PROJECT REPORT

On

Pune House price prediction

➢ Abstract :

The Pune House Price Prediction project uses data analysis and computer programs to guess how much houses in Pune cost. It helps people who want to buy or sell houses by guessing how much a house could sell for. We use a computer language called R to make a guess, and we look at things like how many bedrooms, balconies, bathrooms, how big the house is, and where it's located to make our guess. This way, people can have a better idea of what a house might be worth.

The project uses a step-by-step approach. First, we gather dataset from Kaggle, including past house sale records in Pune. We make sure this data is good by cleaning it, which means we remove any missing or wrong information. We use special tools in R, like 'dplyr', 'tidyr', 'glmnet', 'caret' and 'randomForest' and 'shiny' to work with the data. We also look at the data to see how things like the number of bedrooms, balconies, bathrooms, the house's size, and where it's located relate to house prices.

We have used multiple machine learning models for training, testing and fitting and multiple linear regression is found to be most suitable amongst them with an accuracy rate of 78 %.The result on the custom data are displayed on user friendly GUI using shiny package in R after model's deployment.

➢ Kaggle dataset link-:

https://www.kaggle.com/datasets/saipavansaketh/pune-house-data

➢ Project code -:

```
setwd("C:/Users/rajvk/OneDrive/Documents/r language")

#Data preprocessing and cleaning----------------------------------------------------------------------------

library(shiny)
library(shinydashboard)
library(dplyr)
library(tidyr)
library(ggplot2)
library(caret)
library(glmnet)
library(boot)
library(rpart)
library(e1071)
library(randomForest)
library(plotly)

#data cleaning and preprocessing----------------------------------------------------------------------------

f=read.csv("punehouseprediction.csv")
```

```
View(f)
print(dim(f))

area_type_count = table(f$area_type)
print(area_type_count)
f1 = f[, !colnames(f) %in% "availability"]
f2 = f1[, !colnames(f1) %in% "area_type"]
f3 = f2[, !colnames(f2) %in% "society"]
cat("\n")
missing_counts=colSums(is.na(f3))
print(missing_counts)
f4= na.omit(f3)
unique_BHK_values=unique(f4$size)
print(unique_BHK_values) #printing unique values
f4 = f4 %>%mutate(size = as.integer(sapply(strsplit(size, " "), "[[", 1)))
bhkgreaterthan10_f4=subset(f4, size > 10)
print(bhkgreaterthan10_f4)
unique_sqft=unique(f$total_sqft)
print(unique_sqft)
is_float = function(x)
{
  result = tryCatch({
    as.numeric(x)
  }, error = function(e) {
    NA
  })
  !is.na(result)
}
filtered_sq_ft_f5 = f4[!f4$total_sqft %>% sapply(is_float), ]
convert_sqft_to_num = function(x) {
tokens = unlist(strsplit(x, " - "))
 if (length(tokens) == 2) {
   lower_bound = as.numeric(tokens[1])
    upper_bound = as.numeric(tokens[2])
    if (!any(is.na(c(lower_bound, upper_bound)))) {
    return((lower_bound + upper_bound) / 2)
    }
  }

  numeric_value = as.numeric(x)

  if (!is.na(numeric_value)) {
    return(numeric_value)
  }

  cat("Unable to convert:", x, "\n")  # Print the value that caused the issue
  return(NA)
}
```

```
#remove non numeric  and range hyphen sqft function is applied to f4 data frame
f4$total_sqft = sapply(f4$total_sqft, convert_sqft_to_num)

#removing all rows where sqft value is NA
f5=f4[!is.na(f4$total_sqft), ]

# feature engineering and outlier Removal ---------------------------------------------------------------


#price per sq ft column is added to f5 data frame (multiplied by 1lakh becuases prices were in lakhs)
f5$price_per_sqft = (f5$price * 100000) / f5$total_sqft


#finding total number of unique locations and printing them to take an idea
unique_location=unique(f5$location)
print(unique_location)

#counting every location among 97 occured how many times in dataset in tabular format and sorting them afterwards

location_counts = table(f5$location)
location_counts = sort(location_counts)
print(location_counts)

#removing location that appeared less,but every location appears on average 100-120 times except one place where location is not mentioned so removing that blancked place
f6 = f5[f5$location != "", ]

#less than 300sqft per bedroom is unusual things in house price market,so removing such type of outliers
f7= f6[!(f6$total_sqft / f6$size < 300), ]

#describing the properties of price_per_sqft like min,max,std.deviation,mean etc.
print(summary(f7$price_per_sqft))

#for data outlier removing, using traditional technique to keep only one standard deviation below and above data from mean
#f7_out is output data initialized with data frame inside the function
#f7 datasets grouped by location and price-per_sqft are binded rows and output f7_out ias generated

remove_pps_outliers = function(f7) {
  f7_out = data.frame()
  f7_grouped = f7 %>%
    group_by(location)
```

```
  f7_grouped = f7_grouped %>%
