Marketing application for Real Estate management using Machine Learning Algorithms

Amrutha S
CSE Department
PES University
Bangalore, India
amruthashetty299@gmail.com

Ria Kalia
CSE Department
PES University
Bangalore, India
riakalia@gmail.com

Abstract

The real estate industry is one of the most recognized global sectors. The growth of this sector is complemented by the right strategy and planning, which involves sales and marketing. Extensive insight into what attributes defines a property needs to be gathered, and then analyzed. Doing this manually and then coming up with a marketing strategy is strenuous and time-consuming. Through this project, we have classified properties based on several features and have predicted whether it needs marketing or not. Our main focus here is on marketing. We collected data by scraping it from a website to obtain the basic feature set. We then cleaned this dataset based on the required feature set and processed it. We used exploratory analysis to attain clear insights regarding the data obtained. Since this is a classification problem, we made use of the most efficient classification algorithm, Support Vector Machines. We have also implemented models of other algorithms to find the most efficient algorithm for our dataset. Therefore, we are predicting if a property requires marketing or not and it's displayed on a Linux GUI, PyQT5.

I. INTRODUCTION

A look into real estate sales data from 2016 onwards reveals that the real estate industry is going through a massive downturn. Although there has been an increase in the number of properties, the sale of these properties is going down month by month. Well known real estate developers have become defaulters since they took loans of huge amounts of money and have been unable to pay it back.
The graph below depicts a rise in the number of properties in the real estate sector. However, there's a huge gap in demand and supply. The supply is in excess of the demand. We aim to bridge this gap by helping developers determine which properties they need to focus on, and thereby improve their sales.

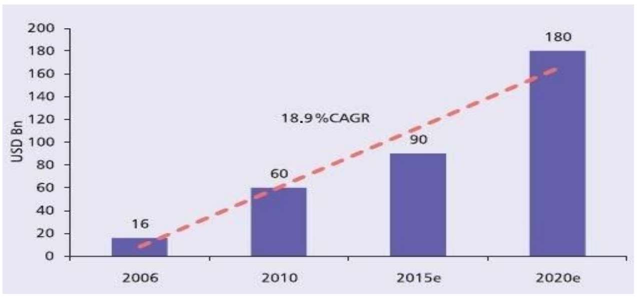


Fig 1: Growth in the number of properties through the years

The Reserve Bank in its article on economic trends in India showed that the per capita income is steadily rising to $5000/annum and growing. GDP is growing at a steady rate of close to 8%, and inflation is stable at around 50%. This clearly indicates that there's a growing middle-class population with a disposable income that could be invested in home buying.

In this paper, we propose an automated marketing strategy handler that can take decisions on which property needs to be marketed. We also tried to implement decisions based on the locality of the property, but due to several constraints in the dataset and time, it has been cut down to the selection of property for marketing, based on several features.

In order to run this application, we had to find historical data relevant to the requirements of our algorithm. The main challenge was in the collection of data since most websites have privacy concerns with respect to scraping data. Although we managed to collect data through web-scraping techniques, extracting the right information from this data was a tough job.

In this research, we found out the attributes that play an important role in deciding the sale of a property, which in turn tells which property requires marketing. This was done using several exploratory analysis techniques and data visualization techniques. We then made a few inferences which we used to classify and rank the variations in attributes.

Since our dataset had many features, we implemented dimension reduction on it, so that the number of attributes would fit our machine learning model. We used concepts such as covariance, eigenvalues, and eigenvectors and reduced the feature set by applying Principal Component Analysis on it. We chose Principal Component Analysis since it uses orthogonal transformations to convert correlated values into linearly independent variables called principal components. All of this is done with minimal loss of information.

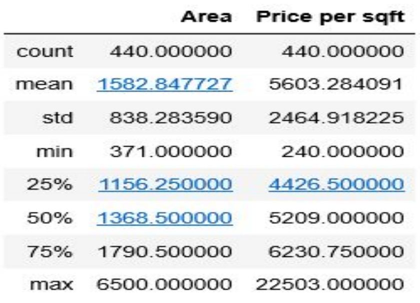
We experimented on the data after dimension reduction by using various machine learning models such as Support Vector Machine, K-Nearest Neighbors algorithm and Naive Bayes algorithm. We then compared their performance metrics and concluded that Support Vector Machines is the best classification algorithm for our dataset. After the selection of the model, for demonstration purposes, we have created a user interface for entering property details and displaying the prediction.

