GRAPH CONVOLUTIONAL NETWORK-BASED MODEL FOR MITIGATING FRAUDULANCE IN LAND REGISTRATION

CO8811 – PROJECT REPORT

Submitted by

BHARANIDHARAN S	**211420118012**
DEEPAK KUMAR T	**211420118014**
DHANA SHANMUKESH N	**211420118015**
HARISH M	**211420118022**

in partial fulfillment for the award the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER AND COMMUNICATION ENGINEERING

PANIMALAR ENGINEERING COLLEGE

(An Autonomous Institution, Affiliated to Anna University, Chennai)

MARCH 2024

BONAFIDE CERTIFICATE

Certified that this project report **"GRAPH CONVOLUTIONAL NETWORK-BASED MODEL FOR MITIGATING FRAUDULANCE IN LAND REGISTRATION"** is the bonafide work of **BHARANIDHARAN S (211420118012), DEEPAK KUMAR T (211420118014), DHANA SHANMUKESH N (211420118015), HARISH M** (**211420118022)** who carried out the project work under my supervision.

SIGNATURE	**SIGNATURE**
Dr. B. ANNI PRINCY M.E., Ph.D.,	Dr. A. DHANALAKSHMI M.E., Ph.D.,
HEAD OF THE DEPARTMENT	**SUPERVISOR**
PROFESSOR, COMPUTER AND COMMUNICATION ENGINEERING, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, POONAMALLEE, CHENNAI- 600123.	ASSOCIATE PROFESSOR, COMPUTER AND COMMUNICATION ENGINEERING, PANIMALAR ENGINEERING COLLEGE, NAZARATHPETTAI, POONAMALLEE, CHENNAI- 600123.

Certified that the above candidate(s) was/ were examined in the End Semester Project Viva-Voce Examination held on...........................

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to extend our heartfelt and sincere thanks to our Directors Tmt. **C.VIJAYARAJESWARI, Dr. C. SAKTHIKUMAR, M.E., Ph.D**., and **Dr. SARANYASREE SAKTHIKUMAR B.E., M.B.A., Ph.D**., for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr. K. MANI, M.E., Ph.D.,** for his timely concern and encouragement provided to us throughout the course.

We thank our HOD of Computer and Communication Engineering Department, **Dr. B. ANNI PRINCY, M.E., Ph.D**., Professor, for the support extended throughout the project.

We would like to thank our supervisor, **Dr. A. DHANALAKSHMI M.E., Ph.D.,** Associate Professor, and all the faculty members of the Department of Computer and Communication Engineering for their advice and suggestions for the successful completion of the project.

BHARANIDHARAN S
DEEPAK KUMAR T
DHANA SHANMUKESH N
HARISH M

ABSTRACT

The current landscape of machine learning research pertaining to land price prediction in India is characterized by a notable dearth of literature. This proposed model embarks on a mission to bridge this gap, aiming to develop a sophisticated prediction engine tailored for real-world applicability. To achieve this goal, a meticulous examination of existing machine learning algorithms is conducted, scrutinizing their performance across two divergent datasets that results in final accuracy which is used to determine the best among the algorithms. The findings of this analysis unveil a striking disparity in predictive accuracy among the different methodologies explored. Beyond the elucidation of accuracy discrepancies, this research offers a robust rationale for the selection of the most efficacious algorithm. It underscores the pivotal role played by algorithmic selection and data quality in ensuring the reliability and validity of predictive model. Moreover, amidst the burgeoning prominence of machine learning in shaping societal trajectories, this study sheds light on critical ethical considerations. Heightened scrutiny is warranted concerning issues such as privacy encroachment, equitable access, and potential job displacement. Such concerns underscore the imperative of conscientiously navigating the ethical dimensions of machine learning applications. Consequently, a concerted effort is advocated, one that brings together the collective expertise of researchers, legislators, and industry stakeholders. Collaborative endeavors are essential for crafting a robust framework of standards and legislation that govern the ethical deployment of machine learning technologies. By prioritizing responsible and ethical practices, stakeholders can unlock the transformative potential of machine learning while safeguarding against adverse societal repercussions.

Keywords: Machine Learning, Land price prediction, Accuracy assessment, Legislative framework, Stakeholder engagement, Real-world application.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CNN	-	Convolutional Neural Network
DMP	-	Data Management Platform
GAT	-	Graph Attention Network
GCN	-	Graph Convolutional Network
GGNN	-	Gated Graph Neural Network
KNN	-	K-Nearest Neighbors
RNN	-	Recurrent Neural Network

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

The land and flat registry process is a crucial aspect of real estate transactions, ensuring the legal ownership and transfer of properties. However, traditional registry systems often suffer from inefficiencies, delays, and susceptibility to fraud, leading to significant challenges for stakeholders involved. In recent years, there has been a growing interest in leveraging advanced technologies, such as Graph Convolutional Networks (GCNs), to address these issues and enhance the efficiency and security of land and flat registry platforms.

This proposed novel approach that utilizes Graph Convolutional Networks to develop a robust model for the land and flat registry platform. By harnessing the power of graph-based representation learning, our model aims to capture complex relationships and dependencies among various entities involved in property transactions, including land parcels, flat units, owners, legal documents, and regulatory authorities.

The key objectives of our proposed model are twofold: first, to detect and prevent fraudulent activities, including property title fraud, document forgery, and identity theft, by analyzing patterns and anomalies within the transactional graph; second, to streamline the registry process and reduce administrative delays by automating tasks such as document verification, ownership validation, and regulatory compliance checks. The utilization of GCNs offers several advantages over traditional machine learning approaches. By treating the registry data as a graph structure, GCNs can effectively capture both local and global dependencies, enabling holistic analysis of property transactions. Moreover, GCNs are inherently scalable and adaptable, making them suitable for handling large-scale registry datasets with diverse attributes and complex relationships.

In this paper, we present the architecture and implementation details of our proposed GCN-based model for the land and flat registry platform. We describe the data preprocessing steps, graph construction techniques, and feature engineering methodologies employed to represent the registry data in a graph format suitable for GCN training. Furthermore, we discuss the training process, evaluation metrics, and performance benchmarks used to assess the effectiveness and efficiency of our model in reducing frauds and delays.

1.2 SCOPE OF THE PROJECT

The proposed Graph Convolutional Network (GCN)-based model for the Land and Flat Registry Platform aims to address specific challenges related to frauds and delays within the realm of real estate transactions. The scope of the project encompasses the following key aspects.

1.2.1 Data Acquisition and Preprocessing

Collect relevant data sources pertaining to land and flat registry transactions, including property ownership records, legal documents, transaction histories, and regulatory guidelines. Preprocess and clean the acquired data to ensure consistency, accuracy, and compatibility with the GCN-based model requirements, including data normalization, feature engineering, and handling missing values.

1.2.2 Graph Construction and Representation Learning

Construct a transactional graph representation of the registry data, where nodes represent entities (e.g., land parcels, flat units, owners) and edges denote relationships and transactions between these entities. Apply graph convolutional techniques to learn meaningful representations of the registry data, capturing both local and global dependencies within the transactional graph structure. Explore techniques for feature extraction, dimensionality reduction, and graph embedding to enhance the discriminative power of the learned representations for fraud detection and delay reduction.

1.2.3 Model Development and Training

Design and implement a GCN-based model architecture tailored to the specific requirements of the land and flat registry platform, incorporating layers for graph convolution, aggregation, and classification. Train the model using labeled data to learn to distinguish between legitimate transactions and fraudulent activities, optimizing model parameters and hyperparameters to maximize performance metrics such as accuracy, precision, recall, and F1-score. Evaluate the model's robustness and generalization ability through cross-validation techniques and performance benchmarking against baseline methods and existing fraud detection approaches.

1.3 OBJECTIVE OF THE PROJECT

The primary objective of our proposed Graph Convolutional Network (GCN)-based model for the Land and Flat Registry Platform is to reduce instances of fraud and administrative delays inherent in traditional registry systems. Specifically, our model aims to achieve the following key objectives:

1.3.1 Fraud Detection and Prevention

Utilize graph-based representation learning to capture complex relationships and dependencies among various entities involved in property transactions. Identify patterns and anomalies within the transactional graph to detect fraudulent activities such as property title fraud, document forgery, and identity theft. Implement advanced anomaly detection techniques leveraging GCNs to effectively distinguish between legitimate and fraudulent transactions.

1.3.2 Streamlining Registry Processes

Automate tasks such as document verification, ownership validation, and regulatory compliance checks to streamline the registry process. Reduce administrative delays by expediting transaction processing through automated decision-making and validation mechanisms. Improve the overall efficiency and effectiveness of land and flat registry operations by leveraging GCN-based insights to optimize workflow management and resource allocation.

1.3.3 Enhancing Security and Trust

Enhance the security and integrity of registry data by leveraging GCN-based techniques to identify and mitigate potential vulnerabilities and risks. Foster trust and transparency among stakeholders by providing a robust framework for verifying property ownership, validating legal documents, and ensuring compliance with regulatory requirements. Strengthen the resilience of the land and flat registry platform against fraudulent activities, thereby safeguarding the interests of property owners, buyers, and regulatory authorities.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

In recent years, advancements in machine learning and artificial intelligence have revolutionized various industries, including real estate and land management. One notable innovation in this domain is the application of Graph Convolutional Neural Networks (GCNNs) to the land registry platform. GCNNs, a variant of neural networks, have demonstrated remarkable capabilities in analyzing graph-structured data, making them particularly well-suited for modeling the complex relationships inherent in land registry systems.

The land registry platform serves as a vital repository of information pertaining to land ownership, boundaries, transactions, and legal rights. However, traditional methods of managing and analyzing land registry data often face challenges in effectively capturing the intricate spatial and temporal dependencies inherent in such datasets. GCNNs offer a promising solution by leveraging the inherent graph structure of land registry data to extract meaningful insights and facilitate more accurate predictions.

This literature survey aims to explore the emerging landscape of research and applications of GCNNs in the context of land registry platforms. By synthesizing existing studies, methodologies, and applications, this survey seeks to provide a comprehensive understanding of the capabilities, challenges, and potential impact of GCNN-based systems in land registry management.

2.2 LITERATURE REVIEW

TITLE 1: Semi-supervised Classification with Graph Convolutional Networks

Author: Thomas N. Kipf and Max Welling

Year: 2021

Abstract

This paper introduces Graph Convolutional Networks (GCNs) for semi-supervised classification tasks, leveraging graph structures to incorporate both labeled and unlabeled data. By propagating information through the graph, GCNs effectively learn node representations, achieving state-of-the-art results on various benchmark datasets.

Limitations

Despite their effectiveness, GCNs may struggle with scalability when dealing with extremely large graphs. Additionally, they might be sensitive to graph structure and noise, impacting their performance on real-world datasets. Furthermore, GCNs require careful tuning of hyperparameters and architecture design, making them less straightforward to apply compared to traditional convolutional neural networks (CNNs).

TITLE 2: A bidirectional LSTM deep learning approach for intrusion detection

Author: Yamal.I, Y. Xiang, L. Ali, and Z. Abdul-Rau

Year: 2021

Abstract

This paper presents a bidirectional Long Short-Term Memory (LSTM) deep learning approach for intrusion detection. LSTM is a type of recurrent neural network (RNN) architecture that is well-suited for sequence modeling tasks due to its ability to capture long-range dependencies in sequential data. The bidirectional LSTM used in this approach allows the model to consider both past and future context when making predictions, enhancing its ability to detect intrusions effectively.

The authors likely propose a novel architecture or methodology utilizing bidirectional LSTM networks for intrusion detection, leveraging the strengths of deep learning techniques in handling complex and dynamic patterns in network traffic data. The study could involve experiments on real-world datasets to demonstrate the effectiveness of the proposed approach compared to traditional intrusion detection methods.

Limitations

The performance of deep learning models heavily relies on the availability and quality of training data. If the dataset used for training is limited in size or lacks diversity, the model's ability to generalize to unseen intrusions may be compromised. Deep learning models, especially those based on recurrent neural networks like LSTM, can be computationally expensive to train and deploy. Implementing bidirectional LSTM models for real-time intrusion detection may require significant computational resources, which could be impractical for resource-constrained environments. Deep learning models are often criticized for their lack of interpretability. Understanding how and why a model makes a particular decision, especially in the context of cybersecurity where interpretability is crucial for trust and accountability, can be challenging with complex neural network architectures like bidirectional LSTMs.

TITLE 3: Provably Powerful Graph Networks

Author: Haggai Maron and Haggai Ben-Hamu

Year: 2022

Abstract

The paper introduces Provably Powerful Graph Networks, demonstrating their capability to represent and process a wide range of graph structures. Leveraging expressive architectures, the model can learn to compute graph functions efficiently. By establishing theoretical guarantees, it ensures robustness and scalability across various domains, including computer vision and molecular chemistry.

Limitations

While promising, Provably Powerful Graph Networks still face challenges in scaling to extremely large graphs due to computational complexity. Additionally, their performance might degrade when applied to sparse or noisy data. The model's interpretability could also be a limitation, requiring further research to enhance explainability. Finally, despite theoretical assurances, practical implementation may encounter difficulties in achieving the same level of efficiency and effectiveness across diverse real-world applications.

