**House Price Prediction using Machine Learning Algorithm**

**ABSTRACT:**

The "House Price Prediction using Machine Learning" project presents a comprehensive approach to predicting real estate prices by harnessing the power of advanced data analysis techniques. Developed primarily using Python programming language, the project employs the Random Forest Regressor algorithm as its core predictive model. The objective is to accurately estimate the prices of residential properties, contributing to informed decision-making in the real estate market. In this project, a dataset containing 42,703 individual data points from the United States of America is utilized for training and evaluation. The dataset encompasses various essential features that influence property prices, including location, square footage, number of bedrooms and bathrooms, amenities, and more. By leveraging this diverse set of attributes, the Random Forest Regressor algorithm learns intricate patterns and relationships within the data, enabling it to make reliable predictions. The project's success is measured by the achieved performance metrics. During the training phase, the model attains a Mean Absolute Error (MAE) of 1.4606, indicating the average absolute difference between predicted and actual prices on the training set. Furthermore, on the test set, the model demonstrates its generalization capability by achieving a MAE of 3.8313. These metrics underscore the model's ability to make accurate predictions on unseen data, enhancing its practical utility in real-world scenarios. The Proposed House Price Prediction using Machine Learning showcases the efficacy of the Random Forest Regressor algorithm in forecasting residential property prices. The Python-based implementation leverages a dataset comprising thousands of data points from the United States, contributing to a robust and reliable predictive model. The achieved low Mean Absolute Error values on both training and test sets emphasize the model's accuracy and generalization potential. This project holds significant implications for individuals, investors, and real estate professionals seeking data-driven insights to navigate the dynamic real estate market.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  | **ABSTRACT**  **LIST OF FIGURES**  **LIST OF SYMBOLS** | i  v  vii |
| 1. | **CHAPTER 1 : INTRODUCTION**   * 1. GENERAL   2. SCOPE OF THE PROJECT   3. OBJECTIVE   1.4 EXISTING SYSTEM  1.4.1 EXISTINGSYSTEM DISADVANTAGES  1.5 LITERATURE SURVEY  1.6 PROPOSED SYSTEM  1.6.1 PROPOSED SYSTEM ADVANTAGES |  |
| 2. | **CHAPTER 2 :PROJECT DESCRIPTION**  2.1 GENERAL  2.2 METHODOLOGIES  2.2.1 MODULES NAME  2.2.2 MODULES EXPLANATION  2.3 TECHNIQUE OR ALGORITHM |  |
| 3. | **CHAPTER 3 : REQUIREMENTS**  3.1 General  3.2 Hardware REQUIREMENTS  3.3 Software REQUIREMENTS |  |
| 4. | **CHAPTER 4 :SYSTEM DESIGN**  **4.1 general**  **4.2 uml diagrams**  4.2.1 USE CASE DIAGRAM  4.2.2 CLASS DIAGRAM  4.2.3 OBJECT DIAGRAM  4.2.4 STATE DIAGRAM  4.2.5 activity diagram  4.2.6 SEQUENCE DIAGRAM  4.2.7 COLLABORATION DIAGRAM  4.2.8 COMPONENT DIAGRAM  4.2.9 DATA FLOW DIAGRAM  4.2.10 DEPLOYMENT DIAGRAM  4.2.11 SYSTEM ARCHITECTURE |  |
| 5. | **CHAPTER 5 : DEVELOPMENT TOOLS**  5.1 general  5.2 History of Python  5.3 Importance of Python  5.4 Features of Python  5.5 Libraries used in python |  |
| 6. | **CHAPTER 6 :IMPLEMENTATION**  6.1 GENERAL  6.2 IMPLEMENTATION |  |

|  |  |  |
| --- | --- | --- |
| 7. | **CHAPTER 7 :SNAPSHOTS**  7.1 GENERAL  7.2 VARIOUS SNAPSHOTS |  |
| 8. | **CHAPTER 8 :SOFTWARE TESTING**  8.1 GENERAL  8.2 DEVELOPING METHODOLOGIES  8.3 TYPES OF TESTING |  |
| 9. | **CHAPTER 9 :**  **FUTURE ENHANCEMENT**  9.1 FUTURE ENHANCEMENTS |  |
| **10** | **CHAPTER 10 :**  10.1CONCLUSION  10.2 REFERENCES |  |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
|  |  |  |
| 4.1 | Use case Diagram |  |
| 4.2 | Class diagram |  |
| 4.3 | Object diagram |  |
| 4.4 | State Diagram |  |
| 4.5 | Activity Diagram |  |
| 4.6 | Sequence diagram |  |
| 4.7 | Collaboration diagram |  |
| 4.8 | Component Diagram |  |
| 4.9 | Data flow diagram |  |
| 4.10 | Deployment Diagram |  |
| 4.11 | Architecture Diagram |  |

**LIST OF SYSMBOLS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **NOTATION**  **NAME** | **NOTATION** | **DESCRIPTION** |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | Represents a collection of similar entities grouped together. |
| 2. | Association | name  Class B  Class A  Class A  Class B | Associations represents static relationships between classes. Roles represents the way the two classes see each other. |
| 3. | Actor | Class A  Class A  Class B  Class B | It aggregates several classes into a single classes. |
| 4. | Aggregation | Interaction between the system and external environment |

|  |  |  |  |
| --- | --- | --- | --- |
| 5. | Relation  (uses) | uses | Used for additional process communication. |
| 6. | Relation  (extends) | extends | Extends relationship is used when one use case is similar to another use case but does a bit more. |
| 7. | Communication |  | Communication between various use cases. |
| 8. | State | State | State of the processes. |
| 9. | Initial State |  | Initial state of the object |
| 10. | Final state |  | Final state of the object |
| 11. | Control flow |  | Represents various control flow between the states. |
| 12. | Decision box |  | Represents decision making process from a constraint |
| 13. | Use case |  | Interact ion between the system and external environment. |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which are a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard, sensors, etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**CHAPTER-1**

**INTRODUCTION**

1

1.Introduction :

Predicting house prices are one of the important factors for the non-house holders, as they need to plan their budget [1]. The system has been proposed with various algorithms and the algorithm that gives the most correct accurate level is considered [2]. Houses are the central need for people and the prices of the houses vary from one place to another. House price prediction is a method that is used by any kind of people as no one can estimate the houses based on the location or the facilities in the area [3]. The house prices detection is a major and difficult task. The proposed model would make accurate predictions of the houses.