This paper is organized as follows: Section II details the design and methodology used. Section III explains the comparison between the machine learning models we have used. Section IV includes the implementation details. Section V includes details about demonstration and results. Finally, Section VI includes the technologies used. Section VII includes future works and Section VIII contains the conclusion.

II. DESIGN AND METHODOLOGY

A. *Collecting Data*

We obtained the dataset by scraping data from www.makaan.com. The attributes we got after scraping were property name, price, area, price per square feet, location, posting details and posting description.



	Area	Price per sqft
count	440.000000	440.000000
mean	1582.847727	5603.284091
std	838.283590	2464.918225
min	371.000000	240.000000
25%	1156.250000	4426.500000
50%	1368.500000	5209.000000
75%	1790.500000	6230.750000
max	6500.000000	22503.000000

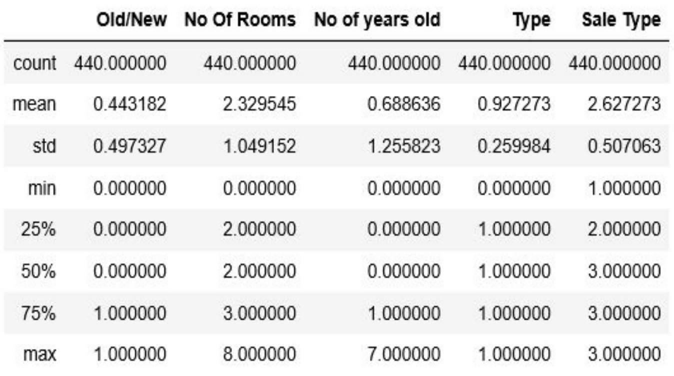
Fig 2: Data.describe( )

B. *Processing Data*

Since we scraped data from a website, it was present in an unorganized manner. However, it did not have any missing attributes and required no data cleaning to be performed on it. Data processing was required since the posting details and description was unorganized and vaguely provided. After data processing the final attributes were :

- **Property Name:** Contains the name of the property.
- **Price:** Contains the price of the property.
- **Area:** Contains the area of the property in terms of square feet.
- **Price per square feet:** Contains the price of the property per square feet.
- **Location:** Contains the location of the property.
- **Posting Details**: Contains details regarding how old the property is. The data is in an unorganized manner.
- **Posting Description:** Contains details about the features of the property such as the number of rooms, carpet area etc. The data is in an unorganized manner.
- **Old/New:** Depicts if the property is old or new.
- **No of rooms:** Depicts the number of rooms in the property.
- **No of years:** Depicts how old the property is.
- **Type:** Depicts if the property is a plot or a house.
- **Sale Type:** Depicts if the property is being sold, resold or is not possessed.

The extra columns that have been added after data processing are described as follows:



	Old/New	No Of Rooms	No of years old	Type	Sale Type
count	440.000000	440.000000	440.000000	440.000000	440.000000
mean	0.443182	2.329545	0.688636	0.927273	2.627273
std	0.497327	1.049152	1.255823	0.259984	0.507063
min	0.000000	0.000000	0.000000	0.000000	1.000000
25%	0.000000	2.000000	0.000000	1.000000	2.000000
50%	0.000000	2.000000	0.000000	1.000000	3.000000
75%	1.000000	3.000000	1.000000	1.000000	3.000000
max	1.000000	8.000000	7.000000	1.000000	3.000000

Fig 3: Data.describe( )

C. *Exploratory Data Analysis*

Since the dataset obtained did not have any classification factors, we performed several data analytics techniques to rank and classify properties based on the features mentioned above.

.

We first plotted a density plot of all the attributes of the dataset to see the distribution of those attributes. This gave us insight into how the attributes were distributed for the properties.

Given below is the density plot for the area of the properties.

