DEFECTIVE DATA CORRECTION (COLD START) OF DATA SET AND PREDICTION OF ADOBE PRICES BASED ON CLASSIFICATION AND REGRESSION PREDICTION

Lahari Reddy Muskula
dept. of Computer Science(Masters)
University of Central Missouri
700741333

Divya Sai Ajay Jasti
dept. of Computer Science(Masters)
University of Central Missouri
700741296

Vineel reddy
dept. of Computer Science(Masters)
University of Central Missouri
700741502

Maneesh Reddy Maram
dept. of Computer Science(Masters)
University of Central Missouri
700742207

***ABSTRACT*—Item cold-start problem is the problem that arises when there is lack of data, like missing values in the dataset, to start the analysis. These missing values are to be filled in with appropriate values, in order to start the analysis. This project identifies and replaces the missing values with the corresponding mean values in the dataset. This project analyses the updated dataset by using Gaussian Naive bayes analysis and decision tree analysis, for estimating house price based on area in square feet and number of bed rooms. This project is implemented through four modules: Data entry module, is used to provide the needed data to the project. The Analysis module is used to analyse and predict the house prices, based on the customer needs. The Front end module is used to create the needed GUI screens for the project. The prediction module is used to predict the price of a house with given parameters. Gaussian Naive Bayes is a variant of Naive Bayes. which is a classification algorithm of Machine Learning based on Bayes theorem which gives the likelihood of occurrence of the event. Gaussian Naive Bayes is Used when we are dealing with continuous data and uses Gaussian distribution, which is also known as normal distribution of a continuous variable, which is usually in the form of a bell shape. Decision tree builds classification or regression models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision node has two or more branches. Decision trees can handle both categorical and numerical data.**

KEYWORDS — Gaussian Naïve Bayes, Decision tree, Regression Predictor, algorithm

I. INTRODUCTION

Various facts were discovered as a result of the investigation, which led to strategic property investment methods. House cost forecasting is a well-known topic, and research teams are increasingly doing relevant studies using deep learning or AI models. However, because some studies haven't taken into account all of the factors that impact home prices, projection findings aren't always accurate. In this vein, we suggest a collaborative self-consideration model for housing expectations that covers the entire process from start to finish. In this model, we use satellite guides to analyse the climate around houses and incorporate information on open houses such as stops, schools, and mass rapid transit stations to address the accessibility of amenities.

We employ consideration components, which are often used in image, discourse, and interpretation assignments, to separate important features that prospective home buyers examine. As a result, the model can downgrade loads at the point when supplied exchange information is available. Our suggested model differs from self-consideration models in that it evaluates the relationship between two distinct provisions in order to comprehend the tangled relationship between highlights in order to improve prediction accuracy.

II. MOTIVATION

The purpose of this code is to construct a fully working application with a front-end and a back end for user interaction with the database and to deliver the most effective market strategy for housing prices to the user. To forecast the values of new house prices, we plan to employ Gaussian-Nave Bayes and Decision tree classifiers. We also plan to employ regression-based prediction to determine new housing prices and assess each system's accuracy. We considered the most critical attributes that stand as the deciding factor for the prediction of these house prices because the pricing of these houses is dependent on various factors such as (bathrooms, bedrooms, latitude, longitude, grade, year built, year of remodel, views, and other parameters). We think that this will be one of the strategies by which individuals gain insight into how to purchase or sell their homes, and that the economy of the country may be enhanced as a result of this strategic marketing strategy. Nobody would be able to fool people when it comes to house purchasing and selling. Our project aim is to clean the out-layer data from a genuine United States Housing dataset using valid Machine Learning methods, then apply the cleaned dataset to Regression tree classifiers, Decision tree classifiers, Gaussian Nave Bayes classifiers and Bagging regression, and Logistic Regression. These classifier findings are displayed to users so that they may use their own property information to forecast or determine the price of their home based on those features. The program should be able to store new houses while also predicting house values based on historical housing datasets. We want to employ a lightweight database for this so that storing vast volumes of data on less capable PCs, laptops, and workstations will be simple.