Machine learning algorithm uses historical data to predict future output values [4]. Machine Learning is an approach to Artificial Intelligence (AI) that enables software applications to become more accurate at predicting outcomes [5-6]. Machine Learning is a term coined by Arthur Samuel, a computer scientist at IBM and a pioneer in AI and gaming. Machine Learning is a subset of AI. A subset of Machine Learning is mostly related to computational statistics, which gives predictions using computers [7-8]. Data mining deals with exploratory learning through unsupervised learning [9-10]. Python is a high-level programming language. It is mostly used in designing websites, tasks, supports automated tasks and helps in conducting data analysis [11]. It was developed by Guido van Rossum. It is a very easy and understandable programming language. Python supports OOPS concept. They are portable and extendable. Python library contains various modules. Some of the modules are matplotlib, pandas, NumPy and etc. The most commonly used python library for Machine Learning is scikit-learn for preprocessing data and they are open-source [12]. They are highly interactive and can be integrated with other programming languages. Python code is interpreted at run time; they no need to be compiled before executing.

Regression techniques are a subpart of supervised Machine Learning. They are developed by using the connection between an explained variable and a set of exposure variables. Regression is one of the prediction algorithms that predict the outcome based on a value. The main aim of the regression algorithm is to build an equation that defines y as a function of x variables. Linear regressions one of the basic regression techniques used for the prediction and analysis of a data set in Machine Learning. The simple formula for linear regression is y=m\*x+c. Here various regression algorithms are used. Some are Gradient Boosting Regressor, Hist Gradient Boosting Regressor, and Random Forest Regressor are used in finding the predictions of accuracy for the houses.

**1.2 SCOPE OF THE PROJECT**

The scope of the project is to create a comprehensive system capable of predicting property prices based on a multitude of features that influence real estate valuations. These features encompass a wide range, including property size, location, amenities, bedrooms, bathrooms, and more. Through careful data collection, preprocessing, and feature engineering, the project strives to transform raw property data into meaningful input for the predictive model.

**1.3 OBJECTIVE**

The primary objective of the project is to build a predictive model that accurately estimates property prices. The project aims to achieve this by:

* Developing a robust and accurate Random Forest Regressor-based predictive model.
* Ensuring the model's ability to generalize well to new and unseen property data.
* Providing an intuitive user interface that allows users to input property features and obtain estimated prices.
  1. **EXISTING SYSTEM:**
* Gradient Boosting Regressor is a machine learning algorithm that falls under the ensemble learning category. It builds a predictive model in the form of an ensemble of weak learners, typically decision trees, to make accurate predictions.
* The idea behind gradient boosting is to sequentially add new models to correct the errors of the existing ensemble. The algorithm minimizes a loss function, such as mean squared error for regression problems, by iteratively fitting new models to the residuals.
  + 1. **EXISTINGSYSTEM DISADVANTAGES:**
* Gradient Boosting Regressors, especially when using a large number of trees or deep trees, can be computationally expensive and time-consuming.
* Gradient Boosting models, if not properly tuned, are susceptible to overfitting.
* Gradient Boosting Regressors have several hyper parameters that need to be tuned for optimal performance.

**1.5 LITERATURE SURVEY**

**Title:** Time-Aware Latent Hierarchical Model for Predicting House Prices

**Author:** Tan F, Cheng C, Wei Z

**Year:** 2017.

**Description:** It is widely acknowledged that the value of a house is the mixture of a large number of characteristics. House price prediction thus presents a unique set of challenges in practice. While a large body of works are dedicated to this task, their performance and applications have been limited by the shortage of long time span of transaction data, the absence of real-world settings and the insufficiency of housing features. To this end, a time-aware latent hierarchical model is introduced to capture underlying spatiotemporal interactions behind the evolution of house prices. The hierarchical perspective obviates the need for historical transaction data of exactly same houses when temporal effects are considered. The proposed framework is examined on a large-scale dataset of the property transaction in Beijing. The whole experimental procedure strictly complies with the real-world scenario. The empirical evaluation results demonstrate the outperformance of our approach over alternative competitive methods.

**Title:** Predicting the housing price direction using machine learning techniques

**Author:** D. Banerjee and S. Dutta

**Year:** 2017.

**Description**: The phenomenon of the falling or rising of the house prices has attracted interest from the researcher as well as many other interested parties. There have been many previous research works that used various regression techniques to address the question of the changes house price. This work considers the issue of changing house price as a classification problem and applies machine learning techniques to predict whether house prices will rise or fall. This work applies various feature selection techniques such as variance influence factor, Information value, principle component analysis and data transformation techniques such as outlier and missing value treatment as well as box-cox transformation techniques. The performance of the machine learning techniques is measured by the four parameters of accuracy, precision, specificity and sensitivity. The work considers two discrete values 0 and 1 as respective classes. If the value of the class is 0 then we consider that the price of the house has decreased and if the value of the class is 1 then we consider that the price of the house has increased.