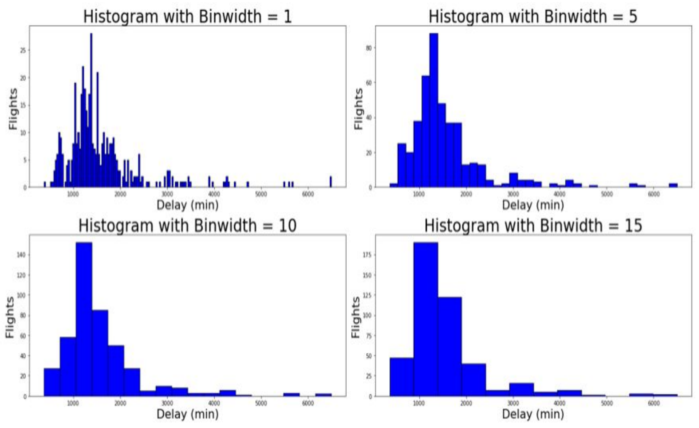
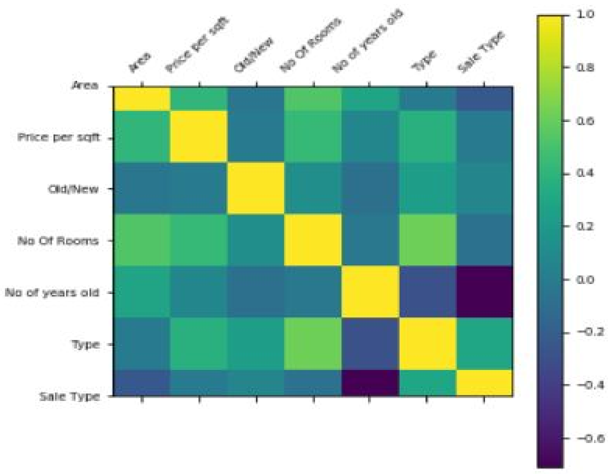
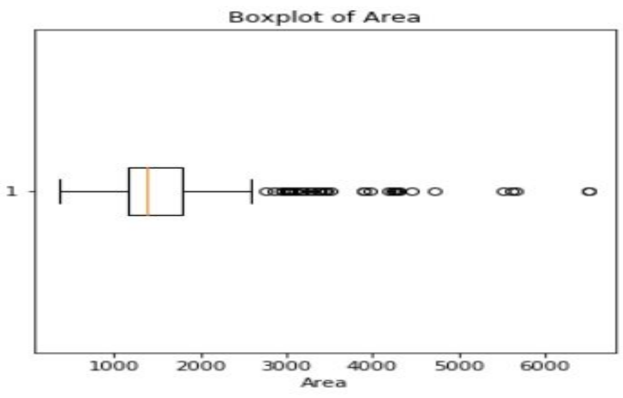


Fig 4: Density graph for Area

We plotted a correlogram to understand the correlation between the various attributes. From the figure given below it is apparent that the age of a property has a high degree of correlation with its sale type. The,price per square feet also has a good amount of correlation with area, number of rooms and the property type.



