CONTENTS

ABSTRACT

Banks derive a significant portion of their profits from loans, yet identifying reliable applicants who will repay loans poses a challenge. Manual loan processing is prone to errors and misconceptions, leading to difficulties in selecting genuine applicants. To address this, a machine learning loan prediction system is being developed, automating the selection of eligible candidates. This system benefits bank staff and applicants by streamlining the loan approval process. The system accurately predicts loan outcomes using machine learning algorithms such as Support Vector Machine (SVM). This automation drastically reduces the time required for loan approval, benefiting both banks and applicants.

Key points: Loan Profitability and Applicant Reliability, Manual Processing Issues, Automation Solutions, Machine Learning Algorithms, Efficiency, and Mutual Benefits.

INTRODUCTION

Loan approval systems in the financial sector evaluate creditworthiness using various criteria. The integration of automated systems and machine learning has revolutionized lending operations. Automated systems expedite decision-making, reducing processing times significantly.

Machine learning enhances accuracy by analyzing diverse data points for more reliable lending decisions. These advancements greatly improve risk management, equipping lenders with sophisticated tools to evaluate and mitigate potential risks.

Consequently, these innovations lead to more efficient lending operations and improved outcomes for both lenders and borrowers. Additionally, automated systems and machine learning algorithms are continually updated with new data, allowing them to adapt to changing market conditions and borrower behaviors.

This adaptability ensures that the loan approval process remains robust and relevant, further enhancing the efficiency and reliability of financial operations. The seamless integration of these technologies not only streamlines operations but also fosters greater trust and transparency between lenders and borrowers, ultimately contributing to a more stable and dynamic financial ecosystem.

LITERATURE SURVEY

Research paper-1: "Prediction of Loan Approval in Banks Using Machine Learning Approach"

- Improve loan approval processes in banks using machine learning techniques.
- Utilized four machine learning algorithms - Random Forest, Naive Bayes, Decision Tree, and KNN.
- Achieved an accuracy rate of 83.73%, with Naive Bayes performing the best.
- Examined borrower characteristics such as marital status and income to determine loan eligibility.
- Enhanced prediction accuracy for loan approvals, reduced processing times, and minimized risk of borrower defaults.

Limitations:

- Naive Bayes can be biased and Random Forest or KNN may overfit, reducing effectiveness on new data.
- Random Forest and KNN lack interpretability, complicating their use in decision-sensitive areas.

Reference:

https://www.researchgate.net/publication/372909313_Prediction_of_Loan_Approval_in_Banks_using_Machine_Learning_Approach

Research paper-2: "THE LOAN PREDICTION USING MACHINE LEARNING“

- Automate the loan approval process in banks using a machine learning model to efficiently and accurately identify eligible loan applicants.
- Implementation of a Decision Tree algorithm, trained on historical data, to predict the likelihood of applicants repaying loans based on various criteria.
- The system automates verification and eligibility checks, reducing loan sanctioning time and minimizing human errors, thereby ensuring a more reliable approval process.

Limitations:

- Prediction accuracy depends on high-quality, relevant training data.
- Decision Trees risk overfitting with noisy data, affecting reliability.
- The model may not perform well in unique or diverse banking situations without specific training.

Reference:

https://www.researchgate.net/publication/357449126_THE_LOAN_PREDICTION_USING_MACHINE_LEARNING

REQUIREMENT SPECIFICATIONS

Functional Requirements:

Data Collection:

- **Sources:** Gather data from financial institutions, credit bureaus, and customer records.
- **Storage:** Store data in a secure, structured database for regular updates and easy access.

Data Preprocessing:

- **Cleaning:** Handle missing values and outliers appropriately.
- **Normalization:** Normalize numerical features for consistent scaling.
- **Feature Engineering:** Create new features that may impact loan approval decisions.
- **Encoding:** Convert categorical data into numerical formats suitable for machine learning algorithms.

Model Development:

- **Algorithms:** Support SVM (Support Vector Machine), logistic regression, and decision trees.
- **Evaluation Metrics:** Evaluate models using accuracy, precision, recall, F1-score, and ROC-AUC metrics suitable for binary classification.

Model Training and Evaluation:

- **Data Splitting:** Divide data into training, validation, and testing sets.
- **Training:** Train models using SVM and other selected algorithms.
- **Evaluation:** Provide reports on model performance metrics to assess accuracy and reliability.