**Title:** Multivariate regression modeling for home value estimates with evaluation using maximum information coefficient.

**Author:** G. Hu, J. Wang and W. Feng

**Year:** 2013.

**Description:** Predictive modeling is a statistical data mining approach that builds a prediction function from the observed data. The function is then used to estimate a value of a dependent variable for new data. A commonly used predictive modeling method is regression that has been applied to a wide range of application domains. In this paper, we build multivariate regression models of home prices using a dataset composed of 81 homes. We then applied the maximum information coefficient (MIC) statistics to the observed home values (Y) and the predicted values (X) as an evaluation of the regression models. The results showed very high strength of the relationship between the two variables X and Y.

**Title:**  Twitter Sentiment Analysis Using Naive Bayes based Machine learning Technique

**Author:** Priya Gour, Sudhanshu Vashistha and Pradeep Jha

**Year:** 2023

**Description**: “Computational” sentiment analysis can determine whether a sentiment is favorable, negative, or neutral. Another term for this approach is “opinion mining,” or obtaining a speaker’s sentiments. Businesses use it to develop strategies, learn what customers think about products or brands, how people react to campaigns or new product releases, and why they do not buy certain products. It is used in politics to keep track of political ideas and to check for contradictions between government claims and actions. It can even be used to predict election results! It is also used to track and analyze social phenomena like recognizing dangerous circumstances and evaluating blogging mood. In this paper, we look tackle the problem of sentiment categorization using the Twitter dataset. To analyze sentiment, preprocessing and Naive Bayes classifier approaches are utilized. As a result, we applied a text preprocessing classification accuracy classifying strategy and improved our classification accuracy score on the Kaggle public leaderboard. The aim of this paper is to classify the twitter sentiments using machine learning algorithm based on Naïve Bayes Classifier. The proposed model indicated better accuracy and precision based on performance parameters such as precision, recall and accuracy.

**Title:** Predicting House Price with a Memristor-Based Artificial Neural Network

**Author**: J. J. Wang et al

**Year:** 2018**.**

**Description:** Synaptic memristor has attracted much attention for its potential applications in artificial neural networks (ANNs). However useful applications in real life with such memristor-based networks have seldom been reported. In this paper, an ANN based on memristors is designed to learn a multi-variable regression model with a back-propagation algorithm. A weight unit circuit based on memristor, which can be programed as an excitatory synapse or inhibitory synapse, is introduced. The weight of the electronic synapse is determined by the conductance of the memristor, and the current of the synapse follows the charge-dependent relationship. The ANN has the ability to learn from labeled samples and make predictions after online training. As an example, the ANN was used to learn a regression model of the house prices of several Boston towns in the USA and the predicted results are found to be close to the target data.

**1.6 PROPOSED SYSTEM**

* Random Forest is a robust ensemble learning algorithm that leverages the strength of multiple decision trees. By constructing a diverse set of trees through techniques like bagging and random feature selection, it achieves higher predictive accuracy and resilience compared to individual trees.
* Known for its versatility, Random Forest is applicable to both classification and regression tasks. Its ability to handle high-dimensional datasets, mitigate overfitting, and provide insights into feature importance makes it a popular and effective choice across various domains.

**1.6.1 PROPOSED SYSTEM ADVANTAGES:**

* Random Forests often provide high accuracy in classification tasks.
* Random Forests can handle large datasets with a large number of features.
* They are computationally efficient and can parallelize the training process, making them suitable for big data applications.

**CHAPTER 2**

**PROJECT DESCRIPTION**

**2.1 GENERAL:**

Accurate property price prediction is crucial for a myriad of real estate-related scenarios. Prospective buyers can benefit from estimated prices when making purchase decisions, sellers can set competitive listing prices, and real estate professionals can offer clients data-driven insights. Moreover, investors can use the predictions to identify potentially profitable opportunities.

The use of the Random Forest Regressor algorithm enhances the project's significance. This algorithm's ensemble nature reduces overfitting, handles outliers, and provides insights into feature importance, making it an ideal candidate for modeling the complexities of property price determinants.

**2.2 METHODOLOGIES**

**2.2.1 MODULES NAME:**

* Data Collection
* Dataset
* Data Preparation
* Splitting the dataset
* Model Selection
* Analyze and Prediction
* Accuracy on test set
* Saving the Trained Model

**2.2.2 MODULES EXPLANATION:**

**Data Collection:**

In the first module we develop the data collection process. This is the first real step towards the real development of a machine learning model, collecting data. This is a critical step that will cascade in how good the model will be, the more and better data that we get; the better our model will perform.

There are several techniques to collect the data, like web scraping, manual interventions. The dataset is referred from the popular dataset repository called kaggle. The following is the dataset link for the House Price Prediction using ML.

Kaggle Dataset Link:

<https://www.kaggle.com/datasets/jayaprakashpondy/us-house-price>

**Dataset:**

The dataset consists of 42703 individual data. There are 20 columns in the dataset, which are described below.

**0. year\_built -** Year in which the house was built

**1. stories -** No. of stories in the house

**2. num\_bedrooms -** No. of bedrooms in the house

**3. full\_bathrooms -** No. of bathrooms in the house

**4. half\_bathrooms -** No. of half bathrooms in the house

**5. livable\_sqft -** Livable sqft in the house

**6. total\_sqft -** Total sqft in the house

**7. garage\_type -** Garage types in the house

**8. garage\_sqft -** Garage sqft in the house

**9. carport\_sqft -** carport sqft in the house

**10. has\_fireplace -** True and False

**11. has\_pool -** True and False

**12. has\_central\_heating -** True and False

**13. has\_central\_cooling -** True and False

**14. house\_number -** House number

**15. street\_name -** Street name

**16. unit\_number –** Unit number for the house

**17. city –** City name

**18. zip\_code –** zip code

**19. sale\_price –** sale price

**Data Preparation:**

Wrangle data and prepare it for training. Clean that which may require it (remove duplicates, correct errors, deal with missing values, normalization, data type conversions, etc.)