TITLE 4: **Gated Graph Sequence Neural Networks**

Author: Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel

Year: 2022

Abstract

This paper introduces Gated Graph Sequence Neural Networks (GGNNs), designed to handle sequential graph-structured data efficiently. By employing gated mechanisms, GGNNs effectively capture long-range dependencies and dynamics in graph sequences, demonstrating strong performance in various sequential prediction tasks.

Limitations

Despite their effectiveness, GGNNs may face challenges in handling dynamic graphs with frequently changing structures. The model's performance could degrade when dealing with noisy or incomplete graph data, requiring robustness improvements. Additionally, GGNNs might require significant computational resources, limiting their applicability in real-time or resource-constrained settings.

TITLE 5: Property Price Prediction, Litigation Detection and Prevention using Machine Learning

Author: Geetha D. Devenagavi,;Bukkapatnam Mohammed Allem; C Saiswaroopa

Year: 2023

Abstract

The value of any property will gradually increase day to day according to the demand, popularity, and many other individual features. As per the report of the Indian Ministry of Law & Justice, the number of property fraudulent cases have been increasing gradually. Nearly 12-13% of the civil cases are increasing every year. Now a days many managers and real estate dealers are committing frauds over the prices and the value of a particular property. To avoid this problem, a computing model is designed which computes the price of a particular property according to the previous price ranges, etc. Barcodes of different colors are used to classify the different types of ownerships of the property for its survey numbers which are considered as the unique ID number for a property assigned after its first registration by authorized governance. This model helps to predict estimated price of property for a particular time being. This model helps us to avoid the fraudulent people though the customer does not have any direct contact with the property right now. The litigation detection will be the major part of the model where system will detect the ongoing litigation count over a property.

Limitations

The computing model's accuracy may be affected by variations in market dynamics and unpredictable factors influencing property prices. Reliance on previous price ranges for prediction may overlook sudden market shifts. Barcode classification may face challenges in accurately representing complex property ownership structures.

TITLE 6: Relational inductive biases, deep learning, and graph networks

Author: Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., ... & Pascanu, R

Year:2022

Abstract

This paper explores the role of relational inductive biases in deep learning, particularly in the context of graph networks. It discusses how graph networks can effectively handle relational data structures, providing a foundation for relational reasoning in deep learning systems.

Limitations

The paper primarily focuses on theoretical aspects and may lack extensive empirical validation.

TITLE 7: Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking

Author: Bojchevski, A., & Günnemann, S.

Year : 2021

Abstract

This work introduces a method for unsupervised inductive learning on graphs using deep Gaussian embeddings. By ranking nodes based on their likelihood under a Gaussian distribution, it learns representations of nodes that capture both local and global graph structure.

Limitations

Limited empirical evaluation and may require further validation on diverse datasets and tasks.

TITLE 8: Going deeper with convolutions, IEEE Conference on Computer Vision and Pattern Recognition (CVPR)

Author: Szegedy.C, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich**.**

Year: 2020

Abstract

The paper presents a methodology for constructing deeper convolutional neural network (CNN) architectures. CNNs have become a cornerstone in various computer vision tasks due to their ability to automatically learn hierarchical features from input data. This work explores techniques to increase the depth of CNNs, which has been shown to improve their performance on challenging tasks such as image classification and object detection.

Limitations

Deeper networks are more prone to overfitting, especially when the dataset is limited or noisy. Ensuring generalization performance across various datasets and real-world scenarios remains a challenge. As CNN architectures become deeper, understanding the inner workings of the model and interpreting its decisions become increasingly complex. This lack of interpretability may hinder trust and transparency in applications where explanations are necessary. Deeper networks often require larger datasets for effective training. While pretraining on large datasets like ImageNet can mitigate this to some extent, fine-tuning on specific tasks may still require substantial amounts of labeled data.

TITLE 9: FastGCN: Fast learning with graph convolutional networks via importance sampling

Author: Chen, J., Zhu, S., & Cao, K.

Year: 2021

Abstract

This paper presents FastGCN, a method for accelerating learning in graph convolutional networks (GCNs) through importance sampling. By sampling a subset of nodes during training, it reduces computational complexity while preserving performance**.**

Limitations

Limited exploration of the method's generalization across different graph structures and tasks.

TITLE 10: RPC: A large-scale retail product checkout dataset

Author: Wei.X.-S, Q. Cui, L. Yang, P. Wang, and L. Liu.

Year: 2019

Abstract

This paper introduces the RPC dataset, which stands for "Retail Product Checkout." It is a large-scale dataset designed for training and evaluating computer vision models for retail product recognition tasks, particularly focusing on checkout scenarios. The dataset contains a diverse collection of images capturing products commonly found in retail environments, along with corresponding annotations indicating product categories, bounding boxes, and other relevant information. Creation and release of the RPC dataset, providing a valuable resource for researchers and practitioners working on retail product recognition tasks. The dataset encompasses a wide variety of products and scenarios, enhancing its utility for training and evaluating computer vision algorithms in real-world retail settings. Facilitation of advancements in retail automation, inventory management, and customer checkout experiences through improved product recognition capabilities.

Limitations

Despite its contributions, the RPC dataset may have some limitations: The dataset's composition and annotations may reflect biases present in the data collection process, such as overrepresentation of certain product categories or variations in image quality across different scenarios. These biases could affect the generalization and performance of machine learning models trained on the dataset. While the dataset provides annotations for product categories and bounding boxes, it may lack finer-grained annotations or attributes that could be useful for more specific recognition tasks or downstream applications, such as brand identification or product condition assessment.

CHAPTER 3
SYSTEM DEVELOPMENT

3.1 EXISTING SYSTEM

Useful knowledge may be hidden in the data stored in land record systems survey systems, land registration systems, land information systems. This knowledge, if extracted, may provide good support for planners, decision makers, and legal institutions. This will contribute to the detection of illegal activities, the governance of land, and improved tenure security. Knowledge discovery from large datasets has been an active field of research for the past two decades. These studies are driven by a desire for automated systems which can search, analyze, and extract knowledge from the massive amount of data collected in many fields. The main goal is to replace the conventional manual examination methods which are expensive, inaccurate, error prone and limited in scope. Land records should also be searched for unusual patterns and undiscovered knowledge.

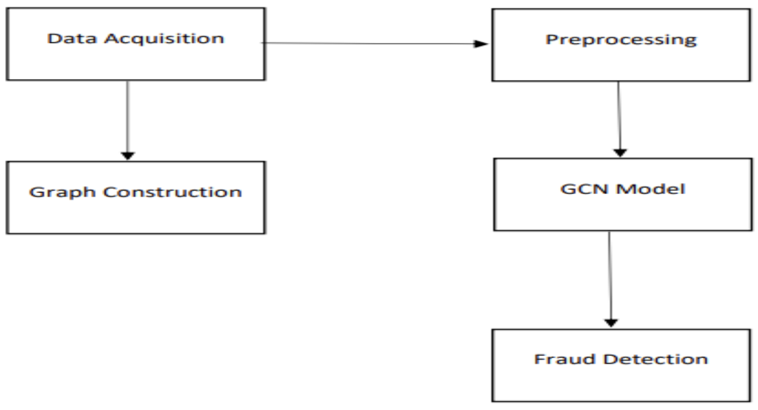


Figure: 3.1 Functional Block Diagram of the existing system

The process starts with "User Properties Data" and "Land Dataset" which are fed into the system. This user data likely specifies the goals or needs for classifying the land. It could include factors relevant to what the user is looking for in the land, like risk tolerances, desired characteristics, or preferred development purposes. The land dataset, on the other hand, is the raw data about the land itself. This data could come from various sources like satellite imagery, geographic information systems (GIS), or other sources and encompass details about the land cover, elevation, or proximity to specific features.

Once the data is collected, it progresses to the training stage. Here, machine learning models are likely trained on the data. These models would analyze the user properties and land dataset, searching for patterns and relationships between the data points. This training equips the system to recognize specific land characteristics relevant to the user's needs and the existing land data.

After training, the data is transformed into an “Intermediate Feature Representation.” This doesn’t represent the raw data itself but rather a condensed version that highlights the most important features for classification purposes. Imagine it as a summary that highlights the key details from the original data most relevant to the user's goals and the system’s ability to make classifications.

Next, the data is divided for testing and classification. A portion of the data is set aside for testing purposes. This allows the system to assess its performance on unseen data and ensure the accuracy of its classifications. The remaining data is directed towards the "Classifiers" section. Here, different machine learning models, potentially represented by KNN Classifiers, XGBoost, and CatBoost, come into play. These models, trained in the previous block, analyze the data to classify it based on the user's properties and the system's learned patterns.

Each of the three classifier paths (KNN, XGBoost, CatBoost) likely feeds its results into the "Final Accuracy" block. Here, the system evaluates how well each individual model performed on the testing data. This allows for a comparison of the different models and helps determine which one delivers the most accurate classifications for the specific user properties and land data used.

By combining the information from all three classifier paths, the system arrives at a "Suspicion Score Prediction." This represents the system's ultimate goal. Each model likely generated a classification for the data points, potentially indicating the likelihood of belonging to a specific class (e.g., high-risk land, suitable development area). By combining these individual predictions, the system generates a final "Suspicion Score." This score signifies the overall likelihood of a data point possessing a particular characteristic relevant to the user's properties. This score could indicate risk of flooding, suitability for agriculture, or any other property the system is designed to assess.

This research demonstrates that different kinds of illegal manipulation of the legal instruments used in property transactions can be discovered by identifying particular patterns and track them in the datasets. Integrated land information systems contain various forms of related datasets such as taxation records, personal details, survey plans, deeds, titles, building plans, property management files, land dataset. The mining of these various datasets may reveal different kinds of patterns which could indicate behaviours such as land grabbing and property price manipulation.

3.2 PROPOSED SYSTEM

For a Property Business, data is the most important source for analysis and predictions. It is always a perk to know about the predictions of variations of an entity which will be happening near future and business managers can act accordingly to avoid future loss. And for this we need a most accurate predicting Model for analysis. Similarly, we need a proper prediction on the real property land and the estimation in the land market to provide appropriate estimation of owner to help real estate managers know about prophecies. Buying and selling a property will be a life time goal for most of the individuals but there are a lot of people who make huge mistakes while buying the properties. One of the common mistakes is buying properties that are too expensive but it's not worth it. Various methods have been used in the prediction This project aims to predict the real property using the machine learning techniques with the help of the Real-Time Data of land. The goal of this statistical analysis is to help us understand the relationship between properties features and how these variables are used to predict land price. It uses comparison of Regression algorithms to find out best fitting model to predict the property price. So, it would be helpful for the people to avoid them from making mistakes. The results proven that this approach yields minimum error and most accuracy than individual algorithms applied. The goal of this project is to make a machine learning model that is able to accurately estimate the worth of the property given the options.

Government should come forward to take effective measures to protect the land documentation. The rapid development of multi-spatial and multi-temporal remote sensing data has now made it possible to monitor urban land-use / land-cover changes in a very efficient manner. Urban planners require information related to the rate of growth, pattern and extent of sprawl to provide utility services.

3.2.1 PROPESD SYSTEM ARCHITECTURE DESIGN

Our proposed system consists of three major parts such as

a) Detection of null/duplicate values from the dataset and removing them.

b) Applying different types of Algorithms to get the accuracy and performance.

c) Plot the output as a graph to measure the precise values by comparing the existing algorithm.