Model Deployment:

- **Deployment:** Deploy the trained model as a web service accessible via API.
- **Real-time Prediction:** Allow real-time predictions based on user input for loan applications.
- **Scalability:** Ensure the system can handle multiple concurrent requests efficiently.

Monitoring and Maintenance:

- **Performance Monitoring:** Continuously monitor model performance and update with new data.
- **Logging:** Log user interactions and predictions for auditing and analysis purposes.
- **Maintenance:** Provide mechanisms for regular maintenance and updates to the model and system infrastructure.

Software Requirements:

- **Operating System:** Compatible with Windows 10, macOS, or Linux.
- **Programming Language:** Python for development and implementation.
- **Development Environment:** Jupyter Notebook for interactive development and scikit-learn for machine learning tasks.
- **Libraries:** Pandas for data manipulation, NumPy for numerical computations, Matplotlib and Seaborn for data visualization, and scikit-learn for implementing SVM and other machine learning algorithms.

PROBLEM STATEMENT

Loan approval processes in banks are vital for profitability, yet identifying reliable borrowers is challenging, leading to errors and delays. Manual evaluations often fail to distinguish genuine applicants from potential defaulters, impacting financial health and inconveniencing applicants. To address this, we intend to develop a genuine loan applicant prediction powered by machine learning, specifically a Support Vector Machine (SVM), for predicting loan outcomes. This solution automates evaluation, enabling swift identification of eligible candidates. By inputting applicant data, banks can receive instant predictions, reducing processing time and biases. This benefits both banks and applicants by streamlining approval processes and enhancing decision-making accuracy

Fig: Loan Approval Prediction

PROPOSED SOLUTION

The proposed solution focuses on leveraging customer behavior data to create a comprehensive dataset for analysis. By collecting and utilizing these records, machine learning models can be trained to predict the approval status of loan applications. This approach offers several advantages, including the ability to quickly assess a customer's eligibility based on predefined algorithmic criteria, while minimizing the risk of overfitting due to the model's robustness against minor data changes. However, the system also faces challenges, particularly concerning privacy. Safeguarding the extensive customer records used for training is crucial to prevent breaches that could compromise confidentiality and erode trust. Mitigating these risks is essential to ensure the ethical and secure deployment of predictive models in financial decision-making processes.