Fi 5: Correlation matrix



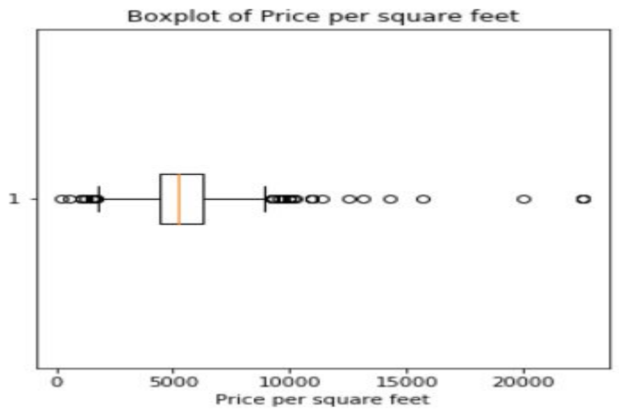


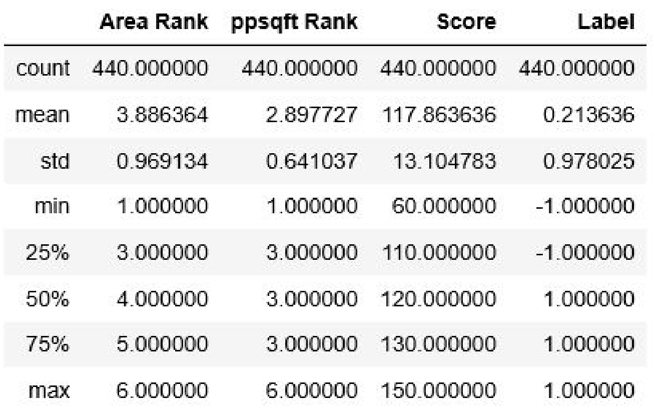
Fig 6: Box Plot for Area (above) and Price per square feet (below)

We plotted a box plot for area and price per square feet with the intention of ranking it based on its range. From the boxplots above, we considered the ranges of the attributes and ranked them. Using the analysis we performed, our next step was to infer the classification label.
We then computed the sum of all ranks to obtain a score for each property and determined the label.

Thus, our final dataset was ready with these extra attributes:

- **Area Rank:** Depicts the rank based on the area range it lies in.
- **Price Per square feet Rank:** Depicts the rank based on the price per square feet range it lies in.
- **Score:** Depicts the computed sum of all ranks.
- **Label:** Depicts the label, i.e. if the property needs to be marketed or not.

The final dataset has been appended with these features.

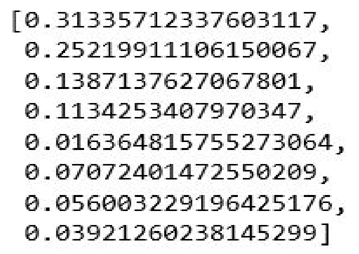


	Area Rank	ppsqft Rank	Score	Label
count	440.000000	440.000000	440.000000	440.000000
mean	3.886364	2.897727	117.863636	0.213636
std	0.969134	0.641037	13.104783	0.978025
min	1.000000	1.000000	60.000000	-1.000000
25%	3.000000	3.000000	110.000000	-1.000000
50%	4.000000	3.000000	120.000000	1.000000
75%	5.000000	3.000000	130.000000	1.000000
max	6.000000	6.000000	150.000000	1.000000

Fig 7: Data.describe( )

D. Principal Component Analysis (PCA)

After all the data processing and analysis was performed, categorical data was removed, leaving behind 8 features. We then performed dimension reduction on our dataset to reduce the number of features, so that it would fit our machine learning models.
We used Principal Component analysis for dimension reduction. We did this by first standardizing the data and computing the eigenvalues and eigenvectors for all the attributes. We then sorted these eigenvalues in descending order and computed their variance. From this we concluded that some of the attributes had very less significance and that they could be dropped with minimal loss of data.



```
[0.31335712337603117,
 0.25219911106150067,
 0.1387137627067801,
 0.1134253407970347,
 0.016364815755273064,
 0.07072401472550209,
 0.056003229196425176,
 0.03921260238145299]
```

Fig 8: Significance values

To correctly determine the number of columns that the dataset could be reduced to, we plotted a graph of explained variance versus the number of components (Fig 9). From this graph we could conclude that having five or six principal components would not lead to much information loss. We then constructed the projection matrix of the required components and transformed the dataset into a six-dimensional feature subspace.
Therefore, we performed Principal Component Analysis and reduced our dataset.
We preferred PCA over Singular Value Decomposition and various other dimensionality reduction techniques since the latter involves feature dropping, which leads to loss of information that might have been relevant to the model.