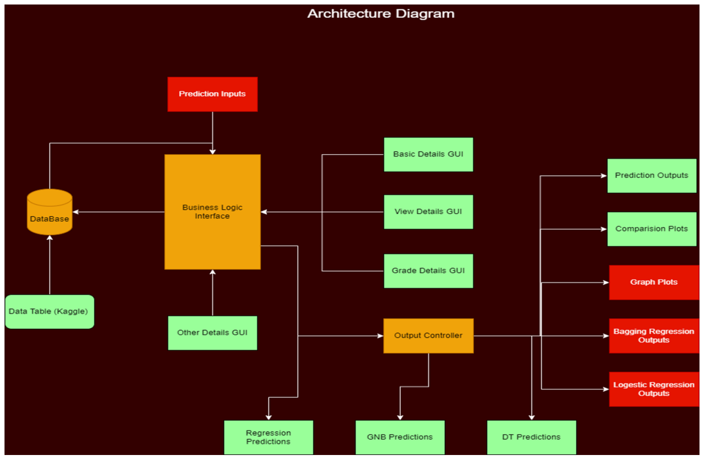
III. MAIN CONTRIBUTIONS & OBJECTIVES

The cold start problem refers to the presence of out-layer data in a dataset or database. It is vital to eliminate these out-layers; otherwise, the classifiers and regression models would produce erroneous results. We demonstrated two techniques to solve the cold start problem in this project. The fundamental concept is to take the mean of all the dimensions and use it to replace the out-layer data. There are multiple ways to remove the out-layers in the dataset such as: Univariate Method: The use of box plots is perhaps the simplest method for finding exceptions. A box plot is a graphical representation of how information is transmitted. The middle, lower, and higher quartiles are used in box plots. An exception, according to Tukey's method, is defined as the benefits of a variable that are far removed from the central issue. The cleaning border is the maximum distance from the information's focus point that will be allowed. When the cleaning boundary is broad, the test is less sensitive to exceptions. In reality, many attributes are labeled as exceptions if it is too minimal. Multivariate Method: Anomalies should not be exaggerated attributes. To be sure, as we've seen with objective B, the univariate approach doesn't always work well. The variable method attempts to address this by constructing a discriminating model using all of the data available and cleaning out those situations with errors beyond a certain value. For this case, we've created a neural network that incorporates all of the available data (however purpose A , avoided by the univariate strategy). Then we have a propensity to do a rapid relapse assessment in order to compile the subsequent chart. In contrast to the genuine features, the expected attributes are intended.

IV. RELATED WORK

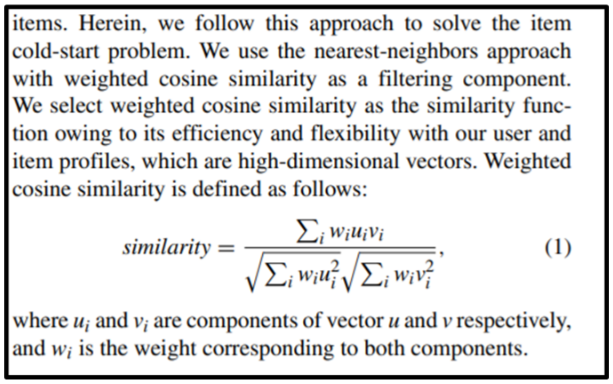
In this study, we use real estate information from a whole country. This is justified by the fact that users might be interested in various real estates from several distant locations. Badriyah proposed a property recommendation system based on content-based filtering and association rules. Their approach created user and item profiles from the collection of words in advertisements and performed term frequency-inverse document frequency (TF-IDF). Thereafter, it generated association rules using the user profiles as item sets via the appropriate algorithm and recommended property products based on these rules. Knoll et al. [16] conducted an experiment on extracted real estate website data for comparing a deep learning approach with the factorization machine. They have adapted neural collaborative filtering (NCF) to consider item features together with user and item identifiers.

Their results demonstrated that deep learning outperforms the factorization machine in both overall and cold-start results. Nevertheless, these methods omit sequential and context information and suffer from the item cold-start problem.



A deep learning approach has been applied to recommendation tasks for capturing sequential patterns.
It benefits from a nonlinear transformation, representation learning, and sequence modeling.

To solve the cold start problem, methods used: nearest neighbours using cosine similarity


items. Herein, we follow this approach to solve the item cold-start problem. We use the nearest-neighbors approach with weighted cosine similarity as a filtering component. We select weighted cosine similarity as the similarity function owing to its efficiency and flexibility with our user and item profiles, which are high-dimensional vectors. Weighted cosine similarity is defined as follows:

$$similarity = \frac{\sum_i w_i u_i v_i}{\sqrt{\sum_i w_i u_i^2}\sqrt{\sum_i w_i v_i^2}}, \quad (1)$$

where u_i and v_i are components of vector u and v respectively, and w_i is the weight corresponding to both components.