    mutate(mean_pps = mean(price_per_sqft),
   std_pps = sd(price_per_sqft))
  f7_filtered = f7_grouped %>%
    filter(price_per_sqft > (mean_pps - std_pps) & price_per_sqft <= (mean_pps + std_pps))
  f8_out = bind_rows(f7_out, f7_filtered)
  return(f8_out)
}
f8 = remove_pps_outliers(f7)

print(dim(f))

#plotting the scatter plot of location vs price_per_sqft to observe the outliers
#bhk2 and bhk3 data are initialized with f8 dataset with bhk 2 and 3 bedrooms and location parameter
#scatter plot is plotted using ggplot(),with size,shape being adjusted,x and y labels are given as total sq feet area and lak indian rupees
#minimal theme is choosen with legend at top
#minimal theme being predefined theme and legend is info of symbols on top
plot_scatter_chart = function(f8, location)
{

  bhk2 = f8[f8$location == location & f8$size == 2, ]
  bhk3 = f8[f8$location == location & f8$size == 3, ]

  p = ggplot() +
    geom_point(data = bhk2, aes(x = total_sqft, y = price), color = 'blue', size = 3, shape = 19) +
    geom_point(data = bhk3, aes(x = total_sqft, y = price), color = 'green', size = 3, shape = 3) +
    labs(x = "Total Square Feet Area", y = "Price (Lakh Indian Rupees)", title = location) +
    theme_minimal() +
    theme(legend.position = "top") +
    scale_shape_manual(values = c(19, 3), name = "BHK", labels = c("2 BHK", "3 BHK")) +
    scale_color_manual(values = c("blue", "green"), name = "BHK", labels = c("2 BHK", "3 BHK"))

  print(p)
}

#plotting the scatter_plots of bibwewadi and katraj areas to test,detect and observe outlier points
plot_scatter_chart(f8, "Katraj")
plot_scatter_chart(f8, "Bibvewadi")


#custom function to remove outlliers and such rows where mean price_per_sqft of 1 bhk is greater than price per sq_feet of 2 bhk
#f8 is group by location and multiple manipulations are done on it using pipeline operator
#then it will check for mean of 1 bhk price per sqfeet and remove/ungroup all those rows where it is less than price per sqft for 2 bhk
```

```
remove_bhk_outliers = function(f8)
{
  f8 %>%
    group_by(location) %>%
    mutate(mean_1bhk_pps = mean(price_per_sqft[size == 1])) %>%
    filter(!(size== 2 & price_per_sqft < mean_1bhk_pps)) %>%
    ungroup()
}

# Call the function to remove outliers
f9 = remove_bhk_outliers(f8)

#replotting scatter plots for katraj and bibvewadi regions for difference identification in outlier removal
plot_scatter_chart(f9, "Katraj")
plot_scatter_chart(f9, "Bibvewadi")

# adjust the dimensions of the histogram such as height and the weight
options(repr.plot.width=20, repr.plot.height=10)

# Create a histogram having intervals of 20,on price_per_sqft whose name in the main,x & y lab as x and y axis names,color and range adjustments
hist(f9$price_per_sqft, breaks = 20, main = " count of Price Per Square Feet Histogram",
     xlab = "Price Per Square Feet", ylab = "Count", col = "red", border = "black", xlim = c(0, max(f9$price_per_sqft)))

#printing all unique no of bathroom values to look for outliers
unique_bath_values=unique(f9$bath)
print(unique_bath_values)

# Set the plot size (width,height and dimensions,name(main),color,border etc of bathroom counts)
options(repr.plot.width = 8, repr.plot.height = 6)

#histogram for bathrooms is just for outliers observation
hist(f9$bath, breaks = 20, main = "Number of Bathrooms Histogram",
     xlab = "Number of Bathrooms", ylab = "Count", col = "blue", border = "black", xlim = c(0, max(f9$bath)))

#searching for rows where bath greater than 10 to observe ambiguities
filtered_f9_bath = f9[f9$bath > 10, ]

#it is unusual to have two more bathrooms than number of bedrooms,so removing such type of outliers
filtered_bath_f9final = f9[f9$bath > f9$size + 2, ]

f10 = f9[f9$bath < f9$size + 2, ]
```

```
#searching for balcony outliers

unique_balcony_values=unique(f9$balcony)
print(unique_balcony_values)

#usually galleries not greater than number of rooms,so removing such outliers
filtered_f10_balcony = f10[f10$balcony > f10$size+1, ]
#no outliers as filtered_f10_balcony has no values

#removing size and price_per_sqft as they are useles now after outlier removal



df = f10 %>%
  select(-price_per_sqft,-mean_pps,-std_pps,-mean_1bhk_pps)

#------------------------------------------------------------------------------------------------

#  1.linear regression model on the dataset-----------------------------------------------------



#declaring our feature input matrix or independent variable x and dependent or target variable
y

#x will include all values in dataset df2 and exclude the target column price
X = df[, !names(df) %in% c("price")]

#y will form a vector of only target value price
Y = df$price

#checking for observations first few values of x and y and their dimensions

print(head(X, 3))
colnames(X) = gsub("df.location", "", colnames(X))

print(head(Y, 3))


print(dim(X))

#get length of y vector
print(length(Y))



# Split the data into training and testing sets
set.seed(20)
```