Randomize data, which erases the effects of the particular order in which we collected and/or otherwise prepared our data

Visualize data to help detect relevant relationships between variables or class imbalances (bias alert!), or perform other exploratory analysis

Split into training and evaluation sets

**Splitting the dataset:**

Data Splitting and Validation is crucial for training and evaluating the model. This module divides the dataset into training, validation, and testing sets. It ensures that the model's performance is assessed accurately using proper validation techniques like cross-validation. Split the dataset into train and test. 80% train data and 20% test data.

**Model Selection:**

We used Random Forest Regressor machine learning algorithm, we got a Training Set Mean Absolute Error: 1.4606 and Test Set Mean Absolute Error: 3.74

**The Random Forests Algorithm**

Let’s understand the algorithm in layman’s terms. Suppose you want to go on a trip and you would like to travel to a place which you will enjoy.

So what do you do to find a place that you will like? You can search online, read reviews on travel blogs and portals, or you can also ask your friends.

Let’s suppose you have decided to ask your friends, and talked with them about their past travel experience to various places. You will get some recommendations from every friend. Now you have to make a list of those recommended places. Then, you ask them to vote (or select one best place for the trip) from the list of recommended places you made. The place with the highest number of votes will be your final choice for the trip.

In the above decision process, there are two parts. First, asking your friends about their individual travel experience and getting one recommendation out of multiple places they have visited. This part is like using the decision tree algorithm. Here, each friend makes a selection of the places he or she has visited so far.

The second part, after collecting all the recommendations, is the voting procedure for selecting the best place in the list of recommendations. This whole process of getting recommendations from friends and voting on them to find the best place is known as the random forests algorithm.

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

**How does the algorithm work?**

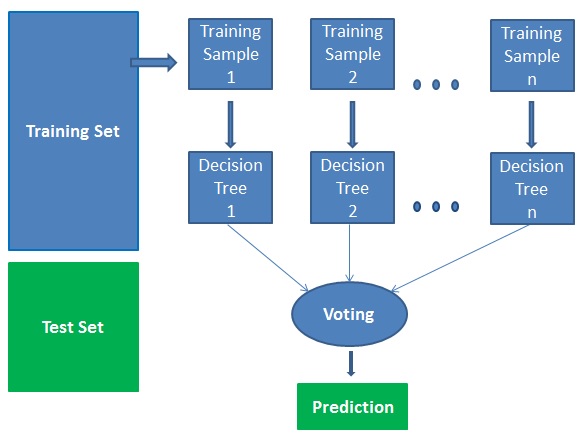
It works in four steps:

Select random samples from a given dataset.

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

Perform a vote for each predicted result.

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



**Finding important features**

Random forests also offers a good feature selection indicator. Scikit-learn provides an extra variable with the model, which shows the relative importance or contribution of each feature in the prediction. It automatically computes the relevance score of each feature in the training phase. Then it scales the relevance down so that the sum of all scores is 1.

This score will help you choose the most important features and drop the least important ones for model building.

Random forest uses gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

**Analyze and Prediction:**

In the actual dataset, we chose only 13 features :

**0. year\_built -** Year in which the house was built

**1. stories -** No. of stories in the house

**2. num\_bedrooms -** No. of bedrooms in the house

**3. full\_bathrooms -** No. of bathrooms in the house

**4. half\_bathrooms -** No. of half bathrooms in the house

**5. livable\_sqft -** Livable sqft in the house

**6. total\_sqft -** Total sqft in the house

**7. garage\_type -** Garage types in the house

**8. has\_fireplace -** True and False

**9. has\_pool -** True and False

**10. has\_central\_heating -** True and False

**11. has\_central\_cooling -** True and False

**12. city –** City name

**13. sale\_price –** sale price

# Accuracy on test set:

The Accuracy Evaluation Module is a pivotal component within the "House Price Prediction using Machine Learning" project. This module is dedicated to assessing and quantifying the accuracy of the house price prediction model developed in the project. Ensuring accurate predictions is essential for the reliability and utility of the system, as financial decisions heavily rely on the precision of these forecasts. We got a score 97.6% and Training Set Mean Absolute Error: 1.4606 and Test Set Mean Absolute Error: 3.74

**Saving the Trained Model:**

Once you’re confident enough to take your trained and tested model into the production-ready environment, the first step is to save it into a .h5 or .pkl file using a library like pickle.

Make sure you have pickle  installed in your environment.

Next, let’s import the module and dump the model into .pkl file.

**2.3 TECHNIQUE USED OR ALGORITHM USED**

**2.3.1 EXISTING TECHNIQUE: -**

* **Gradient Boosting Regressor:**
* Gradient Boosting Regressor is a boosting algorithm that builds an additive model in a forward stage-wise fashion. It combines the predictions from multiple weak learners to create a strong predictive model.
* The algorithm minimizes the loss function by adjusting the parameters of each weak learner during each iteration, emphasizing areas where the current model makes errors.

**2.3.2 PROPOSED TECHNIQUE USED OR ALGORITHM USED:**

* **Random Forest Regressor:**
* Random Forest is a machine learning algorithm that builds a multitude of decision trees during training, creating an ensemble model for classification or regression tasks.
* It operates by employing two key sources of randomness: bagging (bootstrap aggregating) and random feature selection.
* Bagging involves training each tree on a different subset of the training data, sampled with replacement, while random feature selection considers only a subset of features at each split in the decision trees.

.