To solve the item cold-start problem, Wei proposed a hybrid recommendation model combining a time-aware model, timeSVD++ [4], with a deep learning architecture and a stacked denoising autoencoder (SDAE) for movie rating prediction. They used the descriptions of the cold-start item to predict its latent factor through the SDAE. Consequently, they found the top-n nearest warm-start items using Pearson's correlation coefficient and used the mean of their predicted ratings as the prediction.

Vartak introduced a meta-learning perspective for solving the item cold-start problem in the recommendation system using a cold-start item representation together with user representation from a learned history to predict the engagement between the cold-start item and the user. Furthermore, they proposed a linear and nonlinear classifier with weight and bias adaptations, respectively. These two classifiers were separately used for each user with different weights or biases depending on specific histories. Pan used meta-embedding to assess the item cold-start problem in click-through rate prediction to make the model better at dealing with the cold-start and faster at warming-up. When a new item was detected, they used its features to learn the representation using a meta-embedding generator designed to update the weight with future interactions until it became warm. These works demonstrated that using item attributes enable solving the item cold-start problem. However, sequential information was omitted and the proposed systems were reliant on the factorization machines, which are considered improper for real estate recommendations.

Content-based recommendation systems are another means of solving the item cold-start problem, and benefit from using only, the attributes that can work with any number of records. Setiadi recommended scientific articles through content-based filtering using two matching algorithms, i.e., k-means clustering and cosine similarity. They used TF-IDF to represent both user and item profiles, and elucidated the advantage of using k-means clustering over cosine similarity to obtain relevant items. Luostarinen and Kohonen used a form of topic modeling, latent dirichlet allocation (LDA), to represent and recommend news through the naive Bayes classifier, nearest neighbor regression, and linear regression with cosine similarity. Deldjoo integrated audio and visual descriptors to the hybrid recommendation system for solving the movie cold-start problem. They used metadata and extracted features to help recommend cold-start movies. The aforementioned works illustrated the advantages of content-based filtering for solving the item cold-start problem; however, they omitted sequential and context information, and their corresponding items do not include real estate.

Herein, we solve the item cold-start problem while using sequential and context information through a content-based approach by modifying a session-based recommendation system to be a profile learner. We also apply an attention mechanism and LC to efficiently deal with sequential and context information, respectively.

V. PROPOSED FRAMEWORK

In order to distinguish between healthy and diseased leaves from the generated data sets, we proposed the below algorithms for classification.

1. Gaussian Naïve Bayes Classifier:

The Bayes hypothesis underpins the Naive Bayes characterisation approach. It's a simple yet effective calculation for demonstrating foresight in controlled learning calculations. The Naive Bayes approach is simple to understand. When we have a large amount of data to focus on, Naive Bayes has a greater precision and speed.

There are three sorts of Naive Bayes models: Gaussian, Multinomial, and Bernoulli.
•**Gaussian Naive Bayes** – This is a variant of Naive Bayes that emphasizes consistent features and assumes that each class is transmitted normally.

•**Multinomial Naive Bayes** – Another variant is an occasion-based model with highlights as vectors, where sample(feature) refers to the frequency with which specific events have occurred.

•**Bernoulli** – This variety is also event-based, with components that are free Boolean and have a two-fold structure.

2. **Decision Tree Classifier:**
Decision Trees (DTs) are a non-parametrically applied learning tool for relapse and arrangement. With a lot of assuming else decision rules, decision trees gain from information to inexact a sine bend.
A decision tree is a tree like structure which assembles characterisation or relapse models. It divides an informative index into more modest and more modest subgroups while also progressively evolving an associated decision tree. The end result is a tree with leaf hubs and decision hubs. The root hub is the highest decision hub in a tree and is linked to the best indication. Decision trees can handle both unprocessed and mathematical data.

There are a few stages engaged with the structure of a decision tree:

• **Splitting**: The process of segmenting an informative collection into subgroups. Parts are framed around a certain variable.

• **Pruning**: Parts of the tree are being shortened. Pruning is the process in which the size of a tree is reduced by converting certain branch hubs to leaf hubs and then removing the leaf hubs beneath the initial branch. Pruning is beneficial because order trees may match the preparation information well.