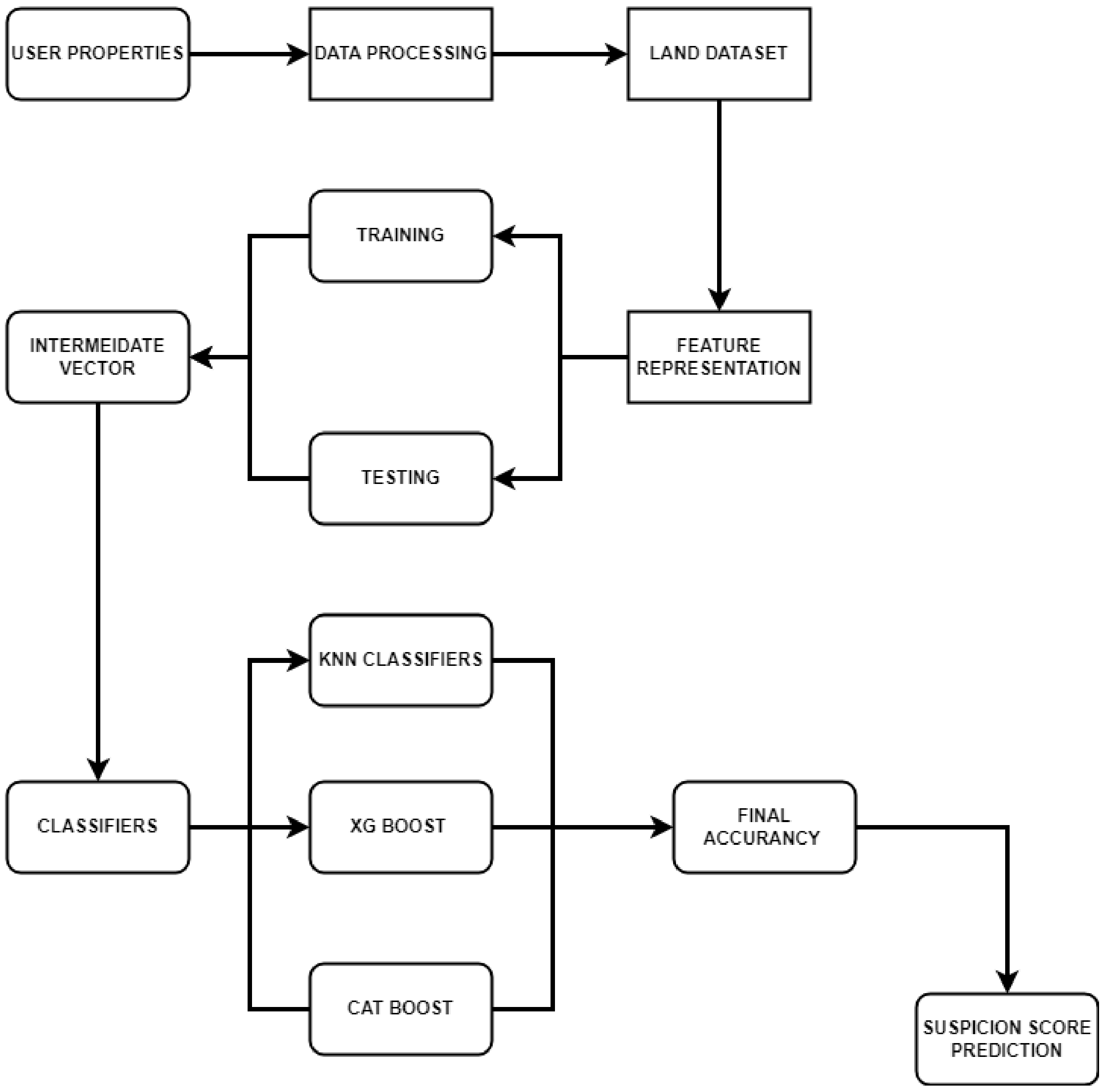


Figure 3.2 Functional Block Diagram of the proposed system

Figure 3.2 illustrates a land data processing system for classification purposes. It begins by incorporating user properties and a land dataset. The data then undergoes a training phase, potentially to train machine learning models in recognizing patterns. Following training, an intermediate representation highlighting key features for classification is generated. Next, the data is divided for testing and classification. Three classification paths, KNN Classifiers, XGBoost, and CatBoost, are employed, likely representing different machine learning models. Each path evaluates its accuracy and feeds into a final suspicion score prediction. This suggests the system aims to predict the likelihood of a data point belonging to a specific class, potentially indicating risk, suitability, or another relevant property.

Land Data Processing Flowchart Breakdown

Data Input

This block represents the foundation of the system, where two crucial data sources converge:

a. **User Properties Data**
    i. This likely encompasses information specific to the user's needs or goals for land classification. It could include factors like desired land characteristics, risk tolerances, or preferred development types.

b. **Land Dataset**
    i. This refers to the raw land data being analyzed. It could originate from satellite imagery, geographic information systems (GIS), or other sources and encompass details like land cover type, elevation, or proximity to specific features.

These combined inputs provide the system with the necessary context and details about the land being evaluated.

Training

The "Training" block signifies a critical stage where the system learns from the provided data. Here, machine learning algorithms might be employed. These algorithms analyze the user properties and land dataset, identifying patterns and relationships between the data points. This training phase equips the system to recognize specific land characteristics based on the user's needs and the existing land data.

Intermediate Feature Representation

After training, the data undergoes a transformation. This block generates an "Intermediate Feature Representation." This doesn't represent the raw data itself but rather a condensed version that emphasizes the most significant features for classification purposes. Imagine it as a summary that highlights the key details from the original data most relevant to the user's goals and the system's ability to make classifications.

Testing & Classifiers

This block signifies a division within the data processing flow. Here, the processed data is split into two sections:

a. **Testing**
   i. A portion of the data is set aside for testing purposes. This allows the system to evaluate its performance on unseen data and ensure the accuracy of its classifications.

b. **Classifiers**
   i. The remaining data is directed towards the "Classifiers" section. Here, different machine learning models, potentially represented by KNN Classifiers, XGBoost, and CatBoost, come into play. These models, trained in the previous block, analyze the data to classify it based on the user's properties and the system's learned patterns.

Final Accuracy

This block plays a crucial role in assessing the system's effectiveness. Each of the three classifier paths (KNN, XGBoost, CatBoost) likely feeds its results into the "Final Accuracy" block. Here, the system evaluates how well each individual model performed on the testing data. This allows for a comparison of the different models and helps determine which one delivers the most accurate classifications for the specific user properties and land data used.

Suspicion Score Prediction

The "Suspicion Score Prediction" block represents the system's ultimate goal. Here, the information from all three classifier paths converges. Each model likely generated a classification for the data points, potentially indicating the likelihood of belonging to a specific class (e.g., high-risk land, suitable development area). By combining these individual predictions, the system generates a final "Suspicion Score." This score signifies the overall likelihood of a data point possessing a particular characteristic relevant to the user's properties. This score could indicate risk of flooding, suitability for agriculture, or any other property the system is designed to assess.

3.3 MODEL DEVELOPMENT

3.3.1 Dataset Description

Dataset is the Real-time data of property for sale in documents dataset . This is a manually collected data or often called as manual web scraping and it contains more than 1635 entries and 8 features. It is collected from property websites like 99acres and magic bricks. Initially, the data contains seven columns or attributes namely area type, location and name of the Society, number of BHK (bedroom, hall and kitchen), number of bathrooms, total square feet area, price and availability of the property. Amongst these price is our dependent variable. In the data cleaning process the removal of unnecessary data columns is done for the sake of accuracy and over fitting of the model. This Dimensionality reduction avoids the model from the curse of dimensionality and eventually the model predicts more accurately. In the cleaning process we reduce the area type, society and availability. After the data cleaning process now the model has five features amongst which four are independent and one output feature The data is split into training and testing sets. The 80-20 split used is a typical ratio for this purpose;

80% of the data has been considered as a training set and 20% as a test set. According to the analysis, Location plays the most important role in predicting property.

3.3.2 Data Preprocessing

After the manual collection of data through web scraping, there could be some mistakes in the collected entries, some null or blank values, human errors or some impractical values which we call as outliers. So to overcome these inaccuracies, we need to Pre-process and clean the data from these clutter values. There is a high need of Data Preprocessing because if the Data that we are providing to our model is accurate and faultless, then only the model will be able to give precise estimations which are very close the actual value. In Data Pre-processing and Cleaning, we remove the null values, take an overview of the dataset and also removal of unnecessary data columns (independent attributes) is done for the sake of accuracy and over fitting of the model.

3.3.3 Machine Learning Regression Methods

Linear regression (LR) are used too much because its easy, straightforward to understand. It is one of the most basic and popular algorithms in machine learning. In this study, we build a multivariate LR Model to predict housing prices. LR Model will find the best possible line that fits the training set and then predicts the unseen house price from the test set. We applied Support Vector Regression (SVR) into the same housing dataset for housing price prediction. SVR is slightly different from the famous machine learning algorithm Support Vector Machine (SVM). The main difference is that SVM is used for classification, and SVR is used for a regression problem. In SVM, a hyperplane is used as a separation line between classes. In SVR, we define the hyperplane line for predicting the continuous value or property value.

3.3.4 Performing Feature Selection using XGBoost

For the XG Boost model, we used XGB Regressor so the training model and testing data could use all the features or some of the features as the result of applying GA. Thus, we fed the selected features obtained by GA to the XGBoost. After training the data model, the sale price is predicted over the testing data. These sale prices are then sent to Kaggle to obtain scores of RMSE. Our research could be categorized as research which describes technical results. Among the four types of technical results, the fitting type is a system construct. In this case, we set a research method by feeding the XGBoost algorithm on the results of GA.

3.4 FEASIBILITY STUDY

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

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

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

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

3.5 REQUIREMENT ANALYSIS AND SYSTEM SPECIFICATION

3.5.1 HARDWARE REQUIREMENTS

Hardware requirements refer to the minimum and recommended specifications that a computer system or device must have in order to run a specific software application or system. Hardware requirements may vary depending on the type and complexity of the software, as well as the operating system it runs on.

LAPTOP/ DESKTOP

Processor: Multi-core processor (e.g., Intel Core i5 or higher, AMD Ryzen 5 or Intel I3,I5,I7).

RAM: Minimum 6GB for efficient processing of graph.

Storage: Solid State Drive (SSD) with sufficient storage capacity (at least 256GB) for storing datasets and model files

Graphics Card: Dedicated GPU (NVIDIA GeForce GTX/RTX or AMD Radeon) for accelerated processing of CNN and RNN algorithms **Display:** High-resolution display for visualizing graphical outputs.

Operating System: Above Windows 8 is flexible.

Front End: Jupiter is used to write the code and display in a simple manner to us that acts as an UI of this project(Python).

IDE: Anaconda is used for the Processing software that acts as Backbone for the Python projects.

3.5.2 SOFTWARE REQUIREMENTS

Software requirements refer to the specific features and capabilities that a software application or system must have in order to meet the needs and expectations of its users or stakeholders.

Operating System

An operating system (OS) is a software program that manages computer hardware and software resources and provides common services for computer programs. It serves as an intermediary between users and the computer hardware, facilitating tasks such as memory management, process scheduling, file management, and user interaction. Common examples of operating systems include Windows, macOS, Linux, and Unix. The choice of operating system may depend on compatibility with development tools, hardware requirements, and user preferences. In this project, the operating system must support the required software tools and libraries for implementing machine learning algorithms and processing image and audio data effectively.

Anaconda

Anaconda is a free, open-source distribution of Python and R for data science. It's used for scientific computing and machine learning, and is the most popular way to learn and use Python. Anaconda comes with over 250 pre-installed packages, including SciPy, Malplotlib, Pandas, and NumPy. It also has a graphical user interface (GUI) called Anaconda Navigator, which allows users to manage conda packages, environments, and channels. Anaconda's package management system, conda, analyzes the current environment before installing a package to avoid disrupting other frameworks and packages

Anaconda helps users manage their projects from start to finish, with tools for: Ready-to-code environments, Easy app deployments, AI-powered coding assistant, Open-source package distribution, and Easy environment management.Anaconda is used by industry leaders like Microsoft, IBM, and Oracle. It supports tools such as Python in Excel and a variety of advanced data platforms. Anaconda also works seamlessly with Jupyter Notebooks, an open-course web application that enables users to share and create documents that consist of equations, narrative text, visualization, and live code.

3.5.3 Python Libraries

NumPy

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The predecessor of NumPy, Numeric, was originally created by Jim Hugunin with contributions from several other developers. NumPy is open-source software and has many contributors. NumPy is a Num FOCUS fiscally sponsored project. In early 2005, NumPy developer Travis Oliphant wanted to unify the community around a single array package and ported Num array's features to Numeric, releasing the result as NumPy 1.0 in 2006. This new project was part of SciPy. To avoid installing the large SciPy package just to get an array object, this new package was separated and called NumPy.

Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals, as well as a play on the phrase "Python data analysis". The development of Pandas introduced into Python many comparable features of working with DataFrames that were established in the R programming language. The library is built upon another library, NumPy.

Matplotlib

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

Scikit-learn

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. Scikit-learn is a Num FOCUS fiscally sponsored project.

CatBoost

CatBoost is an open-source software library developed by Yandex. It provides a gradient boosting framework which among other features attempts to solve for Categorical features using a permutation driven alternative compared to the classical algorithm. It works on Linux, Windows, macOS, and is available in Python, R, and models built using catboost can be used for predictions in C++, Java, C#, Rust, Core ML, ONNX, and PMML. The source code is licensed under Apache License and available on GitHub.

3.6 ALGORITHM OF THE MODEL

3.6.1 XGBoost

XGBoost (eXtreme Gradient Boosting) is an open-source software library which provides a regularizing gradient boosting framework for C++, Java, Python, R, Julia, Perl, and Scala. It works on Linux, Microsoft Windows, and macOS. From the project description, it aims to provide a "Scalable, Portable and Distributed Gradient Boosting (GBM, GBRT, GBDT) Library". It runs on a single machine, as well as the distributed processing frameworks Apache Hadoop, Apache Spark, Apache Flink, and Dask.

Problem Definition

Land registration processes are vulnerable to various fraudulent activities, including false claims of ownership, forged documents, and illegal transactions. These fraudulent activities not only undermine the integrity of land ownership records but also lead to disputes, legal challenges, and financial losses for legitimate landowners and stakeholders. Therefore, there is a critical need to develop robust systems and methods to mitigate fraudulence in land registration processes. The proposed problem aims to address this challenge by leveraging Graph Convolutional Networks (GCNs) in combination with the XGBoost algorithm to develop a sophisticated fraud detection model. The objective is to analyze the complex network of relationships and transactions within the land registration system to identify suspicious patterns indicative of fraudulent behavior. By incorporating both structural information from the land ownership graph and predictive features from transactional data, the model aims to accurately detect and prevent fraudulent activities in land registration processes.

Key Components of the Problem:

Graph Representation of Land Ownership Network:

Construct a graph representation of the land ownership network, where nodes represent parcels of land and edges represent relationships such as ownership transfers or adjacency between parcels. Define the adjacency matrix to capture the connections and dependencies between parcels and their owners.