 # the seed for the model is set to seed value 10.This ensures that accuracy value or predicted values generated should be reproducible

 # a training index variable is declared using function createDatapartittion inbuilt function having targeted value y,80% trained dataset and 20% tested dataset
#list=false ensures that we will get vector or index as output
#times indicate total number of times the data is splitted
trainIndex = createDataPartition(Y, p = .8, list = FALSE, times = 1)

#show training and testing data and their dimensions

Training_data=df[trainIndex,]
Testing_data=df[-trainIndex,]

print(dim(Training_data))
print(dim(Testing_data))

#5777 trained and 1442 tested-------

#this line contains the variable X_train where all training independent data(80%)is stored and
#y-train where all output traind data is stored
X_train = X[trainIndex,]
y_train = Y[trainIndex]
#xtest and y test contains all data except traindatset that is test data
X_test = X[-trainIndex,]
y_test = Y[-trainIndex]

 # Train the Linear Regression model(lm is function used to execute linear ml model)
#linear regression is executed on Xtrain and output as y_train,"."indicates all variables in x are used as predictors
linear_regression <- lm(y_train ~ ., data = X_train)

 # Check the the summary of performance of linear reggression model
print(summary(linear_regression))

 # predicting the house price for testing dataset using linear regression model
predicted_houseprice = predict(linear_regression, X_test)