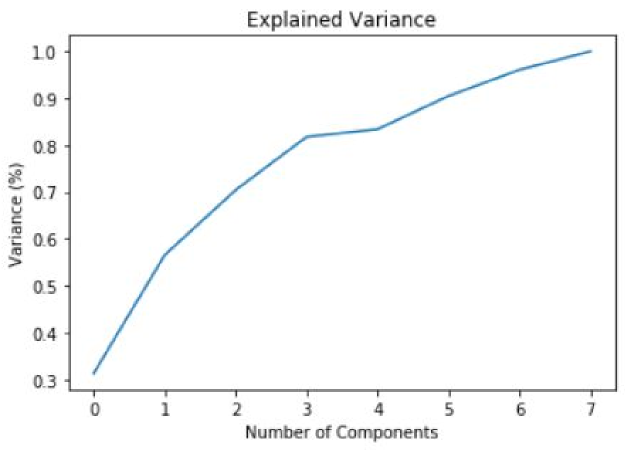


Fig 9: Explained Variance Graph

E. Machine Learning Models

To obtain the most optimal algorithm, we implemented different machine learning models from scratch. The models we implemented and compared are Support Vector Machine, K Nearest Neighbors and Naive Bayes. We compared these models through their performance metrics and came to the conclusion that Support Vector Machines is the most optimal algorithm for our model.

1. **Support Vector Machine:** A Support Vector Machine (SVM) is a supervised algorithm that can be used for classification as well as regression purposes. SVM is based on the idea of finding the hyperplane that best divides the dataset into two classes. Fig 10 depicts how our model classified the data into two classes.

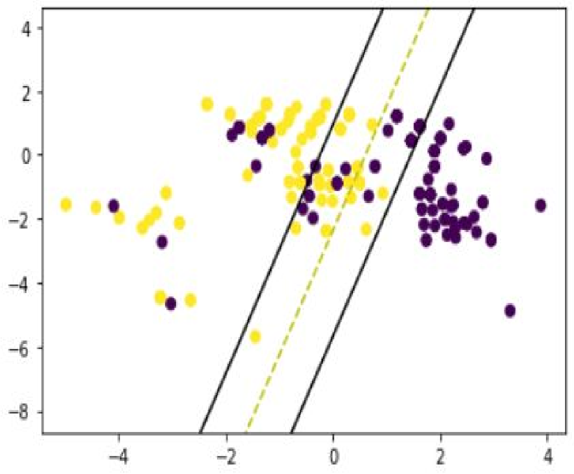
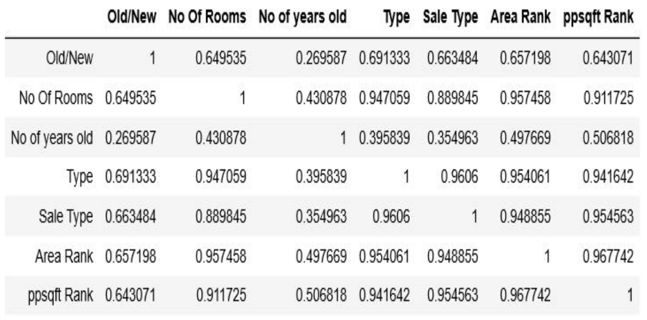


Fig 10: SVM Visualization

2. **K Nearest Neighbors:** It is a supervised algorithm that can be used to solve both classification and regression problems. It assumes that similar things exist in close proximity to each other. It calculates the Euclidean distance between the data points and classifies them based on similarity. We also used cosine distance to calculate the distance between the points. Fig 11 shows the similarity matrix between the attributes.



	Old/New	No Of Rooms	No of years old	Type	Sale Type	Area Rank	ppsqft Rank
Old/New	1	0.649535	0.269587	0.691333	0.663484	0.657198	0.643071
No Of Rooms	0.649535	1	0.430878	0.947059	0.889845	0.957458	0.911725
No of years old	0.269587	0.430878	1	0.395839	0.354963	0.497669	0.506818
Type	0.691333	0.947059	0.395839	1	0.9606	0.954061	0.941642
Sale Type	0.663484	0.889845	0.354963	0.9606	1	0.948855	0.954563
Area Rank	0.657198	0.957458	0.497669	0.954061	0.948855	1	0.967742
ppsqft Rank	0.643071	0.911725	0.506818	0.941642	0.954563	0.967742	1