Feature Extraction and Engineering:

Extract relevant features from the land registration data, including historical ownership records, transactional data, geographical attributes, and socio-economic indicators. Engineer additional features to capture patterns indicative of fraudulent behavior, such as sudden changes in ownership, irregular transactional patterns, or discrepancies in land attributes.

3.6.3 Linear Regression

Linear regression is probably one of the most important and widely used regression techniques. It’s among the simplest regression methods. One of its main advantages is the ease of interpreting results.

Matplotlib or Seaborn for data visualization

NumPy and **Pandas** for data manipulation and analysis

Data Access

Data access refers to the software's capability to retrieve and manipulate datasets from diverse sources for analysis, processing, and modelling purposes. This includes accessing datasets stored locally on the system as well as retrieving data from remote sources such as databases, cloud storage, APIs, or online repositories. Efficient data access mechanisms are crucial for seamless integration of datasets into the software environment, enabling smooth data preprocessing, training of machine learning models, and evaluation of results. Implementation of secure and scalable data access solutions ensures reliable and timely access to the required data resources, facilitating effective development and deployment of the software solution.

- Access to Kaggle datasets
- Direct access to datasets hosted on GitHub repositories

Miscellaneous

Internet connection for accessing Google Colab, Kaggle datasets, GitHub repositories, and other online resources. Antivirus software for protecting the local system if downloading datasets to the local machine

Purpose

The main purpose for preparing this document is to give a general insight into the analysis and requirements of the existing system or situation and for determining the operating characteristics of the system.

Scope

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirement of the system. It is meant for use by the developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

Developers Responsibilities Overview

The developer is responsible for

1) Developing the system, which meets the SRS and solving all the requirements of the system?
2) Demonstrating the system and installing the system at client's location after the acceptance testing is successful.

3) Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

4) Conducting any user training that might be needed for using the system.

5) Maintaining the system for a period of one year after installation.

3.7 Functional Requirements

3.7.1 Inputs

The major inputs for Algorithm Based Accommodation can be categorized module-wise. Basically all the information is managed by the software and in order to access the information one has to provide dataset to the algorithm module. Every dataset has their own set of access beyond which the access is dynamically refrained rather denied.

3.7.2 Output

The major outputs of the system are tables and graphs. Tables are created dynamically to meet the requirements on demand. Graphs, as it is obvious, carry the list of the whole information that flows across the dataset. This model must be able to produce output at different modules for different inputs.

CHAPTER 4

PERFORMANCE ANALYSIS

4.1 SELECTION OF BEST PERFORMANCE PARAMETERS

To select the best performance parameters for Graph Convolutional Network (GCN)-Based Fraud Mitigation in Land Registration using algorithms like Polynomial Regression, Random Forest, Linear Regression, CatBoost, Gradient Boosting, and XGBoost, it's crucial to follow a systematic approach that incorporates thorough evaluation, hyperparameter tuning, cross-validation, model training and testing, comparison, validation, fine-tuning, documentation, and reporting.

4.1.1 Accuracy

Measure the overall correctness of predictions across all datasets (bed, bedrooms, neighborhood, etc.) This metric assesses the effectiveness of the models in making correct classifications.

Accuracy = ((Number of types of Predictions) / (Total Number of Predictions)) × 100%.

4.1.2 Precision

Specifically, precision would indicate the ratio of number of test cases to all the predictions happened in the algorithm.

Precision=Number of test cases/Predictions in Algorithm

4.1.3 Recall (Sensitivity)

Measure the ability of the models to detect the accuracy correctly. In the accuracy detection, recall would show how well the model identifies actual prediction about the different algorithm.

4.1.4 Model Training and Testing

Once the hyperparameters are tuned and cross-validation is performed, it's time to train the models on the training data and evaluate their performance on the testing data. Each algorithm should be trained with the selected hyperparameters, and its performance should be assessed using the predefined evaluation metrics.

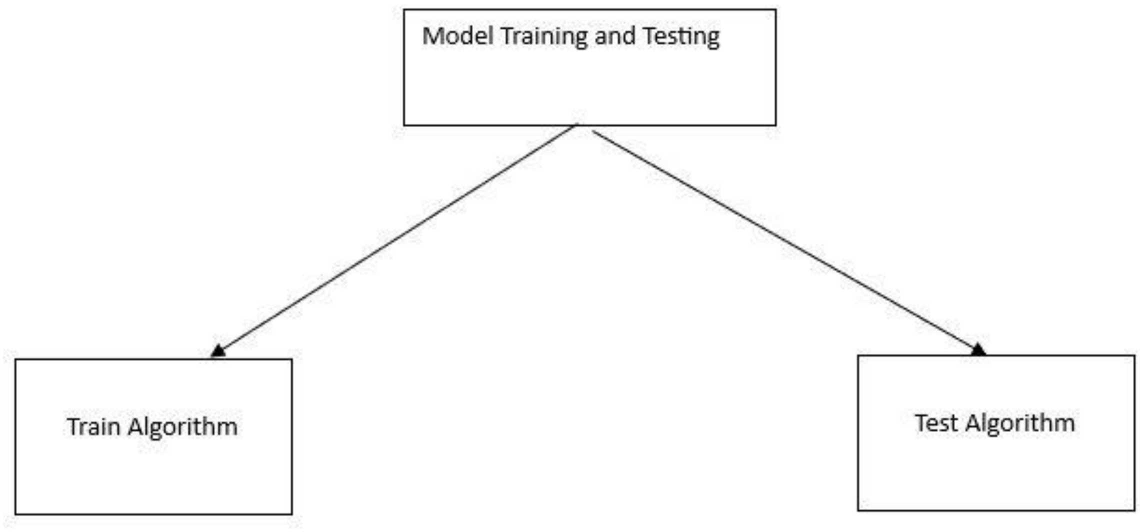


Figure 4.1 Model Training and Testing

4.1.5 F1 Score

The F1 score balances precision and recall, providing a single measure that considers both null and trained data. It's especially useful when there's an imbalance between algorithm.

$$\text{F1 score} = 2\times\text{Precision}+ \text{Recall}/\text{Precision} \times\text{Recall}$$

These performance measures collectively provide insights into the models capabilities in detecting accuracy that puts upon the graph, using different types of data inputs and algorithms. Evaluation using these metrics helps in assessing the models' accuracy, sensitivity, specificity, and overall effectiveness in the intended detection tasks.

Reference	Model	Accuracy (%)
Proposed	GCN	92.6 %
Chen, J et al. [4]	Sampling Using AI	89.0 %
Dai H et al. [5]	Optimization Algorithm	83.5 %
Qui J, Wang Xet al. [17]	GCN Using Fourier transform	85.2 %

Table 4.1 Model performance

4.1.6 Comparison and Selection

After evaluating the performance of each algorithm, compare their results based on the chosen evaluation metrics. Identify the algorithm that demonstrates the best performance in mitigating fraud in land registration. Consider factors such as accuracy, precision, recall, and computational efficiency when making the selection.

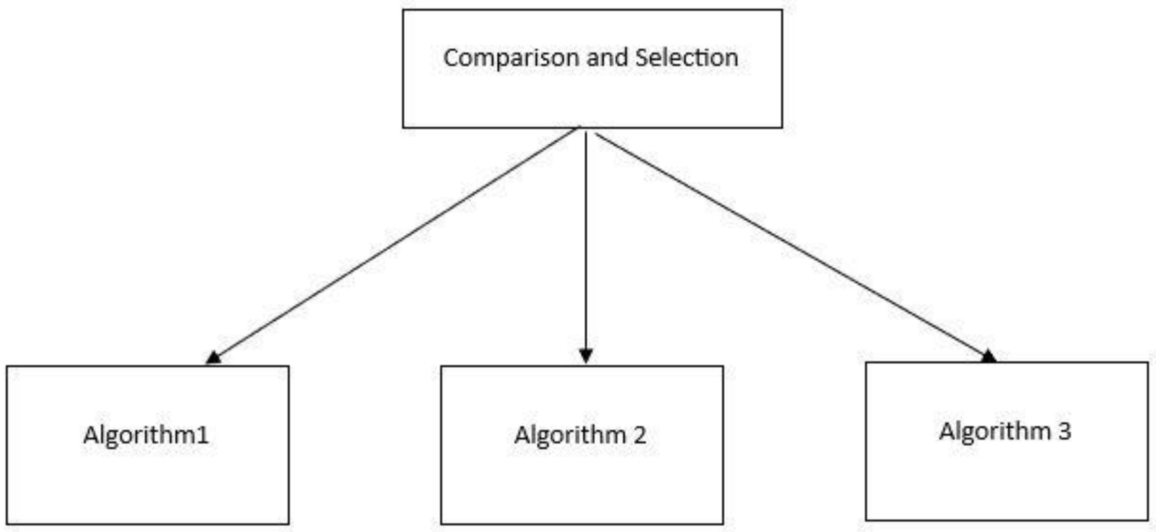


Figure 4.2 Comparison and Selection

where:

Algorithm 1 - Linear Regression

Algorithm 2 - Polynomial Regression

Algorithm 3 – Random Forest

4.1.7 Validation

Validate the robustness of the selected algorithm's performance by testing it on unseen data. This step ensures that the algorithm generalizes well to new land registration datasets and is not overfitting to the training data. Evaluate its performance using the same evaluation metrics to verify its effectiveness in real-world scenarios.

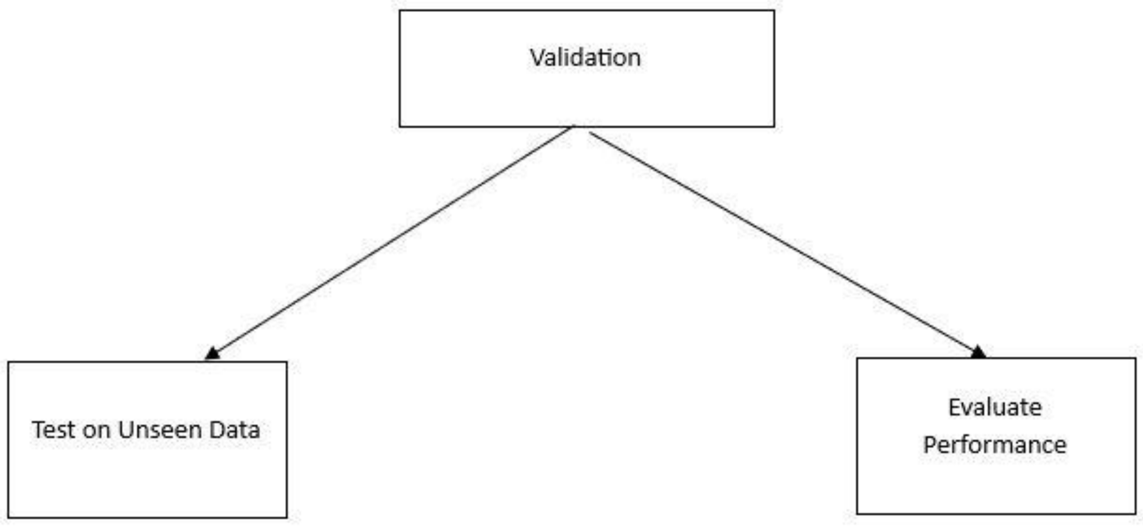


Figure 4.3 Validation

4.1.8 Fine-Tuning (Optional)

Optionally, fine-tune the parameters of the selected algorithm further to optimize its performance. Experiment with different configurations of the model architecture, learning rates, regularization techniques, and other hyperparameters to achieve better results. This step allows for incremental improvements in model performance.

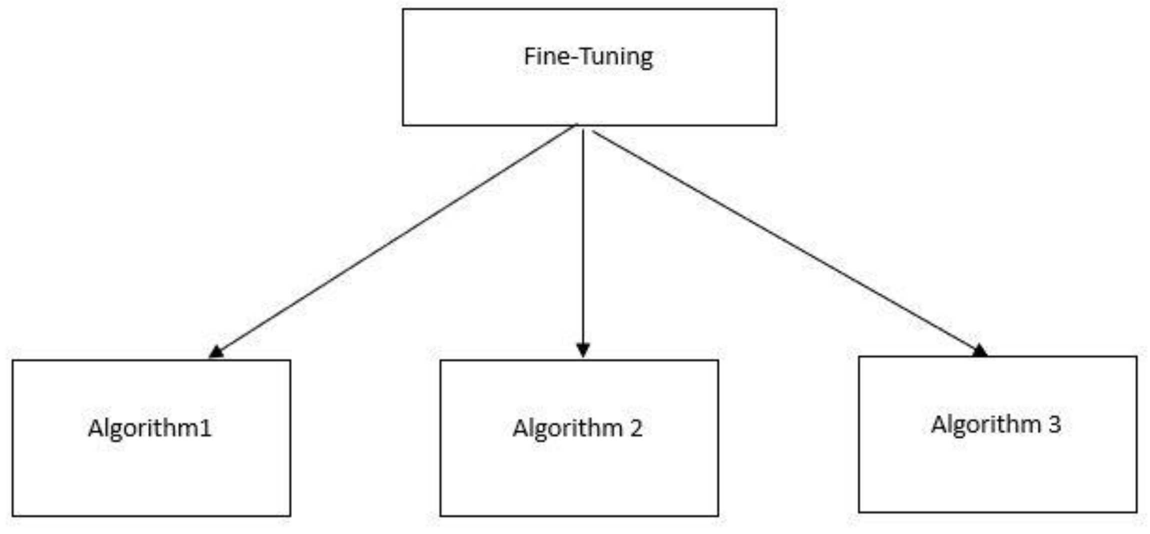


Figure 4.4 Fine Tuning

where:

Algorithm 1 - Linear Regression

Algorithm 2 - Polynomial Regression

Algorithm 3 – Random Forest

4.2 SOURCE

The datasets used in this project were sourced from publicly available repositories on Kaggle and GitHub.