**CHAPTER 3**

**REQUIREMENTS ENGINEERING**

**3.1 GENERAL**

These are the requirements for doing the project. Without using these tools & software’s we can’t do the project. So we have two requirements to do the project. They are

* Hardware Requirements.
* Software Requirements.

**3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 4GB DD RAM
* HARD DISK : 250 GB

**3.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* OPERATING SYSTEM : WINDOWS 7/8/10
* PLATFORM : SPYDER3
* PROGRAMMING LANGUAGE : PYTHON
* FRONT END : SPYDER3
* WEB FRAMEWORK : FLASK

**3.4 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, The project's user interface streamlined the process of obtaining property price predictions. Users were empowered to input property features through a user-friendly interface, and the system responded with estimated prices based on the learned patterns from the training data. This accessibility made the system applicable to a wide range of users, including buyers, sellers, and real estate professionals.

**3.5 NON-FUNCTIONAL REQUIREMENTS**

**The major non-functional Requirements of the system are as follows**

**Usability**

The system is designed with completely automated process hence there is no or less user intervention.

**Reliability**

The system is more reliable because of the qualities that are inherited from the chosen platform python. The code built by using python is more reliable.

**Performance**

This system is developing in the high level languages and using the advanced back-end technologies it will give response to the end user on client system with in very less time.

**Supportability**

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is built into the system.

**Implementation**

The system is implemented in web environment using Jupyter notebook software. The server is used as the intellignce server and windows 10 professional is used as the platform. Interface the user interface is based on Jupyter notebook provides server system.

**CHAPTER 4**

**DESIGN ENGINEERING**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering.

**4.2 UML DIAGRAMS**

**4.2.1 USE CASE DIAGRAM**



**EXPLANATION:**

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. The above diagram consists of user as actor. Each will play a certain role to achieve the concept.

**4.2.2 CLASS DIAGRAM**

****

**EXPLANATION**

In this class diagram represents how the classes with attributes and methods are linked together to perform the verification with security. From the above diagram shown the various classes involved in our project.

**4.2.3 OBJECT DIAGRAM**

****

**EXPLANATION:**

In the above digram tells about the flow of objects between the classes. It is a diagram that shows a complete or partial view of the structure of a modeled system. In this object diagram represents how the classes with attributes and methods are linked together to perform the verification with security.

**4.2.4 STATE DIAGRAM**



**EXPLANATION:**

State diagram are a loosely defined diagram to show workflows of stepwise activities and actions, with support for choice, iteration and concurrency. State diagrams require that the system described is composed of a finite number of states; sometimes, this is indeed the case, while at other times this is a reasonable abstraction. Many forms of state diagrams exist, which differ slightly and have different semantics.

**4.2.5 ACTIVITY DIAGRAM**



**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

**4.2.6 SEQUENCE DIAGRAM**

****

**EXPLANATION:**

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

**4.2.7 COLLABORATION DIAGRAM**



**EXPLANATION:**

A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML). The concept is more than a decade old although it has been refined as modeling paradigms have evolved.

**4.2.8 COMPONENT DIAGRAM**



**EXPLANATION**

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems. User gives main query and it converted into sub queries and sends through data dissemination to data aggregators. Results are to be showed to user by data aggregators. All boxes are components and arrow indicates dependencies.

**4.2.9 DATA FLOW DIAGRAM**

**Level 0**

Dataset

Data Collection

User

**Level 1:**

Input Dataset

Preprocessing

Training dataset

Random Forest Regressor

Prediction/Classification

Testing Data

Predicted Results: House sales price

Fig 4.9: Data Flow Diagrams

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often they are a preliminary step used to create an overview of the system which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of processes, or information about whether processes will operate in sequence or in parallel.

**4.2.10 DEPLOYMENT DIAGRAM**

****

**EXPLANATION:**

Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. It maps software pieces of a system to the device that are going to execute it.

**SYSTEM ARCHITECTURE:**

US Dataset

Random Forest Regressor

Predicted Results: House sales price

Graph

Fig 4.11: System Architecture

**CHAPTER 5**

**DEVELOPMENT TOOLS**

**5.1 Python**

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

## 5.2 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

#### 5.3 Importance of Python

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

#### 5.4 Features of Python

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* IT supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**5.5 Libraries used in python**

* numpy - mainly useful for its N-dimensional array objects.
* pandas - Python data analysis library, including structures such as dataframes.
* matplotlib - 2D plotting library producing publication quality figures.
* scikit-learn - the machine learning algorithms used for data analysis and data mining tasks.



Figure : NumPy, Pandas, Matplotlib, Scikit-learn

**CHAPTER 6**

**IMPLEMENTATION**

**6.1 GENERAL**

**Coding:**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.preprocessing import PolynomialFeatures

from sklearn.linear\_model import LinearRegression

import pandas as pd

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

from sklearn import ensemble

from sklearn.metrics import mean\_absolute\_error

ss=pd.read\_csv("ml\_house\_data\_set.csv")

ss

year\_built stories num\_bedrooms full\_bathrooms half\_bathrooms livable\_sqft total\_sqft garage\_type garage\_sqft carport\_sqft has\_fireplace has\_pool has\_central\_heating has\_central\_cooling house\_number street\_name unit\_number city zip\_code sale\_price

0 1978 1 4 1 1 1689 1859 attached 508 0 True False True True 42670 Lopez Crossing NaN Hallfort 10907 270897.0

1 1958 1 3 1 1 1984 2002 attached 462 0 True False True True 5194 Gardner Park NaN Hallfort 10907 302404.0

2 2002 1 3 2 0 1581 1578 none 0 625 False False True True 4366 Harding Islands NaN Lake Christinaport 11203 2519996.0

3 2004 1 4 2 0 1829 2277 attached 479 0 True False True True 3302 Michelle Highway NaN Lake Christinaport 11203 197193.0

4 2006 1 4 2 0 1580 1749 attached 430 0 True False True True 582 Jacob Cape NaN Lake Christinaport 11203 207897.0

... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ... ...

