

strs=strs+'1994-10-24,'+str(price1)+'\n'

price1 = price1 + 0.023

file = open('temp1.txt','w')

file.write(strs)

file.close()

df\_data\_1 = pd.read\_csv('temp1.txt')

df\_data\_1\_test = df\_data\_1[0:734]

test\_set = df\_data\_1\_test.iloc[:,1:2].values

sc = MinMaxScaler(feature\_range = (0, 1))

scaled\_real\_bcg\_values\_test = sc.fit\_transform(np.float64(test\_set))

X\_test = []

for i in range(timesteps, 734):

X\_test.append(scaled\_real\_bcg\_values\_test[i-timesteps:i, 0])

X\_test = np.array(X\_test)

X\_test = np.reshape(X\_test, (X\_test.shape[0], X\_test.shape[1], 1))

predicted\_bcg\_values\_test\_mae = model.predict(X\_test, batch\_size=batch\_size)

scores = model.evaluate([X\_test,X\_test], X\_test, verbose=2)

lstm\_accuracy = 100 - scores

model.reset\_states()

predicted\_bcg\_values\_test\_mae = np.reshape(predicted\_bcg\_values\_test\_mae, (predicted\_bcg\_values\_test\_mae.shape[0], predicted\_bcg\_values\_test\_mae.shape[1]))

predicted\_bcg\_values\_test\_mae = sc.inverse\_transform(predicted\_bcg\_values\_test\_mae)

y\_test = []

for j in range(0, 704 - timesteps):

y\_test = np.append(y\_test, predicted\_bcg\_values\_test\_mae[j, timesteps-1])

y\_test = np.reshape(y\_test, (y\_test.shape[0], 1))

index = 0

cp = float(current\_price)

for i in range(0,len(y\_test)):

if y\_test[i-timesteps] < cp:

index = index + 1

text.insert(END,"Current Price : "+current\_price+" Predicted Price : "+str(y\_test[i-timesteps])+"\n")

if index > 5:

break;

index = 0

for i in range(0,len(y\_test)):

if y\_test[i-timesteps] > cp:

index = index + 1

text.insert(END,"Current Price : "+current\_price+" Predicted Price : "+str(y\_test[i-timesteps])+"\n")

if index > 5:

break;

text.insert(END,"\n\nLSTM Accuracy : "+str(lstm\_accuracy))

else:

df\_data\_1 = pd.read\_csv(filename)

length = len(df\_data\_1)

length \*= 1 - 0.1

length = get\_train\_length(df\_data\_1, batch\_size, 0.1)

upper\_train = length + timesteps\*2

df\_data\_1\_train = df\_data\_1[0:upper\_train]

training\_set = df\_data\_1\_train.iloc[:,1:2].values

sc = MinMaxScaler(feature\_range = (0, 1))

training\_set\_scaled = sc.fit\_transform(np.float64(training\_set))

X\_train = []

y\_train = []

for i in range(timesteps, length + timesteps):

X\_train.append(training\_set\_scaled[i-timesteps:i,0])

y\_train.append(training\_set\_scaled[i:i+timesteps,0])

X\_train, y\_train = np.array(X\_train), np.array(y\_train)

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

y\_train = np.reshape(y\_train, (y\_train.shape[0], y\_train.shape[1], 1))

inputs\_1\_mae = Input(batch\_shape=(batch\_size,timesteps,1))

lstm\_1\_mae = LSTM(10, stateful=True, return\_sequences=True)(inputs\_1\_mae)

lstm\_2\_mae = LSTM(10, stateful=True, return\_sequences=True)(lstm\_1\_mae)

output\_1\_mae = Dense(units = 1)(lstm\_2\_mae)

regressor\_mae = Model(inputs=inputs\_1\_mae, outputs = output\_1\_mae)

regressor\_mae.compile(optimizer='adam', loss = 'mae')

regressor\_mae.summary()

parameters = 4 \* 10 \* (1 + 10 + 1)

parameters = 4 \* 10 \* (10 + 10 + 1)

for i in range(epochs):

print("Epoch: " + str(i))

regressor\_mae.fit(X\_train, y\_train, shuffle=False, epochs = 1, batch\_size = batch\_size)

regressor\_mae.reset\_states()

regressor\_mae.save(filepath="models/mymodel.h5")

def runARMA():

global arma\_accuracy

current\_price = simpledialog.askstring(title="Stock Price",prompt="Enter Current Crude Oil Price")

text.delete('1.0', END)

text.insert(END,"ARMA Predicted Prices loaded\n\n");

series = pd.read\_csv(filename)

strs = 'Date,Value\n'

price = float(current\_price)

for i in range(0,10):

strs=strs+'1994-10-24,'+str(price)+'\n'

price = price + 0.005

file = open('temp1.txt','w')

file.write(strs)

file.close()

series1 = pd.read\_csv("temp1.txt")

X = np.array(series["Value"])

X = np.reshape(X,(-1,1))

Y = np.array(series["Value"])

tt = np.array(series1["Value"])

tt = np.reshape(tt,(-1,1))

train, test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.2, random\_state=0)

predictions = []

history = [x for x in train]

correct = 0;

for i in range(len(tt)):

model = ARIMA(history, order=(5,1,0))

model\_fit = model.fit(disp=0)

output = model\_fit.forecast()

yhat = output[0]

predictions.append(yhat)

obs = tt[i]

history.append(obs)

predict = abs(yhat - obs)

print('predicted=%f, expected=%f' % (yhat, float(current\_price)))

text.insert(END,"Current Price : "+current\_price+" Predicted Price : "+str(yhat)+"\n")

if predict < 10:

correct = correct + 1

arma\_accuracy = (correct/len(tt))\*100

text.insert(END,"\n\nARMA Accuracy : "+str(arma\_accuracy))

def graph():

height = [lstm\_accuracy,arma\_accuracy]

bars = ('LSTM Model Accuracy','ARMA Model Accuracy')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

def exit():

global main

main.destroy()

font = ('times', 16, 'bold')

title = Label(main, text='Forecasting Crude Oil Prices: a Deep Learning based Model')

title.config(bg='greenyellow', fg='dodger blue')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=50,y=120)

text.config(font=font1)

font1 = ('times', 14, 'bold')

uploadButton = Button(main, text="Upload Crude Oil Dataset", command=upload)

uploadButton.place(x=50,y=550)

uploadButton.config(font=font1)

lstmButton = Button(main, text="Run Deep Learning LSTM Model & Forecast Price", command=runLSTM)

lstmButton.place(x=330,y=550)

lstmButton.config(font=font1)

armaButton = Button(main, text="Run Deep Learning ARMA Model & Forecast Price", command=runARMA)

armaButton.place(x=800,y=550)

armaButton.config(font=font1)

graphButton = Button(main, text="Accuracy Graph", command=graph)

graphButton.place(x=50,y=600)

graphButton.config(font=font1)

exitButton = Button(main, text="Exit", command=exit)

exitButton.place(x=330,y=600)

exitButton.config(font=font1)

main.config(bg='LightSkyBlue')

main.mainloop()