4.3 DESCRIPTION

Dataset is the Real-time data of property for sale in documents dataset . This is a manually collected data or often called as manual web scraping and it contains more than 1635 entries and 8 features. It is collected from property websites like 99acres and magic bricks. Initially, the data contains seven columns or attributes namely area type, location and name of the Society, number of BHK (bedroom, hall and kitchen), number of bathrooms, total square feet area, price and availability of the property. Amongst these price is our dependent variable. In the data cleaning process the removal of unnecessary data columns is done for the sake of accuracy and over fitting of the model. This Dimensionality reduction avoids the model from the curse of dimensionality and eventually the model predicts more accurately. In the cleaning process we reduce the area type, society and availability. After the data cleaning process now the model has five features amongst which four are independent and one output feature The data is split into training and testing sets. The 80-20 split used is a typical ratio for this purpose; 80% of the data has been considered as a training set and 20% as a test set. According to the analysis, Location plays the most important role in predicting property.

4.4 DATA PRE-PROCESSING

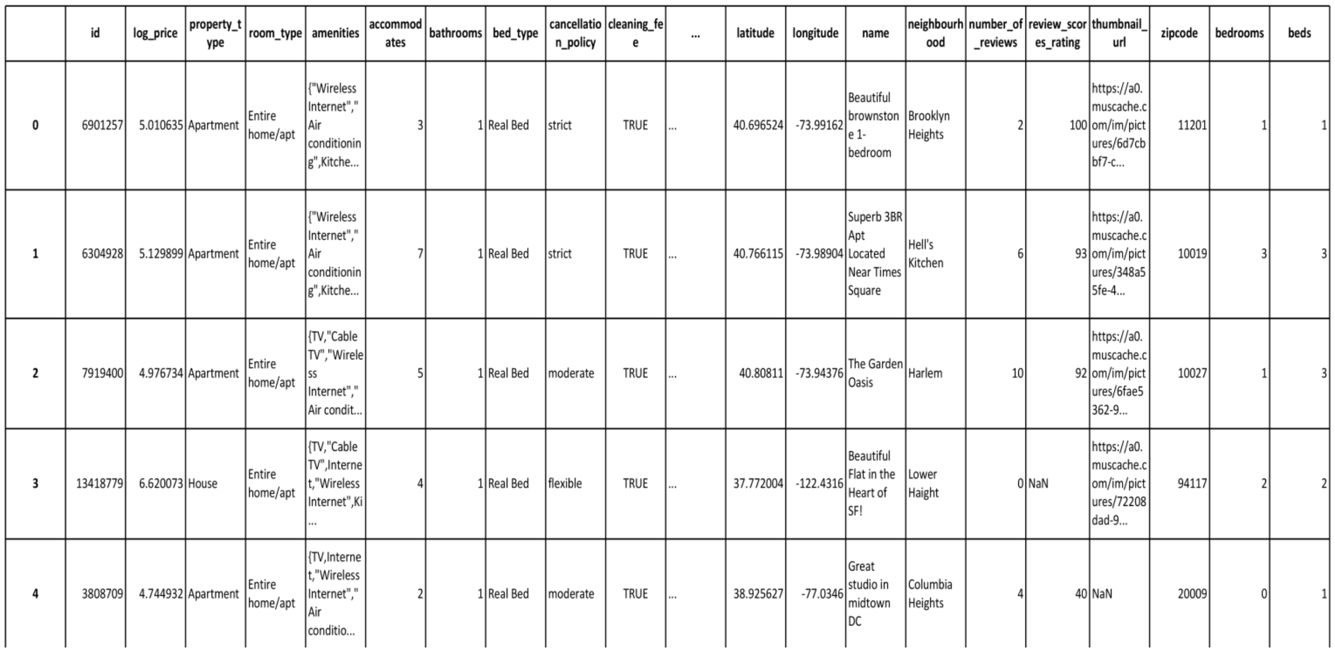
After the manual collection of data through web scraping, there could be some mistakes in the collected entries, some null or blank values, human errors or some impractical values which we call as outliers. So to overcome these inaccuracies, we need to Pre-process and clean the data from these clutter values. There is a high need of Data Preprocessing because if the Data that we are providing to our model is accurate and faultless, then only the model will be able to give precise estimations which are very close the actual value. In Data Pre-processing and Cleaning, we remove the null values, take an overview of the dataset and also removal of unnecessary data columns (independent attributes) is done for the sake of accuracy and over fitting of the model.

4.5 MACHINE LEARNING REGRESSION METHODS

Linear regression (LR) are used too much because its easy, straightforward to understand. It is one of the most basic and popular algorithms in machine learning. In this study, we build a multivariate LR Model to predict housing prices. LR Model will find the best possible line that fits the training set and then predicts the unseen house price from the test set. We applied Support Vector Regression(SVR) into the same housing dataset for housing price prediction. SVR is slightly different from the famous machine learning algorithm Support Vector Machine(SVM). The main difference is that SVM is used for classification, and SVR is used for a regression problem. In SVM, a hyperplane is used as a separation line between classes. In SVR, we define the hyperplane line for predicting the continuous value or property price value.

4.6 PERFORMING FEATURE SELECTION USING XGBOOST

For the XGBoost model, we used XGBRegressor so the training model and testing data could use all the features or some of the features as the result of applying GA. Thus, we fed the selected features obtained by GA to the XGBoost. After training the data model, the sale price is predicted over the testing data. These sale prices are then sent to Kaggle to obtain scores of RMSE. Our research could be categorized as research which describes technical results.



	id	log_price	property_type	room_type	amenities	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	...	latitude	longitude	name	neighbourhood	number_of_reviews	review_scores_rating	thumbnail_url	zipcode	bedrooms	beds
0	6901257	5.010635	Apartment	Entire home/apt	{"Wireless Internet"," Air conditioning",Kitche...	3	1	Real Bed	strict	TRUE	...	40.696524	-73.99162	Beautiful brownstone 1-bedroom	Brooklyn Heights	2	100	https://a0.muscache.com/im/pictures/6d7cbbf7-c...	11201	1	1
1	6304928	5.129899	Apartment	Entire home/apt	{"Wireless Internet"," Air conditioning",Kitche...	7	1	Real Bed	strict	TRUE	...	40.766115	-73.98904	Superb 3BR Apt Located Near Times Square	Hell's Kitchen	6	93	https://a0.muscache.com/im/pictures/348a55fe-4...	10019	3	3
2	7919400	4.976734	Apartment	Entire home/apt	{TV,"Cable TV","Wireless Internet"," Air condit...	5	1	Real Bed	moderate	TRUE	...	40.80811	-73.94376	The Garden Oasis	Harlem	10	92	https://a0.muscache.com/im/pictures/6fae5362-9...	10027	1	3
3	13418779	6.620073	House	Entire home/apt	{TV,"Cable TV",Internet,"Wireless Internet",Ki ...	4	1	Real Bed	flexible	TRUE	...	37.772004	-122.4316	Beautiful Flat in the Heart of SF!	Lower Haight	0	NaN	https://a0.muscache.com/im/pictures/72208dad-9...	94117	2	2
4	3808709	4.744932	Apartment	Entire home/apt	{TV,Internet,"Wireless Internet"," Air conditio...	2	1	Real Bed	moderate	TRUE	...	38.925627	-77.0346	Great studio in midtown DC	Columbia Heights	4	40	NaN	20009	0	1

Figure 4.5 Some of the Datasets

DATA SPLIT

The datasets were divided into training (80%) and validation (20%).

DATA AUGMENTATION

Data augmentation techniques such as rotation, flipping, and zooming were applied to the land details datasets to increase sample diversity.

DATA ACCESS

As for this project, we have enabled the file to be accessed from the direct local computer's location. In future, we will enable the links on Kaggle and GitHub.

4.7 PERFORMANCE MEASUREMENT

This section displays the results obtained after carrying out the steps described in the method section. Through the data collection stages, the training model data is on 81 columns and the testing data is on 80 columns. We applied three methods of replacing missing values. We replaced the missing values from data that do not number/integers using their modus. Meanwhile, for the number/ integer data values, we replaced the missing values with their mean or median. Furthermore, values of "NaN" will be replaced with "0" or "None". The next process is data cleansing. There were as many as 78 training model data features and 78 data testing features within 329 columns. These 329 columns were obtained after per-processing the data using One-hot Encoding method in which every categorical data will be changed into several columns based on their categorical values. We used this method to ease us in the selection phase. In the cleansing step, we deleted columns with many "NaN" values or columns with only one cluster since it indicates the less important or we could say that they are outliers. First, we plotted data using a Scatter plot, then we delete the outliers. Figure 1 shows a correlation among features. the two features that influence the price of houses most are support vector machine and xgboost. Previous research has found that by performing clustering, outliers could be detected and removed. In this research, we found that there are two outliers.

4.7.1 Handling missing values Output

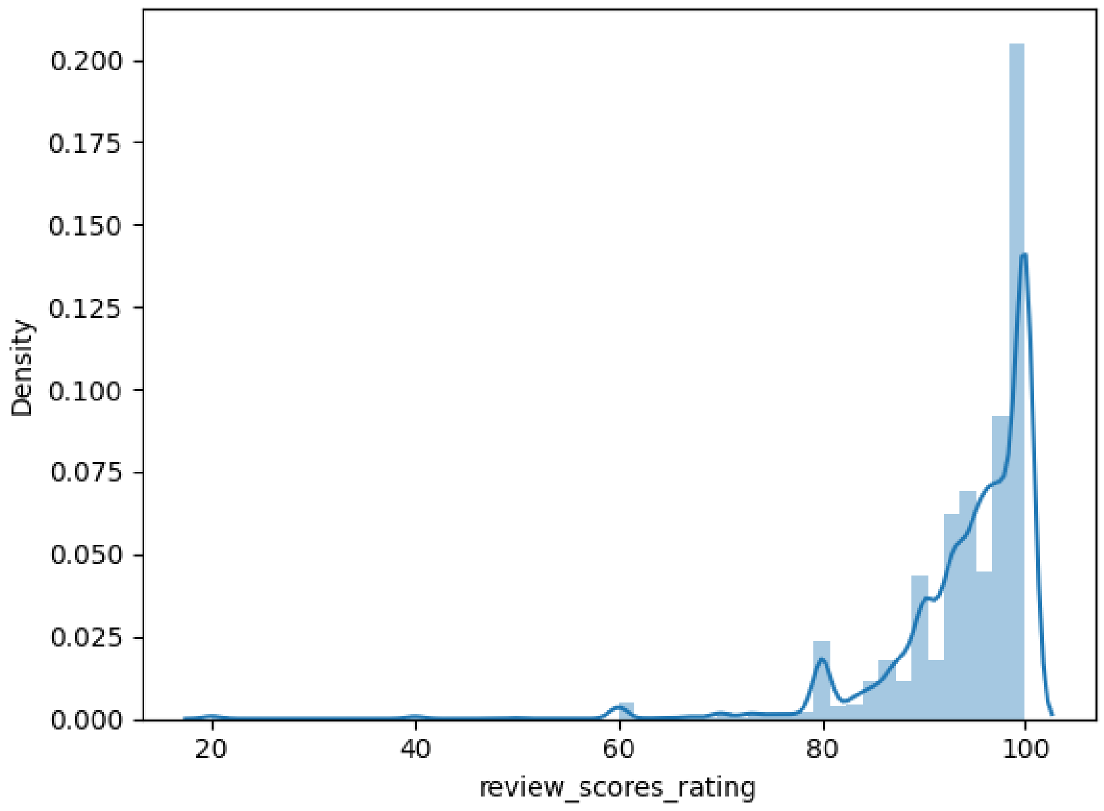


Figure 4.6 Graph for Handling Missing Values

A flowchart illustrating a land data processing system for classification. It starts with incorporating user data and a land dataset. This data is then subjected to a training phase, likely to train machine learning models to identify patterns. Following training, an intermediate representation containing the most important features for classification is generated. Next, the data is divided for testing and classification. There are three classification paths, likely utilizing different machine learning models (KNN Classifiers, XGBoost, CatBoost). Each path assesses its accuracy and contributes to a final "suspicion score" prediction. This suggests the system's goal is to predict the likelihood of a data point belonging to a particular class, potentially indicating risk or suitability based on the user's specifications.