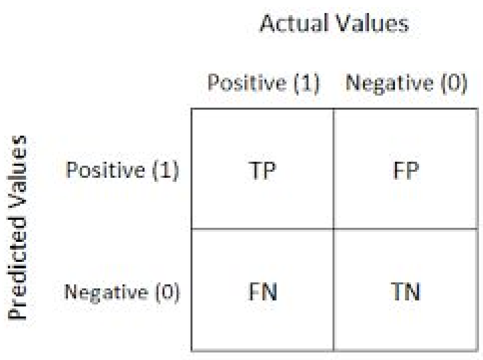
Fig 11: Similarity matrix (cosine angles)

3. **Naive Bayes Algorithm:** It is a classification technique based on Bayes Theorem. It assumes that predictors are independent. This means that the presence of a particular feature is unrelated to the presence of another. We have calculated the mean, variance and prior and posterior probability and returned the class with the highest posterior probability.

III. PERFORMANCE COMPARISON

As explained above, we have tested the dataset on three different models and compared the performance metrics and ROC graph to choose the best algorithm.

Terms used:



True Positives (TP): True positive refers to the cases where the actual class of the data point was True and the predicted value is also True. This means that the model has made the right prediction.

True Negatives (TN): True negative refers to the cases where the actual class of the data point was False and the predicted value is also False. This means that the model has made the right prediction.

False Positives (FP): False positive refers to the cases where the actual class of the data point was False and the predicted value is True. This means that the model has incorrectly classified a negative example as a positive one.

False Negatives (FN): False negative refers to the cases where the actual class of the data point was True and the predicted value is False. This means that the model has incorrectly classified a positive example as a negative one.

Performance metrics used

1. Accuracy: In classification problems, accuracy is the number of correct predictions made by the model over all kinds predictions made.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

2. Precision: It.expresses the proportion of the data points that our model says are relevant, and the ones that actually are relevant.

$$\text{Precision} = \frac{TP}{TP + FP}$$

3. Recall: It's the ability of a classification model to identify all relevant instances.

$$\text{Recall} = \frac{TP}{TP + FN}$$

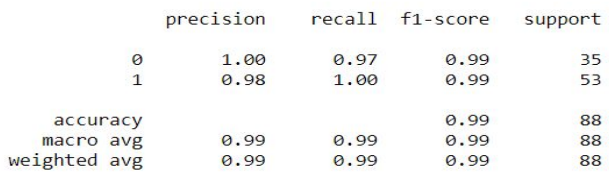
4. F1-score: It's the harmonic mean of precision and recall.

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

5. ROC graph: It shows how the relationship between recall and precision changes as the threshold is varied for identifying a positive example in our model. The threshold represents the value above which a data point is considered in the positive class. It is created by plotting the true positive rate (TPR) (or recall) against the false positive rate (FPR).

A. Support Vector Machine

The performance metrics when the dataset is run on the SVM model is as follows.



```
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        35
           1       0.98      1.00      0.99        53

    accuracy                           0.99        88
   macro avg       0.99      0.99      0.99        88
weighted avg       0.99      0.99      0.99        88
```

Fig 12: Performance metrics for SVM

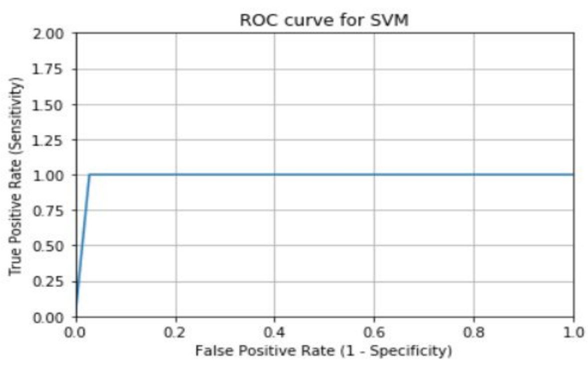
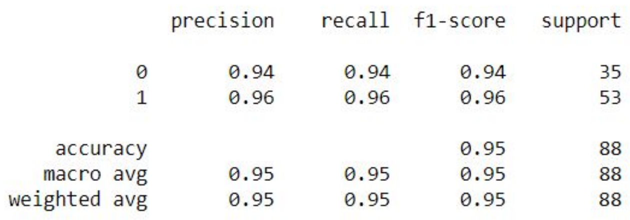