• **Tree Selection**: The most frequent method for locating the smallest tree that matches the data. In general, this is the tree that produces the fewest cross-approved errors.

Entropy: A decision tree is built in a hierarchical fashion from a root hub, and it entails segmenting the data into subsets that comprise instances of varying quality (homogeneous).

Information Gain: The reduction in entropy when a dataset is partitioned on an attribute determines the information gain. Creating a decision tree is linked to identifying the feature that provides the most significant information gain (i.e., the most homogeneous branches). (It references to the cycle in which models are built around the preparation of information so thoroughly that any hiccup in testing information might have negative effects for the model's execution.)

3. **Regression Predictor:**
Regression in Data Mining:
Regression is one of the information mining techniques that is most commonly used to predict a set of consistent features (sometimes known as "numerical attributes") in a dataset. Regression, for example, may forecast agreements, benefits, temperature, distance, and so on...

Uses of Regression:
Regression is used widely in different organisations and businesses. It is also quite well- known. The following are a few examples of Regression's applications: The indicator variable (the attributes that clients identify) and the response variable are frequently included in the Regression process (the qualities that are to be anticipated).

Kinds of Regression in Data Mining:
Two kinds of Regression can be seen in information mining:

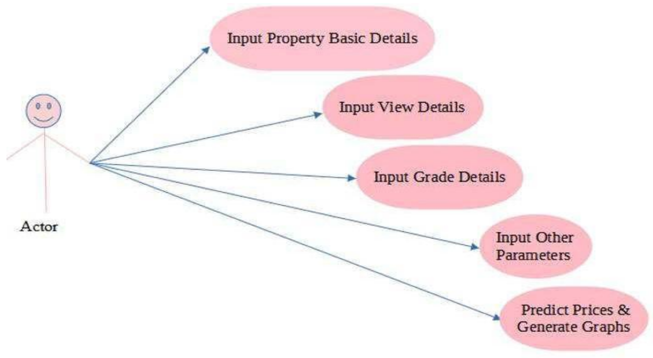
• Linear Regression Model
• Multiple Regression Model

Linear Regression Model:
Linear regression is mostly used to model the relationship between two variables. In most cases, this is completed by fitting a linear condition to observe the data. It may also be used to figure out what the numerical relationship between the variables is. It is the simplest form of Regression. For linear regression, the following equation is used: Y = bX + A is the formula. The value of Y will increment or abatement in a way that the worth of X will change alongside it in a linear way.

Multiple Regression Model:
The most common use of the Various Regression Model is to determine the relationship between multiple free or multiple indicator components. It's possible that it's one of the most well-known forecasting models in datum mining. It usually employs at least two independent elements to predict a result for the clients. The following is the equation that is used in the multiple regression model: +akxk + e = a0 + a1x1 + a2x2+ a3x3+ a4x4+... Y stands for the response variable (the qualities that must be anticipated). The autonomous indicators are X1+X2+X3+X4+Xk. In the preceding equation, 'e' represents an arbitrary error. The regression coefficients are A0, A1, A3, A4, and Ak.

The classification approach provides clients with a predictive model or capability for anticipating new information in distinct categories. This is accomplished with the use of memorable information. Regardless, the regression approach model employs continuously acknowledged capacities to estimate the information's outcome in a never-ending numeric structure. Furthermore, the notion of expected information is unordered in the classification approach, and it is requested in the regression model.

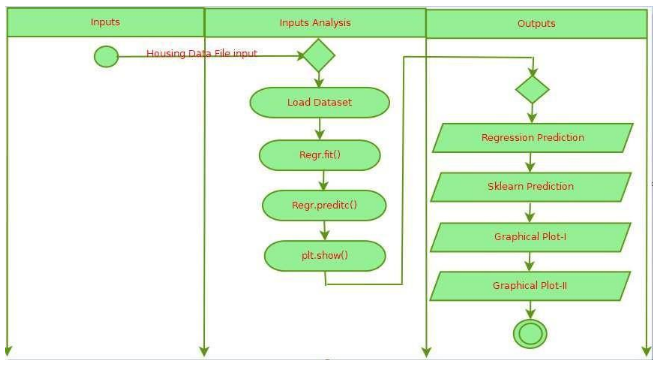
The graphic below depicts a simple USE-CASE diagram of our project.



In the above use case diagram, the actor is the buyer or seller of the house. The following screens are presented to the actor.