4.BLOCK DIAGRAM

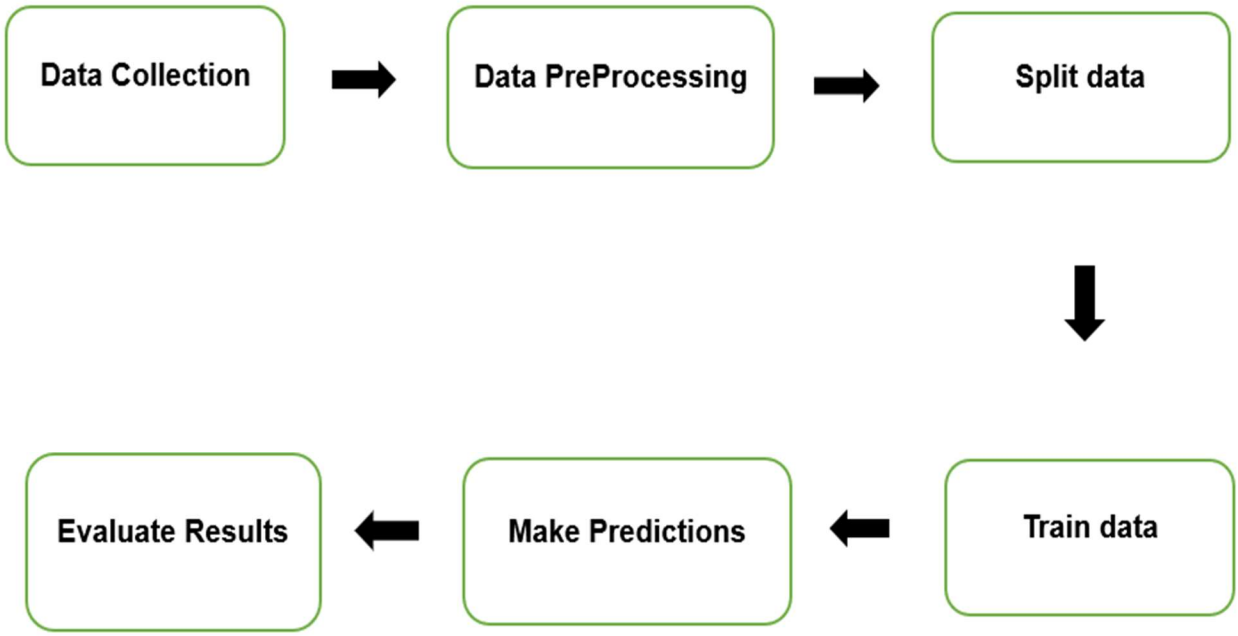


Fig.4.1. Block diagram

Starting with data collection and preprocessing ensures that the dataset is clean and prepared for analysis. By splitting the data into training and testing sets, the model can be effectively trained and evaluated. Making predictions based on this trained model enables efficient decision-making regarding loan applications. This structured approach from data collection to evaluation forms a robust framework for leveraging machine learning in financial decision processes.

Implementation

#import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import classification_report, accuracy_score, precision_score, recall_score, f1_score, confusion_matrix, roc_curve, auc
```

#load and inspect data

```
df=pd.read_excel("C:\Users\Harini\OneDrive\Desktop\LoanDataModified@1.xlsx")
```

#First 10 rows

```
df.head(10)
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome(lakhs)	CoapplicantIncome(lakhs)	LoanAmount_in_lakhs	Loan_Amount_Term
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0

#Last 10 rows

df.tail(10)

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome(lakhs)	CoapplicantIncome(lakhs)	LoanAmount_in_lakhs	Loan_Amount_Term
607	LP002964	Male	Yes	2	Not Graduate	No	3987	1411.0	157.0	360.0
608	LP002974	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0
614	LP002991	Male	YES	3+	Graduate	Yes	8075	498.0	158.0	180.0
615	LP002992	Female	YES	3+	Graduate	Yes	5789	2501.0	159.0	360.0
616	LP002993	Male	YES	3+	Graduate	Yes	9964	1000.0	160.0	180.0

df.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 617 entries, 0 to 616
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Loan_ID                    617 non-null    object
 1   Gender                     604 non-null    object
 2   Married                    614 non-null    object
 3   Dependents                 602 non-null    object
 4   Education                  617 non-null    object
 5   Self_Employed              585 non-null    object
 6   ApplicantIncome(lakhs)     617 non-null    int64
 7   CoapplicantIncome(lakhs)   617 non-null    float64
 8   LoanAmount_in_lakhs        595 non-null    float64
 9   Loan_Amount_Term           603 non-null    float64
 10  Credit_History             567 non-null    float64
 11  Property_Area              617 non-null    object
 12  Loan_Status                617 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.8+ KB
```

df.isnull().sum()

```
Loan_ID                    0
Gender                    13
Married                    3
Dependents                15
Education                  0
Self_Employed             32
ApplicantIncome(lakhs)     0
CoapplicantIncome(lakhs)   0
LoanAmount_in_lakhs       22
Loan_Amount_Term          14
Credit_History            50
Property_Area              0
Loan_Status                0
dtype: int64
```

#handle missing values

df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['LoanAmount_in_lakhs'].fillna(df['LoanAmount_in_lakhs'].mean(), inplace=True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
df.isnull().sum()

```
Loan_ID                   0
Gender                    0
Married                   0
Dependents                0
Education                 0
Self_Employed             0
ApplicantIncome(lakhs)    0
CoapplicantIncome(lakhs)  0
LoanAmount_in_lakhs       0
Loan_Amount_Term          0
Credit_History            0
Property_Area             0
Loan_Status               0
dtype: int64
```

#Engineering new features

```
df['loanAmount_log'] = np.log(df['LoanAmount_in_lakhs'])
df['TotalIncome'] = df['ApplicantIncome(lakhs)'] + df['CoapplicantIncome(lakhs)']
df['TotalIncome_log'] = np.log(df['TotalIncome'])
#converting categorical features into strings
categorical_features = ['Gender', 'Married', 'Dependents', 'Education', 'Self_Employed', 'Property_Area']
df[categorical_features] = df[categorical_features].astype(str)
```

#Define features and target.