42698 1982 1 1 1 0 591 627 none 0 200 False False True True 562 Gregory Ford 253.0 Lake Jack 10825 88197.0

42699 1983 1 1 1 0 592 624 none 0 204 False False True True 62028 Gregory Ford 3560.0 Lake Jack 10825 102690.0

42700 1983 1 1 1 0 594 618 none 0 197 False False True True 62028 Gregory Ford 3931.0 Lake Jack 10825 98280.0

42701 1981 1 3 2 0 1398 1401 attached 401 0 False False True True 7456 Garcia View NaN Lake Jack 10825 98278.0

42702 1980 1 3 2 0 1401 1406 attached 396 0 False False True True 7456 Garcia View NaN Lake Jack 10825 186480.0

42703 rows × 20 columns

ss.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 42703 entries, 0 to 42702

Data columns (total 20 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 year\_built 42703 non-null int64

1 stories 42703 non-null int64

2 num\_bedrooms 42703 non-null int64

3 full\_bathrooms 42703 non-null int64

4 half\_bathrooms 42703 non-null int64

5 livable\_sqft 42703 non-null int64

6 total\_sqft 42703 non-null int64

7 garage\_type 42703 non-null object

8 garage\_sqft 42703 non-null int64

9 carport\_sqft 42703 non-null int64

10 has\_fireplace 42703 non-null bool

11 has\_pool 42703 non-null bool

12 has\_central\_heating 42703 non-null bool

13 has\_central\_cooling 42703 non-null bool

14 house\_number 42703 non-null int64

15 street\_name 42703 non-null object

16 unit\_number 3088 non-null float64

17 city 42703 non-null object

18 zip\_code 42703 non-null int64

19 sale\_price 42703 non-null float64

dtypes: bool(4), float64(2), int64(11), object(3)

memory usage: 5.4+ MB

import seaborn as sns

sns.distplot(ss["total\_sqft"])

ss['total\_sqft'].skew()

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

1.5300209687469986

sns.distplot(ss["livable\_sqft"])

ss['livable\_sqft'].skew()

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

1.2062188978583444

sns.distplot(ss["garage\_sqft"])

ss['garage\_sqft'].skew()

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

1.7262503236443785

sns.distplot(ss["carport\_sqft"])

ss['carport\_sqft'].skew()

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

9.970758215607686

sns.distplot(ss["sale\_price"])

ss['sale\_price'].skew()

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

12.065051245947682

value =ss[['year\_built','stories','num\_bedrooms','full\_bathrooms','half\_bathrooms','livable\_sqft','total\_sqft','garage\_sqft','carport\_sqft','sale\_price']]

f = pd.melt(ss, value\_vars=value)

g = sns.FacetGrid(f, col="variable", col\_wrap=3, sharex=False, sharey=False, height = 5)

g = g.map(sns.distplot, "value")

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

plt.rcParams["figure.figsize"] = (20,10)

corr = ss.corr()

sns.heatmap(corr, annot=True)

<AxesSubplot:>

ss.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 42703 entries, 0 to 42702

Data columns (total 20 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 year\_built 42703 non-null int64

1 stories 42703 non-null int64

2 num\_bedrooms 42703 non-null int64

3 full\_bathrooms 42703 non-null int64

4 half\_bathrooms 42703 non-null int64

5 livable\_sqft 42703 non-null int64

6 total\_sqft 42703 non-null int64

7 garage\_type 42703 non-null object

8 garage\_sqft 42703 non-null int64

9 carport\_sqft 42703 non-null int64

10 has\_fireplace 42703 non-null bool

11 has\_pool 42703 non-null bool

12 has\_central\_heating 42703 non-null bool

13 has\_central\_cooling 42703 non-null bool

14 house\_number 42703 non-null int64

15 street\_name 42703 non-null object

16 unit\_number 3088 non-null float64

17 city 42703 non-null object

18 zip\_code 42703 non-null int64

19 sale\_price 42703 non-null float64

dtypes: bool(4), float64(2), int64(11), object(3)

memory usage: 5.4+ MB

ss['sale\_price'].unique()

array([ 270897., 302404., 2519996., ..., 729544., 102690., 186480.])

sns.distplot(np.cbrt(ss['sale\_price']))

np.cbrt(ss['sale\_price']).skew()

D:\anaconda\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).

warnings.warn(msg, FutureWarning)

0.07824603024501398

output=np.cbrt(ss['sale\_price'])

ss['has\_fireplace'].unique()

array([ True, False])

ss.loc[ss['has\_fireplace']== True, 'has\_fireplace'] = 1

ss.loc[ss['has\_fireplace']== False, 'has\_fireplace'] = 0

ss['has\_fireplace'].unique()

array([1, 0], dtype=object)

ss.loc[ss['has\_pool']== True, 'has\_pool'] = 1

ss.loc[ss['has\_pool']== False, 'has\_pool'] = 0

ss.loc[ss['has\_central\_heating']== True, 'has\_central\_heating'] = 1

ss.loc[ss['has\_central\_heating']== False, 'has\_central\_heating'] = 0

ss.loc[ss['has\_central\_cooling']== True, 'has\_central\_cooling'] = 1

ss.loc[ss['has\_central\_cooling']== False, 'has\_central\_cooling'] = 0

ss['garage\_type'].unique()

array(['attached', 'none', 'detached'], dtype=object)