•Input property Basic Details - Graphical Interface Screen
•Input view Details – Graphical Interface Screen
•Input Grade Details – Graphical Interface Screen
Input Other parameters – Graphical Interface Screen The last screen will provide the outputs of the 3 classification algorithms used in this project.

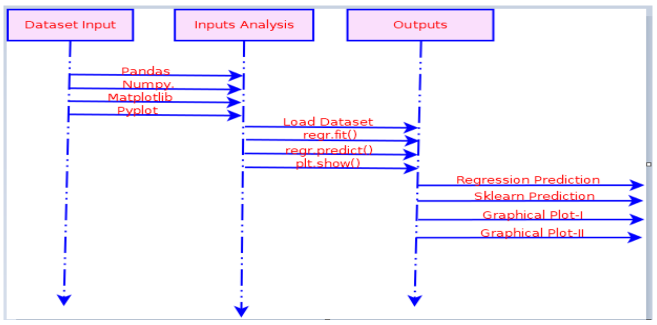
Another important diagram in UML to illustrate dynamic characteristics of the system is the activity diagram. An activity diagram is a kind of flow chart that depicts the flow of information from one activity to another. The action is classified as a system operation. As a result, management flows from one activity to the next. We can see from the activity diagram that the first step is to start the entrance Python procedure, after which we will verify that all of the needed information are present. If it is present, then proceed to the next step; otherwise, fill in the required information. Once the client has successfully entered the basic, grade, and read details, the client must input the opposite parameters and then click the switch, which might result in the analysis and generation of graphs & results.



There are three activation timelines in the above sequence diagram, one for inputs, the second for inputs-analysis, and the third for outputs. This graphic reveal information about the system's internal design. As stated before, the input analyzer receives a Housing Data File in CSV format as input, and the analyzer uses pandas to load the CSV data into the data frame. After the data has been put in the data frame, it is cleaned using appropriate cleaning procedures, which is also known as cold-start problem resolution. The data is now delivered to a regression algorithm, which will train using the data that has been provided.

A sequence diagram is a diagram which shows how protesters collaborate with one another and in what situations. It's a progression of a message flow diagram. A sequence diagram depicts the order in which objects are connected in time.

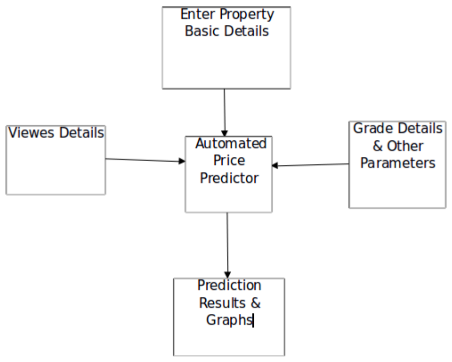
.



Pandas, NumPy, Matplotlib, and Pyplot are the four primary libraries used in this project, according to the sequence diagram. We use these libraries to perform complex mathematical operations on a dataset we obtained from the Kaggle website. To comprehend the relationship between two or more qualities. We use the best feasible regression method after we understand the relationship between the data in the database and different qualities; in our instance, we employed three prediction algorithms: GND, DTC, and regression predictor. Because the python sklearn module contains a large number of prediction algorithms and classifiers, it is critical for the developer to grasp the relationships between the characteristics.

The classifier can forecast the worth of new houses once it has been trained on the cleaned dataset. The plt.show() method is critical in the pyplot lib because without it, the graph depicting the relationship between the two properties would not be visible to the user. We can utilise a formula-based approach or a pre-defined method-based approach for the regression prediction algorithm. In the end, both systems provide the customer with nearly identical expected values.

Details such as fundamental property information, view information, grade information, and other factors should be entered into the system using the appropriate user interfaces. The screenshots of these interfaces are described in the next section. There are a total of five screens that receive user input. These five screens each feature a user interface file, automatic Python code, and human Python code that adds functionality to the entire screen. The system, in its most basic form, has five fundamental interfaces that interact with the price predictor algorithm. The UML diagram that follows will help you understand the overall system architecture.


Enter Property Basic Details
Viewes Details
Automated Price Predictor
Grade Details & Other Parameters
Prediction Results & Graphs

VI.DATA DESCRIPTION

The dataset is taken from the kaggle question pair similarity. In the train.csv, total we have 20,157 records. The data is divided into two parts. The first one is train.csv, which is used in the training part and the second one is the test.csv, where these values are used in test data.