```
X = df.drop(['Loan_Status', 'Loan_ID'], axis=1)
X
```

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome(lakhs)	CoapplicantIncome(lakhs)	LoanAmount_in_lakhs	Loan_Amount_Term
0	Male	No	0	Graduate	No	5849	0.0	146.47563	360.0
1	Male	Yes	1	Graduate	No	4583	1508.0	128.00000	360.0
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.00000	360.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.00000	360.0
4	Male	No	0	Graduate	No	6000	0.0	141.00000	360.0

```
y = df['Loan_Status']
y
```

```
0      Y
1      N
2      Y
3      Y
4      Y
      ..
612    Y
613    N
614    Y
615    Y
616    N
Name: Loan_Status, Length: 617, dtype: object
```

#Split data into training and testing sets.

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

#Balance the classes using SMOTE.

```
numerical_features = ['ApplicantIncome(lakhs)', 'CoapplicantIncome(lakhs)',
'LoanAmount_in_lakhs', 'Loan_Amount_Term', 'Credit_History', 'loanAmount_log',
'TotalIncome', 'TotalIncome_log']

preprocessor = ColumnTransformer(
    transformers=[
        ('num', StandardScaler(), numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

#Balance the classes using SMOTE.
pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('classifier', SVC(probability=True))
])


#Set up and perform grid search to find the best hyperparameters.
param_grid = {
    'classifier__C': [0.1, 1, 10, 100],
    'classifier__kernel': ['linear', 'rbf'],
    'classifier__gamma': ['scale', 'auto', 0.001, 0.01, 0.1, 1]
}

#Train the model and make predictions.
grid_search = GridSearchCV(estimator=pipeline, param_grid=param_grid, cv=5,
scoring='accuracy', n_jobs=-1)
grid_search.fit(x_train, y_train)
```

best_model = grid_search.best_estimator_
y_pred = best_model.predict(x_test)
y_probs = best_model.predict_proba(x_test)[:, 1]

#Evaluate model performance using various metrics.
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
print("Best model parameters:", grid_search.best_params_)

```
Best model parameters: {'classifier__C': 1, 'classifier__gamma': 'scale', 'classifier__kernel': 'linear'}
```

print("Accuracy on test data:", accuracy)

```
Accuracy on test data: 0.8536585365853658
```

print("Confusion Matrix:\n", conf_matrix)

```
Confusion Matrix:
 [[21 17]
 [ 1 84]]
```

print("Accuracy:", accuracy)

```
Accuracy on test data: 0.8536585365853658
```

```
print("Precision:", precision_score(y_test, y_pred, pos_label='Y'))
```

```
Precision: 0.8316831683168316
```

```
print("Recall:", recall_score(y_test, y_pred, pos_label='Y'))
```

```
Recall: 0.9882352941176471
```

```
print("F1 Score:", f1_score(y_test, y_pred, pos_label='Y'))
```

```
F1 Score: 0.903225806451613
```

```
print(classification_report(y_test, y_pred))
```

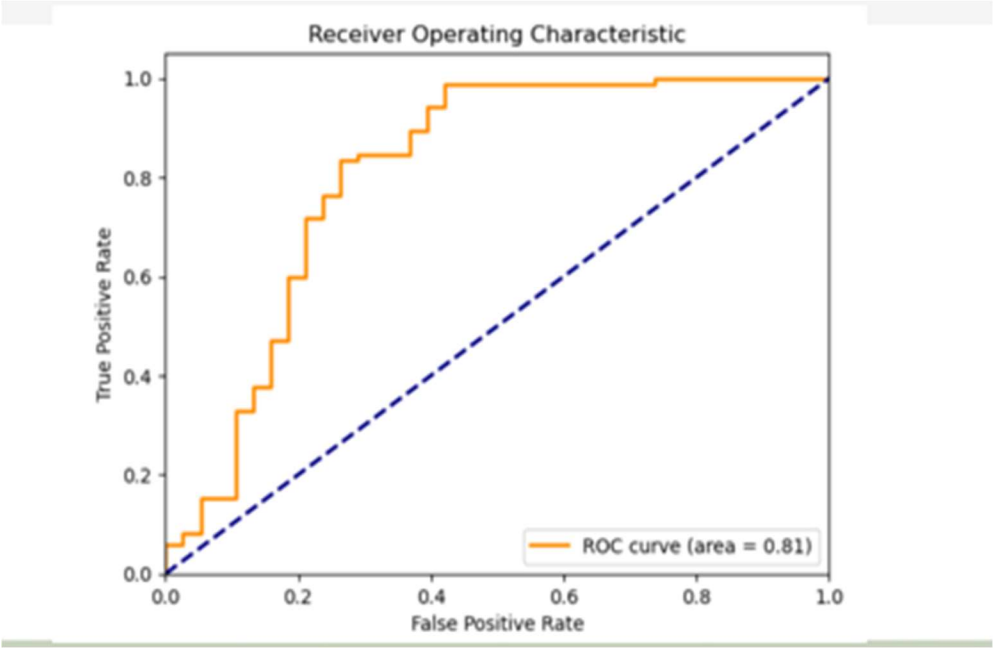
```
              precision    recall  f1-score   support