4.7.2 Data Visualization output

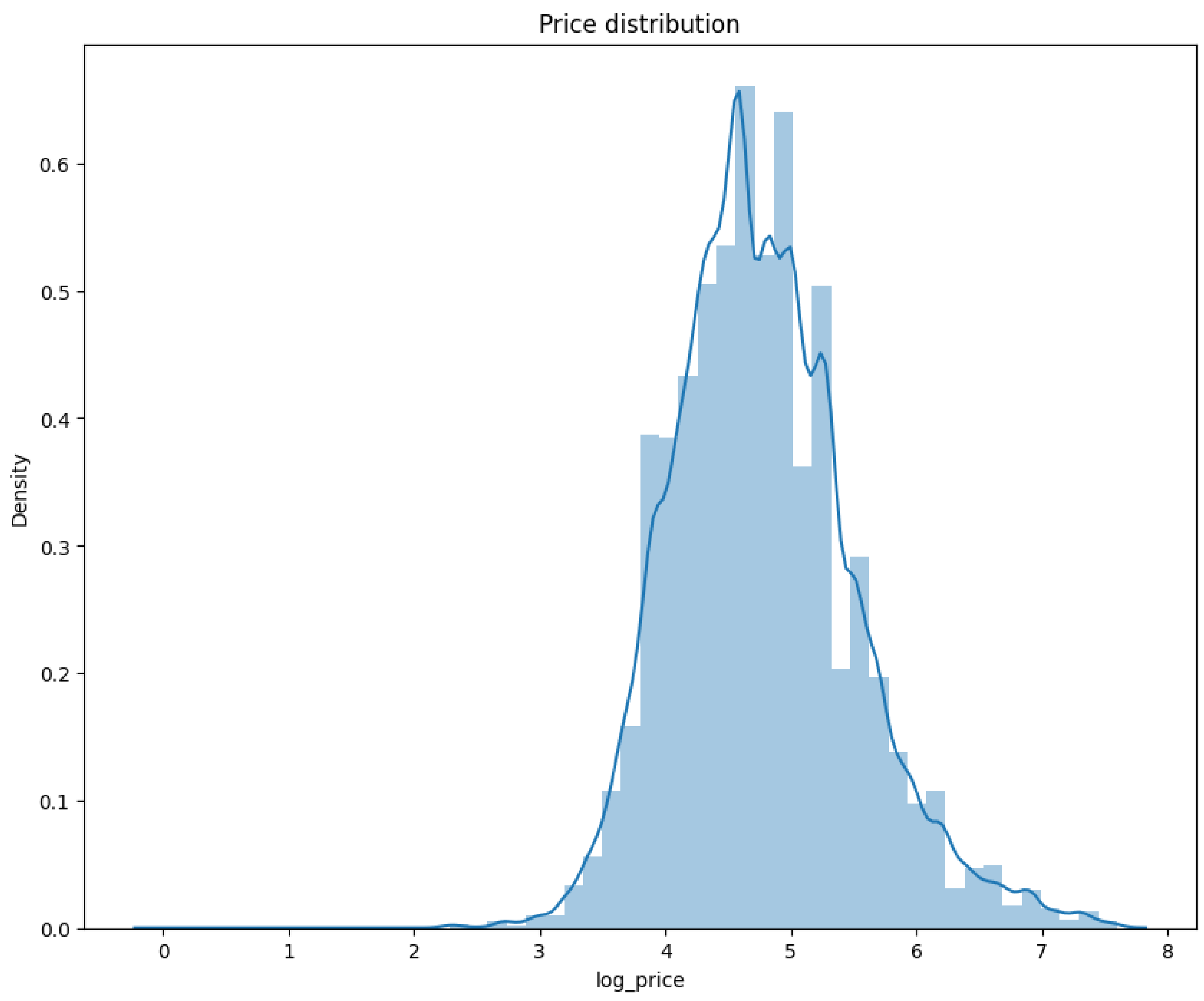


Figure 4.7 Data Visualization

The image shows a flowchart for classifying land data. User data and land details are input first. Machine learning models are likely trained on this data during a training stage. Then, a key feature summary is created. The data is then split for testing and classification. Multiple models analyze the data, and their performance is assessed. Finally, a "suspicion score" is generated, likely indicating the probability of a data point belonging to a specific class based on user input (e.g., risk or suitability).

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1 CONCLUSION

Our research has uncovered significant insights into property prediction methodologies, particularly through the application of a hybrid Genetic Algorithm (GA) with XGBoost. Our findings reveal that this hybrid approach outperforms sole XGBoost models in terms of Root Mean Squared Error (RMSE), processing times, and prediction accuracy. Notably, this improvement suggests the potential for enhanced prediction capabilities by integrating genetic algorithms with XGBoost.

Furthermore, our analysis identified two key features that substantially influence property prediction outcomes: XGBoost, which reflects the overall material and finish quality of the house, and support vector machine (SVM), representing the above-grade living area square footage. These features play pivotal roles in accurately predicting property values and highlight the importance of considering multiple variables in predictive modeling.

Comparison with three prior research studies further validates the effectiveness of our approach. Our research demonstrates superior performance in terms of error metrics compared to two of the previous studies, showcasing the robustness and reliability of our hybrid methodology. This suggests that our hybrid method holds promise for improving predictive accuracy and mitigating the risks associated with fake property predictions.

Moving forward, our findings suggest opportunities for further development and refinement of hybrid techniques for property prediction. By leveraging the strengths of genetic algorithms and XGBoost, we can continue to enhance the accuracy and efficiency of predictive models, ultimately aiding in the detection and prevention of fraudulent property transactions.

In conclusion, our research underscores the efficacy of hybrid methodologies in property prediction tasks. By identifying influential features and achieving superior performance compared to existing approaches, our study contributes valuable insights to the field of real estate analytics. These findings pave the way for the continued advancement of predictive modeling techniques and hold significant implications for addressing challenges related to fake property prediction.

5.2 FUTURE ENHANCEMENT

Future enhancements leveraging the insights from our research could focus on several key areas to further improve property prediction accuracy and address emerging challenges in the real estate domain. Incorporating additional features or engineering more sophisticated feature representations could enhance model performance. Exploring novel data sources, such as satellite imagery, social media sentiment analysis, or economic indicators, may provide valuable insights into property value determinants. Expanding beyond hybrid approaches, ensembling multiple models could leverage the strengths of diverse algorithms, further boosting predictive accuracy and robustness. Techniques like stacking or blending could integrate predictions from various models to generate more reliable estimations. Investigating the application of deep learning architectures, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), may capture complex patterns in property data more effectively. These models can automatically learn hierarchical representations from raw data, potentially uncovering hidden relationships that traditional methods might miss. Incorporating temporal dynamics into predictive models could account for changing market conditions and seasonality effects on property prices. Time-series analysis techniques could capture trends and patterns over time, enabling more accurate long-term predictions and adaptive model updates.

Developing user-friendly visualization tools could empower stakeholders, such as real estate agents, investors, and policymakers, to interactively explore and interpret prediction results. Interactive dashboards or geographic information system (GIS) interfaces could provide intuitive insights into property value trends and hotspots. Integrating external datasets, such as demographic information, infrastructure development plans, or environmental factors, could enrich predictive models and provide a more comprehensive understanding of property value drivers. Data fusion techniques could effectively combine diverse datasets to improve prediction accuracy. Ensuring fairness and transparency in predictive models is essential. Future enhancements should prioritize ethical considerations, such as avoiding biases and discriminatory outcomes, and promote accountability in model development and deployment. By pursuing these future enhancements, we can advance the state-of-the-art in property prediction, contribute to more informed decision-making in the real estate industry, and mitigate risks associated with fraudulent property transactions.

ANNEXURE

SAMPLE CODE

IMPORTING LIBRARIES AND DATASET LOADING

```
import warnings
warnings.simplefilter('ignore')

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn. Preprocessing input LabelEncoder. StandardScaler
from sklearn.model_selection import train_test_split,
GridSearchCV,cross_val_score, KFoldfrom
sklearn import metrics  from
sklearn.linear_model import  LinearRegression from
sklearn.preprocessing import PolynomialFeaturesfrom
sklearn.pipeline import Pipeline
from catboost importCatBoostRegressor
df = pd.read_csv('/kaggle/input/airbnb-price dataset/Airbnb_Data.csv')
df.head()
df.shape
df.columns
Index(['id', 'log_price', 'property_type', 'room_type', 'amenities', 'accommodates',
'bathrooms', 'bed_type', 'cancellation_policy', 'cleaning_fee', 'city', 'description',
'first_review', 'host_has_profile_pic', 'host_identity_verified', 'host_response_rate',
'host_since', 'instant_bookable', 'last_review', 'latitude', 'longitude', 'name',
'neighbourhood', 'number_of_reviews', 'review_scores_rating', 'thumbnail_url',
'zipcode', 'bedrooms', 'beds'],
dtype='object')
```

```
df.describe()
df.info()
df.dtypes
df.isnull().sum()
   df["zipcode"].unique
index= ["host_response_rate","property_type",
"room_type","accommodates","bathrooms","bed_type", "cancellation_policy",
"cleaning_fe "city","instant_bookable", "beds", "bedrooms",
"neighbourhood","first_review", "last_review","zipcode",
"name","host_since","thumbnail_url", "latitude", "longitude",
"host_has_profile_pic", "host_identity_verified"]

for i in index:
print(df[i].value_counts(), "\n")
print("        ---------------")
```

HANDLING MISSING VALUES

```
for column in df.columns:

if df[column].isnull().sum() != 0:
    print("=======================================")
print("\n{} :- {}, dtypes :
{}".format(column,df[column].isnull().sum(),df[column].dtypes))
df.last_review.fillna(method="ffill",inplace=True)
   df.first_review.fillna(method="ffill",inplace=True)
    df.host_since.fillna(method="ffill",inplace=
    sns.distplot(df["bathrooms"])
<Axes: xlabel='bathrooms', ylabel='Density'>
```

```
df["bathrooms"] = df['bathrooms'].fillna(round(df["bathrooms"].median()))
sns.distplot(df["review_scores_rating"])
<Axes: xlabel='review_scores_rating', ylabel='Density'>
df["review_scores_rating"] = df["review_scores_rating"].fillna(0)
sns.distplot(df["bedrooms"]) plt.show()
df["bedrooms"] = df['bedrooms'].fillna((df["bathrooms"].median()))
sns.distplot(df["beds"]) plt.show()
df["beds"] = df["beds"].fillna((df["bathrooms"].median()))
amenities_count = []
for i in df["amenities"]: amenities_count.append(len(i))
df["amenities"] = amenities_count
```

DATA VISUALISATION

```
# Function to plot catplot graphs def plot_catplot(h,v,he,a):
sns.set(font_scale=1.5) sns.catplot(x=h,kind=v,data=df,height=he, aspect = a)
# Function to plot catplot graphs def plot_piechart(h):
sns.set(font_scale=1.5)
fig = plt.figure(figsize=(5,5)) ax = fig.add_axes([0,0,1,1]) ax.axis('equal')
langs = list(df[h].unique()) students =list(df[h].value_counts())
ax.pie(students, labels = langs,autopct='%1.2f%%') plt.show()

plt.figure(figsize = (10, 8)) sns.distplot(df["log_price"]) plt.title('Price distribution')
plt.show()
plot_catplot("room_type", "count", 5, 2)
plot_piechart("room_type")
plot_catplot("city","count", 5, 2)
fig = plt.figure(figsize=(8,8)) ax = fig.add_axes([0,0,1,1]) ax.axis('equal')
langs = list(df.city.unique()) students =list(df.city.value_counts())
```

```
ax.pie(students, labels = langs,autopct='%1.2f%%') plt.show()
data = df.neighbourhood.value_counts()[:15] plt.figure(figsize=(22,22))
x = list(data.index) y = list(data.values) x.reverse() y.reverse()
plt.title("Most popular Neighbourhood") plt.ylabel("Neighbourhood Area")
plt.xlabel("Number of guest who host in this area")
plt.barh(x,y) plt.show()plot_catplot("cancellation_policy","count",10, 2)
plot_catplot("cleaning_fee","count",6,2)
def plot_violinplot(h,v): plt.figure(figsize=(15,8)) sns.set(font_scale=1.5)
sns.violinplot(data=df, x=h, y=v, palette='GnBu_d') plt.title('Density and distribution of
prices ', fontsize=15) plt.xlabel(h)
plt.ylabel(v)
plot_violinplot("city","log_price")
plot_violinplot("room_type","log_price")
plot_violinplot("cancellation_policy","log_price")
plot_violinplot("bed_type","log_price")
lot_catplot("bed_type","count",8, 2)
categorical_col = [] numerical_col = []
for column in df.columns:
if df[column].dtypes != "float64" and df[column].dtypes != "int64":
categorical_col.append(column)
else:
numerical_col.append(column)
numerical_col
['id', 'log_price', 'amenities', 'accommodates', 'bathrooms', 'latitude', 'longitude',
'number_of_reviews', 'review_scores_rating', 'bedrooms',
'beds']
Categorical_col
from sklearn.preprocessing import LabelEncoder le = LabelEncoder()
for col in categorical_col:
```

```
df[col] = le.fit_transform(df[col])
pd.set_option("display.max_columns",None) df
```

HEATMAP

```
plt.figure(figsize = (40,40))
sns.heatmap(df.corr(), annot=True, fmt=".2f", cmap="seismic") plt.show()
x                                                                          =
df.drop(["id","name","log_price","description","first_review","host_since","last_revie
w","neighbourhood", "thumbnail_url", "zipcode"],axis = 1)
y = df.log_price
  from sklearn.model_selection import train_test_split
  x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.2,random_state=101)
```