It contains various attributes like number of bedrooms, number of bathrooms, sqft living room, grade, rating etc… Every attribute plays an important role in the dataset. The prediction is dependent on every attribute present in the dataset,

The entire project is divided into 4 modules

1.Data entry module – used to provide the needed data to the project

2.Analysis module – used to analyze and predict the house prices based on customer needs

3.Front – end module – Provides GUI screens to the user to take the input parameters

4.Prediction module – used to predict the price of a house

Algorithms used: Gaussian Naïve Bayes, Decision tree classification, Regression

Inputs: Defective real estate dataset

Output:

1.Regression predictions

2.Decision tree predictions

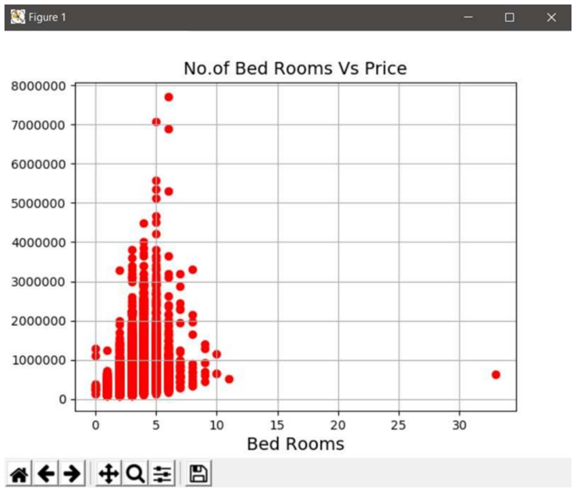
3.Gaussian Naïve bayes predictions

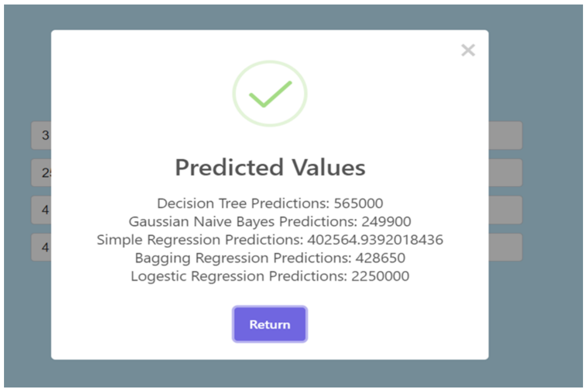
4.Price VS Parameters Graphical plots

VII.RESULTS & ANALYSIS

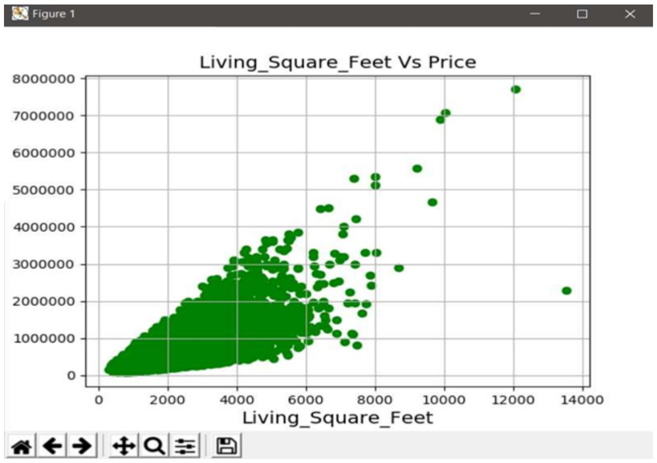
The project's goal is to provide a user-friendly front-end interface that allows users to interact with the database and find property prices accordingly.

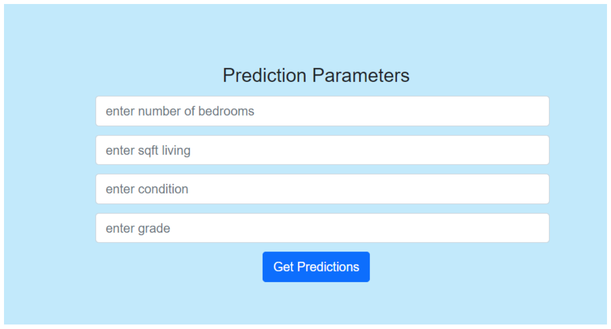
The project is designed in such a way that, it provides the end user's easy access. We have achieved this by integrating the core learning algorithms of the project with the web server. The project directory contains two sub – directories one of the sub-directories is used for the backend and the other is for the front-end. The backend is written in Django and the frontend is written in Angular. Angular is one of the popular front-end frameworks which provide better access to DOM on the front-end. It is better than using Vanilla JavaScript as the actual or raw DOM consumes lot of user code handling and is not much secure.