           N       0.95      0.55      0.70        38
           Y       0.83      0.99      0.90        85

    accuracy                           0.85       123
   macro avg       0.89      0.77      0.80       123
weighted avg       0.87      0.85      0.84       123
```

#Plot the ROC curve to visualize model performance.

```
fpr, tpr, thresholds = roc_curve(y_test.map({'N': 0, 'Y': 1}), y_probs)  # Convert labels to 0 and 1 for ROC
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.81)

RESULT AND DISCUSSION

1. Data Loading and Preprocessing:

- The data has been successfully loaded and preprocessed.
- Columns have been renamed for clarity, enhancing the dataset's usability.

2. Handling Missing Values:

- Missing values in key columns have been addressed using statistical methods:
- Mode for categorical variables.
- Mean for continuous variables.
- This approach is crucial for maintaining data integrity and avoiding errors during model training.

3. Feature Engineering:

- Log transformation of the LoanAmount_in_lakhs has been implemented.
- This transformation helps normalize the data distribution, potentially improving the performance of various machine-learning algorithms.

4. Data Preparation:

- Renaming and cleaning data columns and handling missing values are fundamental steps for reliable machine-learning analysis.

5. Dataset Separation:

- The dataset has been separated into attributes (x) and the label (y).
- This separation sets the stage for the next steps in the machine learning pipeline, including splitting the data into training and testing sets.

6. Model Optimization and Evaluation:

- The SVM model for loan approval prediction was optimized using grid search with cross-validation.
- Applying the best model parameters to the test data resulted in an accuracy of 85%.

7. Confusion Matrix:

- True Positives (TP): 21
- True Negatives (TN): 84
- False Positives (FP): 17
- False Negatives (FN): 1

8. Performance Metrics:

- Precision: 98%
- Recall: 83%
- F1 Score: 90%
- High precision indicates a strong ability to correctly identify loan approvals.
- The F1 score reflects a good balance between precision and recall.
- These results affirm the SVM model's effectiveness in predicting loan approvals, providing a reliable tool for financial institutions.

CONCLUSION AND FUTURE SCOPE

- A machine learning-based loan approval system, leveraging the Support Vector Machine (SVM) algorithm, simplifies evaluation for financial institutions.
- SVM analyzes historical loan data and applicant attributes to categorize loan candidates efficiently.
- Reduces processing time and minimizes default risk by accurately assessing borrower creditworthiness.
- Represents a move towards enhanced efficiency and fairness in loan approval processes. Facilitates improved access to capital, fostering economic growth and opportunity for individuals and businesses.
- The machine learning-based loan approval system, leveraging the Support Vector Machine (SVM) algorithm, simplifies the evaluation process for financial institutions. By automating the decision-making process, it significantly reduces the workload on human underwriters, allowing them to focus on more complex cases.

Future Scope:

- The future scope of the loan approval prediction project includes advanced algorithms, real-time data processing, and integration with financial analytics and blockchain for enhanced security.
- It aims to offer personalized loan products, and dynamic interest rates, and utilize alternative and global data for better assessments.
- This project can automate regulatory compliance, address biases, and improve user experience with intuitive interfaces and mobile apps.
- Ongoing model monitoring, scalable infrastructure, partnerships with credit bureaus, and financial literacy collaborations will enhance accuracy, inclusivity, and user-friendliness, benefiting both financial institutions and borrowers.

REFERENCES

ResearchPaper1:https://www.ijert.org/comparison-of-machine-learning-algorithms-for-house-price-prediction-using-real-time-data

ResearchPaper2:https://www.researchgate.net/publication/371602053_Machine_Learning_Approach_for_House_Price_Prediction/link/655471abb86a1d521be6b2a9/download?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6InB1YmxpY2F0aW9uIiwicGFnZSI6InB1YmxpY2F0aW9uIn19

ResearchPaper3:https://www.irjmets.com/uploadedfiles/paper//issue_3_march_2023/35094/final/fin_irjmets1680263791.pdf

Kaggle :(https://www.kaggle.com/)