LINEAR REGRESSION

```
lr = LinearRegression()
lr.fit(x_train,y_train)
y_pred_lr = lr.predict(x_test)

mae_lr    =    metrics.mean_absolute_error(y_test,    y_pred_lr)    mse_lr    =
metrics.mean_squared_error(y_test, y_pred_lr)
rmse_lr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_lr))
r2_lr = metrics.r2_score(y_test, y_pred_lr)

print('\nRoot Mean Squarred Error of Linear Regression: ', rmse_lr)
print('\nR2 Score of Linear Regression: ', r2_lr)
```

RANDOM FOREST

```
rf = RandomForestRegressor()
```

```
rf.fit(x_train,y_train)
y_pred_rf = rf.predict(x_test)
mae_rf = metrics.mean_absolute_error(y_test, y_pred_rf) mse_rf = metrics.mean_squared_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(metrics.mean_squared_error(y_test, y_pred_rf)) r2_rf = metrics.r2_score(y_test, y_pred_rf)

print('\nMean Absolute Error of Random Forest Regressor : ', mae_rf) print('\nMean Squarred Error of Random Forest Regressor : ', mse_rf) print('\nRoot Mean Squarred Error of Random Forest Regressor: ', rmse_rf)
print('\nR2 Score of Random Forest Regressor : ', r2_rf)
```

POLYNOMIAL REGRESSION

```
from sklearn.linear_model import Ridge model = Pipeline([
('poly', PolynomialFeatures()), ('ridge', Ridge(fit_intercept=True))
])
   param_grid = {
'poly degree': [1, 2, 3],
'ridge alpha': [0.1, 0.5, 1.0, 2.0]
}
# Perform grid search with 5-fold cross-validation
poly_tuned = GridSearchCV(model, param_grid, cv=5
```

TRAINING AND TESTING

```
poly_tuned.fit(x_train, y_train)
y_pred_poly = poly_tuned.predict(x_test)
```

```
mae_poly = metrics.mean_absolute_error(y_test, y_pred_poly) mse_poly = metrics.mean_squared_error(y_test, y_pred_poly) rmse_poly = np.sqrt(metrics.mean_squared_error(y_test, y_pred_poly))
r2_poly = metrics.r2_score(y_test, y_pred_poly)
print('\nRoot Mean Squarred Error of Polynomial Regression: ', rmse_poly)
print('\nR2 Score of Polynomial Regression   : ', r2_poly)
```

CATBOOST

```
model_CBR = CatBoostRegressor()
model_CBR.fit(x_train, y_train)
cross_val_score(model_CBR, x_train, y_train,
scoring='r2', cv=KFold(n_splits=5,
shuffle=True, random_state=2022,
))
y_pred_cbr = model_CBR.predict(x_test)
mae_cbr = metrics.mean_absolute_error(y_test, y_pred_cbr) mse_cbr = metrics.mean_squared_error(y_test, y_pred_cbr)
rmse_cbr = np.sqrt(metrics.mean_squared_error(y_test, y_pred_cbr)) r2_cbr = metrics.r2_score(y_test, y_pred_cbr)
print('\nMean Absolute Error of CatBoost Regressor          :  ',  mae_cbr)  print('\nMean Squarred Error of CatBoost Regressor : ', mse_cbr) print('\nRoot Mean Squarred Error of CatBoost Regressor: ', rmse_cbr)
print('\nR2 Score of CatBoost Regressor        : ', r2_cbr)
```

GRADIENT BOOSTING

```
gb = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1, max_depth=3)
gb.fit(x_train, y_train)
```

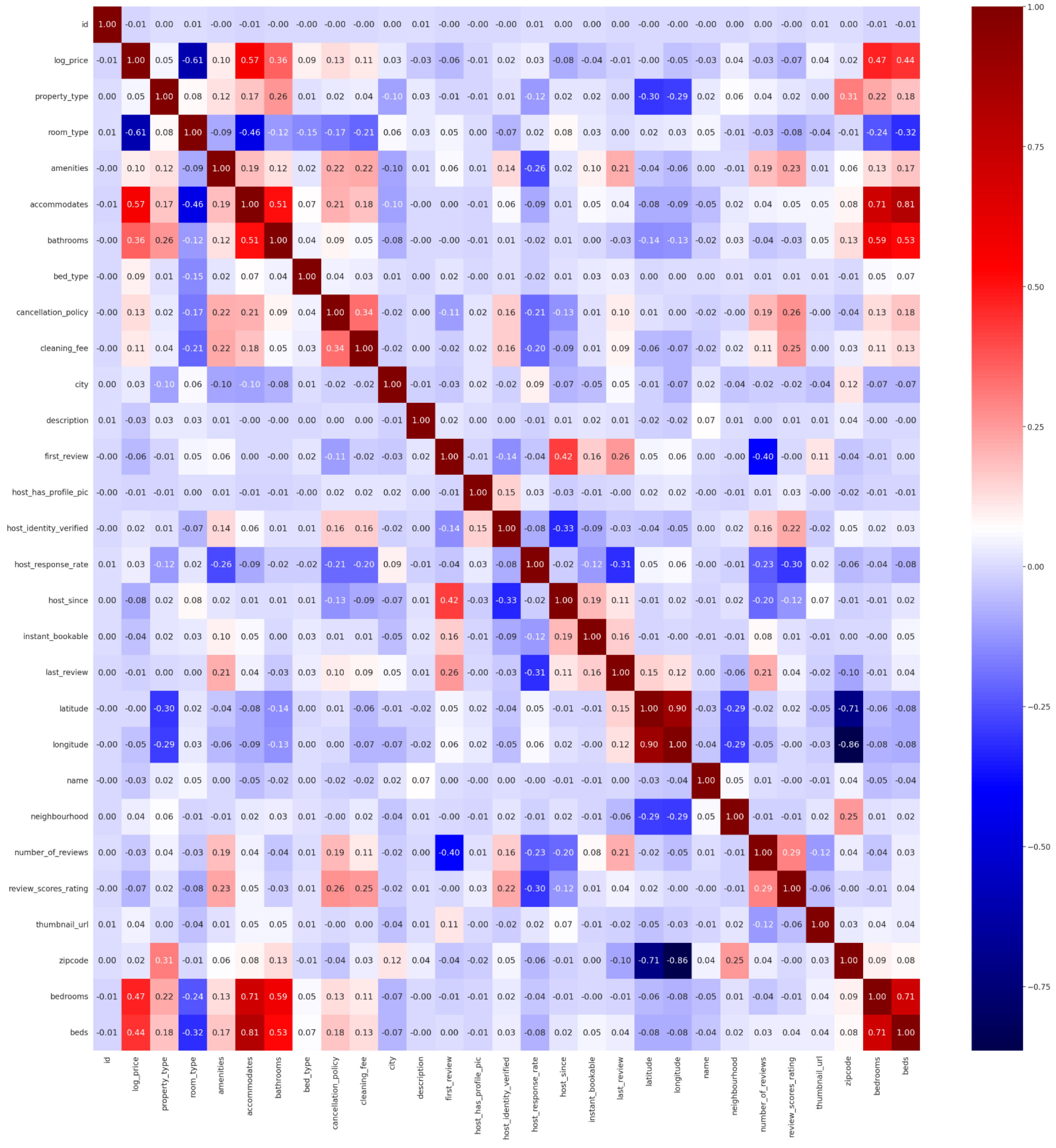
```
y_pred_gb = gb.predict(x_test)

mae_gb = metrics.mean_absolute_error(y_test, y_pred_gb) mse_gb =
metrics.mean_squared_error(y_test, y_pred_gb)
rmse_gb = np.sqrt(metrics.mean_squared_error(y_test, y_pred_gb)) r2_gb =
metrics.r2_score(y_test, y_pred_gb)

print('\nMean Absolute Error of Gradient Boosting : ', mae_gb) print('\nMean Squarred
Error of Gradient Boosting : ', mse_gb) print('\nRoot Mean Squarred Error of
Gradient Boosting: ', rmse_gb)
print('\nR2 Score of Gradient Boosting: ', r2_gb)
```

XGBOOST

```
xgb = XGBRegressor(objective='reg:squarederror')
xgb.fit(x_train, y_train)
```

5.3 SCREENSHOTS

5.3.1 Heatmap



	id	log_price	property_type	room_type	amenities	accommodates	bathrooms	bed_type	cancellation_policy	cleaning_fee	city	description	first_review	host_has_profile_pic	host_identity_verified	host_response_rate	host_since	instant_bookable	last_review	latitude	longitude	name	neighbourhood	number_of_reviews	review_scores_rating	thumbnail_url	zipcode	bedrooms	beds
id	1.00	-0.01	0.00	0.01	-0.00	-0.01	-0.00	-0.00	-0.00	-0.00	0.00	0.01	-0.00	-0.00	-0.00	0.01	0.00	0.00	0.00	-0.00	-0.00	-0.00	0.00	0.00	-0.00	0.01	0.00	-0.01	-0.01
log_price	-0.01	1.00	0.05	-0.61	0.10	0.57	0.36	0.09	0.13	0.11	0.03	-0.03	-0.06	-0.01	0.02	0.03	-0.08	-0.04	-0.01	-0.00	-0.05	-0.03	0.04	-0.03	-0.07	0.04	0.02	0.47	0.44
property_type	0.00	0.05	1.00	0.08	0.12	0.17	0.26	0.01	0.02	0.04	-0.10	0.03	-0.01	-0.01	0.01	-0.12	0.02	0.02	0.00	-0.30	-0.29	0.02	0.06	0.04	0.02	0.00	0.31	0.22	0.18
room_type	0.01	-0.61	0.08	1.00	-0.09	-0.46	-0.12	-0.15	-0.17	-0.21	0.06	0.03	0.05	0.00	-0.07	0.02	0.08	0.03	0.00	0.02	0.03	0.05	-0.01	-0.03	-0.08	-0.04	-0.01	-0.24	-0.32
amenities	-0.00	0.10	0.12	-0.09	1.00	0.19	0.12	0.02	0.22	0.22	-0.10	0.01	0.06	0.01	0.14	-0.26	0.02	0.10	0.21	-0.04	-0.06	0.00	-0.01	0.19	0.23	0.01	0.06	0.13	0.17
accommodates	-0.01	0.57	0.17	-0.46	0.19	1.00	0.51	0.07	0.21	0.18	-0.10	-0.00	0.00	-0.01	0.06	-0.09	0.01	0.05	0.04	-0.08	-0.09	-0.05	0.02	0.04	0.05	0.05	0.08	0.71	0.81
bathrooms	-0.00	0.36	0.26	-0.12	0.12	0.51	1.00	0.04	0.09	0.05	-0.08	-0.00	-0.00	-0.01	0.01	-0.02	0.01	0.00	-0.03	-0.14	-0.13	-0.02	0.03	-0.04	-0.03	0.05	0.13	0.59	0.53
bed_type	-0.00	0.09	0.01	-0.15	0.02	0.07	0.04	1.00	0.04	0.03	0.01	0.00	0.02	-0.00	0.01	-0.02	0.01	0.03	0.03	0.00	0.00	0.00	0.01	0.01	0.01	0.01	-0.01	0.05	0.07
cancellation_policy	-0.00	0.13	0.02	-0.17	0.22	0.21	0.09	0.04	1.00	0.34	-0.02	0.00	-0.11	0.02	0.16	-0.21	-0.13	0.01	0.10	0.01	0.00	-0.02	-0.00	0.19	0.26	-0.00	-0.04	0.13	0.18
cleaning_fee	-0.00	0.11	0.04	-0.21	0.22	0.18	0.05	0.03	0.34	1.00	-0.02	0.00	-0.02	0.02	0.16	-0.20	-0.09	0.01	0.09	-0.06	-0.07	-0.02	0.02	0.11	0.25	0.00	0.03	0.11	0.13
city	0.00	0.03	-0.10	0.06	-0.10	-0.10	-0.08	0.01	-0.02	-0.02	1.00	-0.01	-0.03	0.02	-0.02	0.09	-0.07	-0.05	0.05	-0.01	-0.07	0.02	-0.04	-0.02	-0.02	-0.04	0.12	-0.07	-0.07
description	0.01	-0.03	0.03	0.03	0.01	-0.00	-0.00	0.00	0.00	0.00	-0.01	1.00	0.02	0.00	0.00	-0.01	0.01	0.02	0.01	-0.02	-0.02	0.07	0.01	0.00	0.01	0.01	0.04	-0.00	-0.00
first_review	-0.00	-0.06	-0.01	0.05	0.06	0.00	-0.00	0.02	-0.11	-0.02	-0.03	0.02	1.00	-0.01	-0.14	-0.04	0.42	0.16	0.26	0.05	0.06	0.00	-0.00	-0.40	-0.00	0.11	-0.04	-0.01	0.00
host_has_profile_pic	-0.00	-0.01	-0.01	0.00	0.01	-0.01	-0.01	-0.00	0.02	0.02	0.02	0.00	-0.01	1.00	0.15	0.03	-0.03	-0.01	-0.00	0.02	0.02	-0.00	-0.01	0.01	0.03	-0.00	-0.02	-0.01	-0.01
host_identity_verified	-0.00	0.02	0.01	-0.07	0.14	0.06	0.01	0.01	0.16	0.16	-0.02	0.00	-0.14	0.15	1.00	-0.08	-0.33	-0.09	-0.03	-0.04	-0.05	0.00	0.02	0.16	0.22	-0.02	0.05	0.02	0.03
host_response_rate	0.01	0.03	-0.12	0.02	-0.26	-0.09	-0.02	-0.02	-0.21	-0.20	0.09	-0.01	-0.04	0.03	-0.08	1.00	-0.02	-0.12	-0.31	0.05	0.06	-0.00	-0.01	-0.23	-0.30	0.02	-0.06	-0.04	-0.08
host_since	0.00	-0.08	0.02	0.08	0.02	0.01	0.01	0.01	-0.13	-0.09	-0.07	0.01	0.42	-0.03	-0.33	-0.02	1.00	0.19	0.11	-0.01	0.02	-0.01	0.02	-0.20	-0.12	0.07	-0.01	-0.01	0.02
instant_bookable	0.00	-0.04	0.02	0.03	0.10	0.05	0.00	0.03	0.01	0.01	-0.05	0.02	0.16	-0.01	-0.09	-0.12	0.19	1.00	0.16	-0.01	-0.00	-0.01	-0.01	0.08	0.01	-0.01	0.00	-0.00	0.05
last_review	0.00	-0.01	0.00	0.00	0.21	0.04	-0.03	0.03	0.10	0.09	0.05	0.01	0.26	-0.00	-0.03	-0.31	0.11	0.16	1.00	0.15	0.12	0.00	-0.06	0.21	0.04	-0.02	-0.10	-0.01	0.04
latitude	-0.00	-0.00	-0.30	0.02	-0.04	-0.08	-0.14	0.00	0.01	-0.06	-0.01	-0.02	0.05	0.02	-0.04	0.05	-0.01	-0.01	0.15	1.00	0.90	-0.03	-0.29	-0.02	0.02	-0.05	-0.71	-0.06	-0.08
longitude	-0.00	-0.05	-0.29	0.03	-0.06	-0.09	-0.13	0.00	0.00	-0.07	-0.07	-0.02	0.06	0.02	-0.05	0.06	0.02	-0.00	0.12	0.90	1.00	-0.04	-0.29	-0.05	-0.00	-0.03	-0.86	-0.08	-0.08
name	-0.00	-0.03	0.02	0.05	0.00	-0.05	-0.02	0.00	-0.02	-0.02	0.02	0.07	0.00	-0.00	0.00	-0.00	-0.01	-0.01	0.00	-0.03	-0.04	1.00	0.05	0.01	-0.00	-0.01	0.04	-0.05	-0.04
neighbourhood	0.00	0.04	0.06	-0.01	-0.01	0.02	0.03	0.01	-0.00	0.02	-0.04	0.01	-0.00	-0.01	0.02	-0.01	0.02	-0.01	-0.06	-0.29	-0.29	0.05	1.00	-0.01	-0.01	0.02	0.25	0.01	0.02
number_of_reviews	0.00	-0.03	0.04	-0.03	0.19	0.04	-0.04	0.01	0.19	0.11	-0.02	0.00	-0.40	0.01	0.16	-0.23	-0.20	0.08	0.21	-0.02	-0.05	0.01	-0.01	1.00	0.29	-0.12	0.04	-0.04	0.03
review_scores_rating	-0.00	-0.07	0.02	-0.08	0.23	0.05	-0.03	0.01	0.26	0.25	-0.02	0.01	-0.00	0.03	0.22	-0.30	-0.12	0.01	0.04	0.02	-0.00	-0.00	-0.01	0.29	1.00	-0.06	-0.00	-0.01	0.04
thumbnail_url	0.01	0.04	0.00	-0.04	0.01	0.05	0.05	0.01	-0.00	0.00	-0.04	0.01	0.11	-0.00	-0.02	0.02	0.07	-0.01	-0.02	-0.05	-0.03	-0.01	0.02	-0.12	-0.06	1.00	0.03	0.04	0.04
zipcode	0.00	0.02	0.31	-0.01	0.06	0.08	0.13	-0.01	-0.04	0.03	0.12	0.04	-0.04	-0.02	0.05	-0.06	-0.01	0.00	-0.10	-0.71	-0.86	0.04	0.25	0.04	-0.00	0.03	1.00	0.09	0.08
bedrooms	-0.01	0.47	0.22	-0.24	0.13	0.71	0.59	0.05	0.13	0.11	-0.07	-0.00	-0.01	-0.01	0.02	-0.04	-0.01	-0.00	-0.01	-0.06	-0.08	-0.05	0.01	-0.04	-0.01	0.04	0.09	1.00	0.71
beds	-0.01	0.44	0.18	-0.32	0.17	0.81	0.53	0.07	0.18	0.13	-0.07	-0.00	0.00	-0.01	0.03	-0.08	0.02	0.05	0.04	-0.08	-0.08	-0.04	0.02	0.03	0.04	0.04	0.08	0.71	1.00