ss.loc[ss['garage\_type']== 'attached', 'garage\_type'] = 1

ss.loc[ss['garage\_type']== 'none', 'garage\_type'] = 0

ss.loc[ss['garage\_type']== 'detached', 'garage\_type'] = 2

ss['city'].unique()

array(['Hallfort', 'Lake Christinaport', 'Morrisport', 'West Ann',

'Richardport', 'Lewishaven', 'Clarkberg', 'Chadstad',

'South Stevenfurt', 'Coletown', 'South Anthony', 'Jenniferberg',

'North Erinville', 'Lake Jennifer', 'West Terrence',

'Jeffreyhaven', 'Brownport', 'Scottberg', 'Lake Carolyn',

'East Amychester', 'Joshuafurt', 'Davidtown',

'Port Jonathanborough', 'Lake Jack', 'Port Andrealand',

'Toddshire', 'Port Daniel', 'East Lucas', 'West Gregoryview',

'Lake Dariusborough', 'West Lydia', 'Davidfort', 'Wendybury',

'Amystad', 'Martinezfort', 'East Justin', 'West Brittanyview',

'New Michele', 'East Janiceville', 'Port Adamtown', 'Justinport',

'West Gerald', 'Fosterberg', 'Leahview', 'Rickytown', 'Julieberg',

'New Robinton'], dtype=object)

ss.loc[ss['city']== 'Hallfort', 'city'] = 0

ss.loc[ss['city']== 'Lake Christinaport', 'city'] = 1

ss.loc[ss['city']== 'Morrisport', 'city'] = 2

ss.loc[ss['city']== 'West Ann', 'city'] = 3

ss.loc[ss['city']== 'Richardport', 'city'] = 4

ss.loc[ss['city']== 'Lewishaven', 'city'] = 5

ss.loc[ss['city']== 'Clarkberg', 'city'] = 6

ss.loc[ss['city']== 'Chadstad', 'city'] = 7

ss.loc[ss['city']== 'South Stevenfurt', 'city'] = 8

ss.loc[ss['city']== 'Coletown', 'city'] = 9

ss.loc[ss['city']== 'South Anthony', 'city'] = 10

ss.loc[ss['city']== 'Jenniferberg', 'city'] = 11

ss.loc[ss['city']== 'North Erinville', 'city'] = 12

ss.loc[ss['city']== 'Lake Jennifer', 'city'] = 13

ss.loc[ss['city']== 'West Terrence', 'city'] = 14

ss.loc[ss['city']== 'Jeffreyhaven', 'city'] = 15

ss.loc[ss['city']== 'Brownport', 'city'] = 16

ss.loc[ss['city']== 'Scottberg', 'city'] = 17

ss.loc[ss['city']== 'Lake Carolyn', 'city'] = 18

ss.loc[ss['city']== 'East Amychester', 'city'] = 19

ss.loc[ss['city']== 'Joshuafurt', 'city'] = 20

ss.loc[ss['city']== 'Davidtown', 'city'] = 21

ss.loc[ss['city']== 'Port Jonathanborough', 'city'] = 22

ss.loc[ss['city']== 'Lake Jack', 'city'] = 23

ss.loc[ss['city']== 'Port Andrealand', 'city'] = 24

ss.loc[ss['city']== 'Toddshire', 'city'] = 25

ss.loc[ss['city']== 'Port Daniel', 'city'] = 26

ss.loc[ss['city']== 'East Lucas', 'city'] = 27

ss.loc[ss['city']== 'West Gregoryview', 'city'] = 28

ss.loc[ss['city']== 'Lake Dariusborough', 'city'] = 29

ss.loc[ss['city']== 'West Lydia', 'city'] = 30

ss.loc[ss['city']== 'Davidfort', 'city'] = 31

ss.loc[ss['city']== 'Wendybury', 'city'] =32

ss.loc[ss['city']== 'Amystad', 'city'] = 33

ss.loc[ss['city']== 'Martinezfort', 'city'] = 34

ss.loc[ss['city']== 'East Justin', 'city'] = 35

ss.loc[ss['city']== 'West Brittanyview', 'city'] = 36

ss.loc[ss['city']== 'New Michele', 'city'] = 37

ss.loc[ss['city']== 'East Janiceville', 'city'] = 38

ss.loc[ss['city']== 'Port Adamtown', 'city'] = 39

ss.loc[ss['city']== 'Justinport', 'city'] = 40

ss.loc[ss['city']== 'West Gerald', 'city'] = 41

ss.loc[ss['city']== 'Fosterberg', 'city'] = 42

ss.loc[ss['city']== 'Leahview', 'city'] = 43

ss.loc[ss['city']== 'Rickytown', 'city'] = 44

ss.loc[ss['city']== 'Julieberg', 'city'] = 45

ss.loc[ss['city']== 'New Robinton', 'city'] = 46

ss['city'].unique()

array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,

19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35,

36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46], dtype=object)

**CHAPTER 7**

**SNAPSHOTS**

**7.1 GENERAL:**

This project is implements like application using python and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.

**SNAPSHOTS**

**CHAPTER 8**

**SOFTWARE TESTING**

**8.1 GENERAL**

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.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3 Types of Tests**

**8.3.1 Unit testing**

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

**8.3.2 Functional test**

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

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

**8.3.3 System Test**

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

**8.3.4 Performance Test**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5 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.

**8.3.6 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.

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

**8.2.7 Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**CHAPTER 9**

**FUTURE ENHANCEMENT**

**9.1 FUTURE ENHANCEMENTS:**

In the future, comparisons can be made with different classification algorithms and can measure the performance of the system using various machine learning algorithms.

**CHAPTER 10**

**CONCLUSIONAND REFERENCES**

**10.1 CONCLUSION**

The house price prediction model using machine learning is proposed in this system using various regression and algorithmic techniques. The house prices in general deal with various factors like area, price, number of bedrooms, number of toilets, parking space and so on. This system helps people to choose houses based on their budget and market strategies, which does not affect their financial level.