Fig 13: ROC Graph for SVM

The accuracy achieved though SVM is 98.86%

B. K Nearest Neighbors

The performance metrics when the dataset is run on KNN is as follows



```
              precision    recall  f1-score   support

           0       0.94      0.94      0.94        35
           1       0.96      0.96      0.96        53

    accuracy                           0.95        88
   macro avg       0.95      0.95      0.95        88
weighted avg       0.95      0.95      0.95        88
```

Fig 14: Performance metrics for KNN

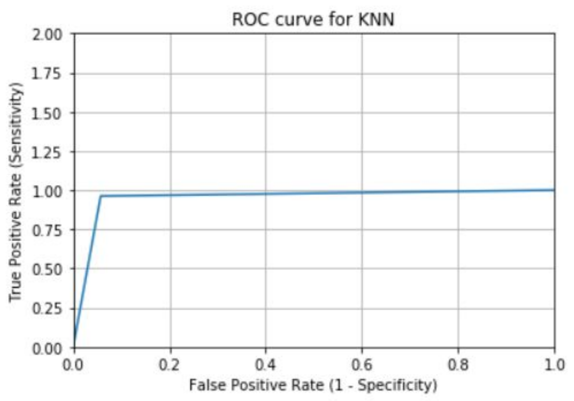
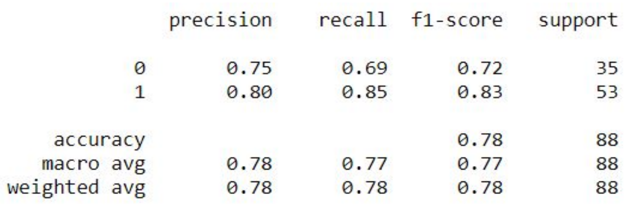


Fig 15: ROC Graph for KNN

The accuracy achieved though KNN is 95.45%

C. Naive Bayes Algorithm

The performance metrics when the dataset is run on Naive Bayes is as follows



```
              precision    recall  f1-score   support

           0       0.75      0.69      0.72        35
           1       0.80      0.85      0.83        53

    accuracy                           0.78        88
   macro avg       0.78      0.77      0.77        88
weighted avg       0.78      0.78      0.78        88
```

Fig 16: Performance metrics for Naive Bayes

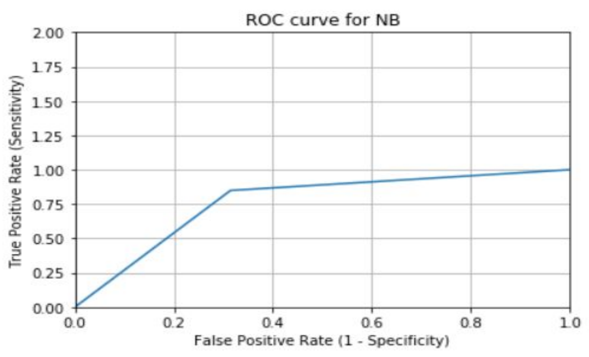


Fig 17: ROC Graph for Naive Bayes

The accuracy achieved though NB is 78 %

Final Prediction

From the above comparison, it is apparent that SVM has the highest accuracy, and is therefore the model that we chose for the further development of the application.

IV. IMPLEMENTATION

The data was scraped from www.makaan.com using a chrome extension called Agenty. All data related processes such as data cleaning, processing and analysis was done in Python on Jupyter Notebook. Principal component analysis and machine learning algorithms were coded from scratch with maximum avoidance of the use of the inbuilt sklearn library. For performing principal component analysis, we generated eigenvalues and eigenvectors, which is used to reduce the data into a lower dimensional space. The support vector machine algorithm uses a linear kernel and gradient descent was used for weight updation. The SVM hyperplane is plotted with the maximum possible margin. Likewise, K Nearest Neighbors was coded using Euclidean distance as the distance factor for classification. As a support, we also created a similarity matrix taking cosine distance into consideration. For the Naive Bayes Classifier, we used basic numpy functions to calculate the mean, variance, prior and posterior probability.