Predicted Values

Decision Tree Predictions: 565000
Gaussian Naive Bayes Predictions: 249900
Simple Regression Predictions: 402564.9392018436
Bagging Regression Predictions: 428650
Logestic Regression Predictions: 2250000





The home page is where the user enters the details of the house and these details are collected into angular application and sent back to the server. On the server side a post route is executed which collects all these parameters from request body and the regression algorithms are performed on this data collected from front-end. Based on the outputs the data is returned to the front-end using HTTP module and on the front-end side this data is collected using a subscription service and presented on the browser with nice user- interface.

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17, 6 (2005), 734–749.
[2] Taleb Alashkar, Songyao Jiang, Shuyang Wang, and Yun Fu. 2017. Examples-rules guided deep neural network for makeup recommendation. In Proceedings of the AAAI. 941–947. https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14773
[3] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. 2014. Multiple object recognition with visual attention. arXiv preprint arXiv:1412.7755 (2014).
[4] Bing Bai, Yushun Fan, Wei Tan, and Jia Zhang. 2017. DLTSR: A deep learning framework for recommendation of long-tail web services. IEEE Transactions on Services Computing (2017), 1–1.
[5] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems. 107–114. http://doi.acm.org/10.1145/2959100.2959180
[6] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. arXiv preprint arXiv:1706.02263 (2017).
[7] Basiliyos Tilahun Betru, Charles Awono Onana, and Bernabe Batchakui. 2017. Deep learning methods on recommender system: A survey of state-of-the-art. International Journal of Computer Applications 162, 10 (Mar 2017).
[8] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. User Modeling and User-Adapted Interaction 12, 4 (2002), 331–370.
[9] Xiaoyan Cai, Junwei Han, and Libin Yang. 2018. Generative adversarial network based heterogeneous bibliographic
network representation for personalized citation recommendation. In Proceedings of the AAAI. 5747–5754. https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/1630
[10] S. Cao, N. Yang, and Z. Liu. 2017. Online news recommender based on stacked auto-encoder. In Proceedings of theICIS.
721–726.https://ieeexplore.ieee.org/document/7960088
[11] Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In Proceedings of TheRecsys.288–296.
http://doi.acm.org/10.1145/3109859.3109878
[12] Cheng Chen, Xiangwu Meng, Zhenghua Xu, and Thomas Lukasiewicz. 2017. Location-aware personalized news recommendation with deep semantic analysis. IEEE Access 5 (2017), 1624–1638.
[13] Cen Chen, Peilin Zhao, Longfei Li, Jun Zhou, Xiaolong Li, and Minghui Qiu. Locally connected deep learning framework for industrial-scale recommender systems. In Proceedings of the WWW. 769–770. https://doi.org/10.1145/3041021.3054227
[14] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In Proceedings of the SIGIR. ACM. http://doi.acm.org/10.1145/3077136.3080797
[15] Minmin Chen, Zhixiang Xu, Kilian Weinberger, and Fei Sha. 2012. Marginalized denoising autoencoders for domain adaptation. arXiv preprint arXiv:1206.4683 (2012).
[16] Shi-Yong Chen, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. 2018. Stabilizing reinforcement learning in dynamic environment with application to online recommendation. In Proceedings of the SIGKDD. 1187–1196. http://doi.acm.org/10.1145/3219819.3220122
[17] Xu Chen, Yongfeng Zhang, Qingyao Ai, Hongteng Xu, Junchi Yan, and Zheng Qin. 2017. Personalized key frame recommendation. In Proceedings of the SIGIR. 315–324. http://doi.acm.org/10.1145/3077136.3080776
[18] Xu Chen, Yongfeng Zhang, Hongteng Xu, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Visually explainable recommendation. arXiv preprint arXiv:1801.10288 (2018).
[19] Yifan Chen and Maarten de Rijke. 2018. A collective variational autoencoder for top-N recommendation with side information. arXiv preprint arXiv:1807.05730 (2018).
[20] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & deep learning for recommender systems. In Proceedings of the Recsys. 7–10. http://doi.acm.org/10.1145/2988450.2988454