In conclusion, the project "House Price Prediction using Random Forest Regressor" realized its goal of providing an accurate, efficient, and reliable tool for property price estimation. By harnessing the power of machine learning, the project's results hold significant implications for informed decision-making within the dynamic real estate market. The successful implementation of the Random Forest Regressor-based predictive model, coupled with meticulous data preprocessing and thoughtful feature engineering, underscores the system's potential to contribute to the advancement of real estate analytics.

**10.2 REFERENCES**

[1] Tan F, Cheng C, Wei Z., "Time-Aware Latent Hierarchical Model for Predicting House Prices", In2017 IEEE International Conference on Data Mining (ICDM), pp. 1111- 1116, 2017.

[2] D. Banerjee and S. Dutta, "Predicting the housing price direction using machine learning techniques", 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), pp. 2998-3000, 2017.

[3] G. Hu, J. Wang and W. Feng, "Multivariate regression modeling for home value estimates with evaluation using maximum information coefficient", Software Engineering Artificial Intelligence Networking and Parallel/Distributed Computing., vol. 1, no. 2, pp. 69-81, 2013.

[4] Priya Gour, Sudhanshu Vashistha and Pradeep Jha, “Twitter Sentiment Analysis Using Naive Bayes based Machine learning Technique", 2nd International Conference on Sentiment Analysis and Deep Learning (ICSADL 2022), Springer - Advances in Intelligent Systems and Comput ing Series 1432, 2023.

[5] J. J. Wang et al., "Predicting House Price With a Memristor-Based Artificial Neural Network," in IEEE Access, vol. 6, pp. 16523-16528, 2018.

[6] R. E. Febrita, A. N. Alfiyatin, H. Taufiq and W. F. Mahmudy, "Data driven fuzzy rule extraction for housing price predict ion in Malang, East Java", 2017 IEEE International Conference on Advanced Computer Science and Information Systems (ICACSIS), pp. 351-358, 2017.

[7] Himanshu Aora, Kiran Ahuja, Himanshu Sharma, Kartik Goyal and Gyanendra Kumar, "Artificial Intelligence and Machine Learning in Game Development", Turkish Online Journal of Qualitative Inquiry (TOJQI), vol. 12, no. 8, pp. 1153-1158, 2021.

[8] Kiran Ahuja, Harsh Sekhawat , Shilpi Mishra and Pradeep Jha, "Machine Learning in Artificial Intelligence: Towards a Common Understanding", Turkish Online Journal of Qualitative Inquiry (TOJQI), vol. 12, no. 8, pp. 1143-1152, July 2021.

[9] Rahul Misra and Dr. Ramkrishan Sahay, "A Review on Student Performance Predication Using Data Mining Approach", International Journal of Recent Research and Review, Vol. X, Issue 4, pp. 45-47, December 2017.

[10] R. Misra and Dr. R. Sahay, "Evaluat ion of Student Performance Prediction Models with Two Class Using Data Mining Approach", International Journal of Recent Research and Review, Vol. XI, Issue 1, pp. 71-79, March 2018.

[11] P. Jha, R. Baranwal, Monika and N. K. Tiwari, "Protection of User’s Data in IOT", 2022 IEEE Second International Conference on Artificial Intelligence and Smart Energy (ICAIS), pp. 1292-1297, 2022.

[12] P. Jha, T. Biswas, U. Sagar and K. Ahuja, "Prediction with ML paradigm in Healthcare System", 2021 IEEE Second International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 1334-1342, 2021.

[13] C. R. Madhuri, G. Anuradha and M. V. Pujitha, "House Price Prediction Using Regression Techniques: A Comparative Study," 2019 International Conference on Smart Structures and Systems (ICSSS), pp. 1-5, 2019.

[14] Y. Tang, S. Qiu and P. Gui, "Predicting Housing Price Based on Ensemble Learning Algorithm", 2018 IEEE International Conference on Artificial Intelligence and Data Processing (IDAP), pp. 1-5, 2018.

[15] P. Durganjali and M. V. Pujitha, "House Resale Price Prediction Using Classification Algorithms", 2019 IEEE International Conference on Smart Structures and Systems (ICSSS), pp. 1-4, 2019.

[16] M. Jain, H. Rajput , N. Garg and P. Chawla, "Prediction of House Pricing using Machine Learning with Python", 2020 IEEE International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 570-574, 2020.

[17] T. D. Phan, "Housing Price Predict ion Using Machine Learning Algorithms: The Case of Melbourne City, Australia", 2018 IEEE International Conference on Machine Learning and Data Engineering (iCMLDE), pp. 35-42, 2018.

[18] H. Arora, G. K. Soni, R. K. Kushwaha and P. Prasoon, "Digital Image Security Based on the Hybrid Model of Image Hiding and Encryption", 2021 IEEE 6th International Conference on Communication and Electronics Systems (ICCES), pp. 1153-1157, 2021.

[19] Himanshu Arora, Shilpi Mishra and Manish Dubey, "Development of the Framework for the Solution of the Security Problems in Data Transmission Involving Advanced Asymmetric Algorithm", International Journal of Emerging Technology and Advanced Engineering, vol. 8, no. 4, pp. 18-20, April 2018.

[20] D. Banerjee and S. Dutta, "Predicting the housing price direction using machine learning techniques," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), pp. 2998-3000, 2017.

[21] Y. Chen, R. Xue and Y. Zhang, "House price predict ion based on machine learning and deep learning methods," 2021 International Conference on Electronic Information Engineering and Computer Science (EIECS), pp. 699-702, 2021.