We then created the confusion matrix and displayed the accuracy, specificity, sensitivity and precision which helped us in the comparison of the models. For the graphic user interface, we used PythonQT5, which is a python friendly user interface, and hence works well with our project. When the user enters the property details, the details are processed into the format of the dataset and are appended to the dataset. The dataset is then tested and the output is hardcoded and displayed in the UI. We have imported and used os libraries for the smooth running of multiple python programs for the purpose of demonstration. Figure 18 depicts the workflow.

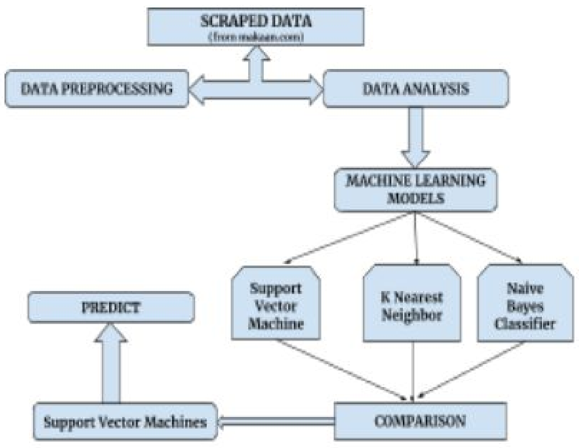


Fig 18: Workflow

V. DEMONSTRATION AND RESULTS

The demonstration of our project can be divided into three parts: The first part is the initial data pre-processing and the building of the dataset. The second is the analysis of the data, which includes various visualizations that helped us understand and work better with the dataset. The third and the final part is the implementation of the working model by entering property details into a form developed using PyQt5. The data entered is passed and the respective predictions are made. The result produced is limited to deciding if a property requires marketing or not and does not cover the other marketing details, which comes under future works.

VI. TECHNOLOGIES USED

The technologies we have used are:

- **Chrome extension - Agenty:** For the scraping of data.
- **Python in Jupyter Notebook:** A major part of this project has been performed on Jupyter Notebook. It has been used for pre-processing and analyzing data, and then to perform PCA and deploy the machine learning models. The modules of python that we have used are numpy, pandas, sklearn, matplotlib and scipy.
- **PyQT5:** It's a user interface that supports python and also supports CSV operations, which makes it very efficient for our project.

VII. FUTURE WORK

We plan on adding the following features to this application in the future:

- Provide marketing strategies such as TV advertisements, hoardings and newspaper ads to the user, to assist them on the best ways to market their property.
- Improve on the accuracy of our machine learning model by using deep learning techniques like SVR and neural networks.

VIII. CONCLUSION

Through research and implementation of the aforementioned algorithms, we have been able to deduce that real estate data management and strategy development is a strenuous job. The collection of data and generation of the right dataset for this purpose was a great challenge. It involved web scraping, data cleaning, preprocessing and analysis of the data. After applying machine learning algorithms on this data, we have achieved a working model that predicts which property requires marketing and which does not.

Acknowledgment

We would like to sincerely thank the Computer Science Department of PES University for giving us the opportunity to work on this project. We would like to thank our for guide Prof.K.S.Srinivas for guiding us through every stage of this research project. Without his support it would have been difficult for us to achieve this. Through this project we have learnt a lot about data scraping, preprocessing, analytics and gained in depth knowledge about various machine learning algorithms.

References

[1] Quigley, John M. "Real estate prices and economic cycles." (2002).

[2] Schölkopf, Bernhard, Alexander Smola, and Klaus-Robert Müller. "Kernel principal component analysis." In International conference on artificial neural networks, pp. 583-588. Springer, Berlin, Heidelberg, 1997.

[3] Suykens, Johan AK, and Joos Vandewalle. "Least squares support vector machine classifiers." Neural processing letters 9, no. 3 (1999): 293-300.

[4] Peterson, Leif E. "K-nearest neighbor." Scholarpedia 4, no. 2 (2009): 1883.

[5] Rish, Irina. "An empirical study of the naive Bayes classifier." In IJCAI 2001 workshop on empirical methods in artificial intelligence, vol. 3, no. 22, pp. 41-46. 2001.