Figure 5.1 HeatMap

Figure 5.3.1 illustrates flowchart outlines a system for classifying land data. User requirements and raw land data are entered. Machine learning models are likely trained on this data to find patterns. A condensed version emphasizing key features is created next. The data is then split for testing and classification. Different models (KNN, XGBoost, CatBoost) analyze the data, and their accuracy is evaluated. Finally, combining info from all models, the system predicts a "Suspicion Score." This score signifies the likelihood of a data point having a specific characteristic important to the user's goals (e.g., flood risk, agricultural suitability).

5.3.2 Metrics Comparison

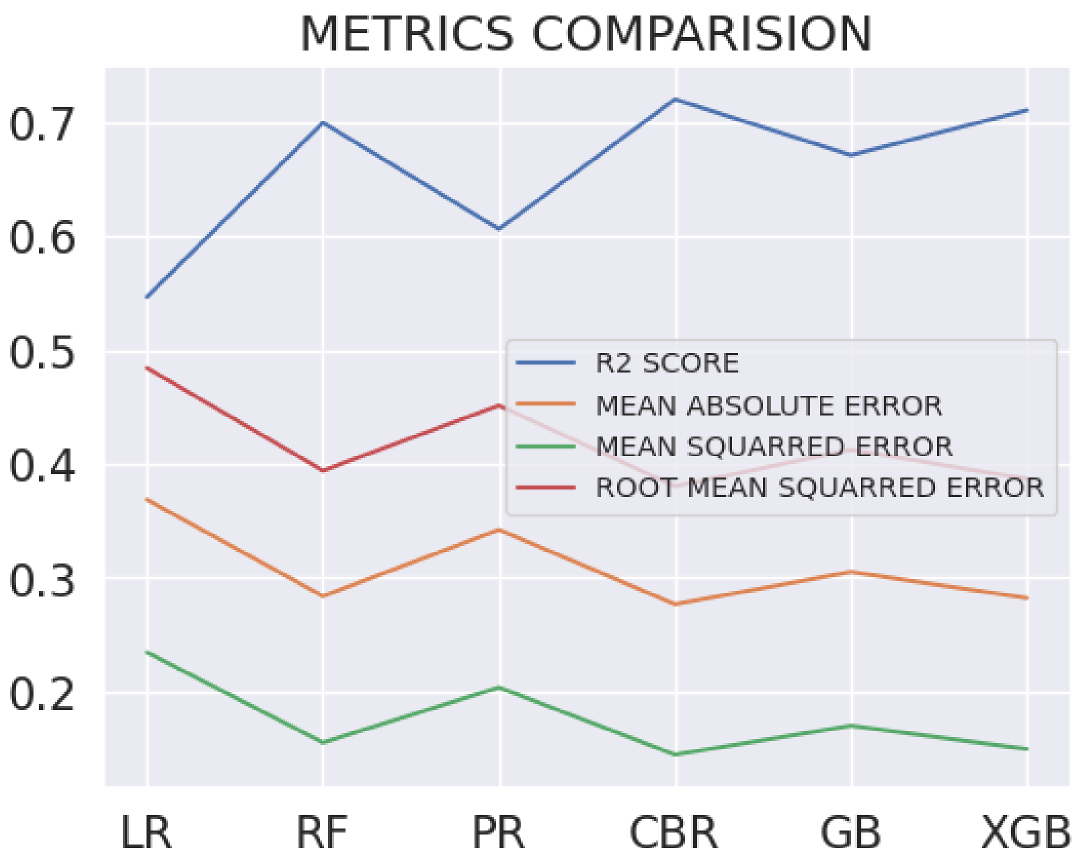


Figure 5.2 Metrics Comparison

Figure 5.3.2 flowchart outlines a land data processing system for classification. User-specified goals and raw land data are fed into the system. The data undergoes training, potentially using machine learning, to identify patterns relevant to the user's needs. A condensed version highlighting key features is then created. Next, the data is split for testing and classification. Multiple machine learning models analyze the data, and their individual performances are evaluated. Finally, by combining information from all models, the system generates a "Suspicion Score," indicating the likelihood of a data point belonging to a specific class relevant to the user's properties (e.g., risk, suitability).

REFERENCES

[1] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., ... & Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261.

[2] Bojchevski, A., & Günnemann, S. (2018). Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking. In International Conference on Learning Representations (ICLR).

[3] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., & Vandergheynst, P. (2017). Geometric deep learning: Going beyond Euclidean data. IEEE Signal Processing Magazine, 34(4), 18-42.

[4] Chen, J., Zhu, S., & Cao, K. (2019). FastGCN: Fast learning with graph convolutional networks via importance sampling. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 3197-3204).

[5] Dai, H., Khalil, E. B., Zhang, Y., Dilkina, B., & Song, L. (2018). Learning combinatorial optimization algorithms over graphs. In Advances in Neural Information Processing Systems (pp. 6348-6358).

[6] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., & Dahl, G. E. (2017). Neural message passing for quantum chemistry. arXiv preprint arXiv:1704.01212.
[7] 07. Hamilton, W. L., & Ying, Z. (2020). Inductive representation learning on large graphs. Journal of Machine Learning Research, 21(78), 1-10.

[8] Hamilton, W. L., Ying, R., & Leskovec, J. (2017). Inductive representation learning on large graphs. In Advances in neural information processing systems (pp. 1024-1034).

[9] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., & Leskovec, J. (2020). Open graph benchmark: Datasets for machine learning on graphs. arXiv preprint arXiv:2005.00687.

[10] Jin, W., & Gao, J. (2018). Junction tree variational autoencoder for molecular graph generation. In Advances in Neural Information Processing Systems (pp. 2010-2020).

[11] Kipf, T. N., & Welling, M. (2016). Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907.

[12] Klicpera, J., Groß, T., & Günnemann, S. (2019). Directional message passing for molecular graphs. In Advances in Neural Information Processing Systems (pp. 10165-10176).

[13] Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. (2015). Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.

[14] Maron, H., & Ben-Hamu, H. (2019). Provably powerful graph networks. In Advances in Neural Information Processing Systems (pp. 10321-10332).

[15] Monti, F., Boscaini, D., Masci, J., Rodolà, E., Svoboda, J., & Bronstein, M. M. (2017). Geometric deep learning on graphs and manifolds using mixture model CNNs. In Proceedings of the IEEE Conference on Computer Vision and Pattern

Recognition (pp. 5115-5124).

[16] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., ... & Grohe, M. (2019). Weisfeiler and Leman go neural: Higher-order graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 4602-4609).

[17] Qiu, J., Wang, X., Yao, H., & Ji, R. (2020). Graph convolutional networks based on graph Fourier transform. IEEE Transactions on Neural Networks and Learning Systems, 32(2), 932-946.

[18] Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., & Welling, M. (2018). Modeling relational data with graph convolutional networks. In European Semantic Web Conference (pp. 593-607).

[19] Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2018). Graph attention networks. arXiv preprint arXiv:1710.10903.

[20] Veličković, P., Cucurull, G., Casanova, A., Liò, P., & Bengio, Y. (2019). Graph attention networks in PyTorch. arXiv preprint arXiv:1906.01549.

[21] Verma, S.,Boyarski, A., & Lipman, Y. (2020). GraphRicciCurvature: A local feature extractor for graphs via tensor network convolution. arXiv preprint arXiv:2007.03623.

[22] Wang, X., Cheng, S., Han, J., & Leung, K. K. (2019). Relational graph convolutional networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 5606-5613).

[23] Wang, X., Cao, Q., Wang, P., & Cui, L. (2020). Graph convolutional networks meet Markov random fields: Semi-supervised community detection in attribute graphs. arXiv preprint arXiv:2009.03218.
[24] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., & Philip, S. Y. (2020). A comprehensive survey on graph neural networks. IEEE Transactions on Neural Networks and Learning Systems, 32(1), 4-24.

[25] Xie, T., & Grossman, J. C. (2018). Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. Physical Review Letters, 120(14), 145301

[26] Xu, K., Hu, W., Leskovec, J., & Jegelka, S. (2018). How powerful are graph neural networks? In International Conference on Learning Representations (ICLR).

[27] Ying, R., He, R., Chen, K., Eksombatchai, C., Hamilton, W. L., & Leskovec, J. (2018). Graph convolutional neural networks for web-scale recommender systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 974-983).

[28] Zeng, W., Tan, K. H., Qiao, Y., & Duan, L. (2020). Dynamic graph convolutional networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 34, No. 04, pp. 4087-4094).

[29] Zhang, M., Cui, Z., Neumann, M., & Chen, Y. (2018). An end-to-end deep learning architecture for graph classification. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).

[30] Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., & Sun, M. (2018). Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434.