#evaluation matrix for the model (R^2 value,MAE(mean absolute error),MSE(mean squared error))
#calculating the r-squared value to determine the model's accuracy
#formula for r-squared=1-(sum of squared residuals/sum of squared totals)
cat("\n \n evaluation matrix for linear regression : \n \n")
R2 = 1 - sum((y_test - predicted_houseprice)^2) / sum((y_test - mean(y_test))^2)

```
cat("\n accuracy for linear regression model:",R2)
# Calculate Mean Absolute Error (MAE)
MAE = mean(abs(y_test - predicted_houseprice))
cat("\n Mean Absolute Error (MAE):", MAE)

# Calculate Mean Squared Error (MSE)
MSE = mean((y_test - predicted_houseprice)^2)
cat("\n Mean Squared Error (MSE):", MSE)

# Calculate Mean Squared Error (MSE)
MSE = mean((y_test - predicted_houseprice)^2)
cat("\n Mean Squared Error (MSE):", MSE)

# Calculate Root Mean Squared Error (RMSE)
RMSE = sqrt(mean((y_test - predicted_houseprice)^2))
cat("\n Root Mean Squared Error (RMSE):", RMSE)

# Calculate Mean absolute percentage Error (MAPE)
MAPE = mean(abs((y_test - predicted_houseprice) / y_test)) * 100
cat("\nMean absolute percentage Error (MAPE):", MAPE)

cat("\n\n\n")

#2.decision trees------------------------------------------------------------------------------------------

# Train the Decision Tree model
decision_tree <- rpart(y_train ~ ., data = X_train)
# Predict house prices using Decision Tree model
predicted_houseprice_tree <- predict(decision_tree,X_test )

# Calculate R-squared
R2_tree <- 1 - sum((y_test - predicted_houseprice_tree)^2) / sum((y_test - mean(y_test))^2)

# Calculate Mean Absolute Error (MAE)
MAE_tree <- mean(abs(y_test - predicted_houseprice_tree))

# Calculate Mean Squared Error (MSE)
MSE_tree <- mean((y_test - predicted_houseprice_tree)^2)

# Calculate Root Mean Squared Error (RMSE)
RMSE_tree <- sqrt(mean((y_test - predicted_houseprice_tree)^2))

# Calculate Mean absolute percentage Error (MAPE)
MAPE_tree <- mean(abs((y_test - predicted_houseprice_tree) / y_test)) * 100

# Print Decision Tree model evaluation metrics
cat("\nEvaluation matrix for Decision Tree:\n\n\n")
cat("Accuracy for Decision Tree model (R-squared):", R2_tree, "\n")
```

```
cat("Mean Absolute Error (MAE):", MAE_tree, "\n")
cat("Mean Squared Error (MSE):", MSE_tree, "\n")
cat("Root Mean Squared Error (RMSE):", RMSE_tree, "\n")
cat("Mean Absolute Percentage Error (MAPE):", MAPE_tree, "\n\n\n")


# 3.support vector machine (SVM0---------------------------------------------------------------------------
-----------



# Assuming you have already split your data into training and testing sets (X_train, y_train, X_test)
# Train the SVR model
svm_model <- svm(y_train ~ ., data = X_train, kernel = "radial")

# Predict house prices for the test dataset
predicted_house_prices <- predict(svm_model, X_test)

# Calculate R-squared
R2_svm <- 1 - sum((y_test - predicted_house_prices)^2) / sum((y_test - mean(y_test))^2)

# Calculate Mean Absolute Error (MAE)
MAE_svm <- mean(abs(y_test - predicted_house_prices))

# Calculate Mean Squared Error (MSE)
MSE_svm <- mean((y_test - predicted_house_prices)^2)

# Calculate Root Mean Squared Error (RMSE)
RMSE_svm <- sqrt(mean((y_test - predicted_house_prices)^2))

# Calculate Mean Absolute Percentage Error (MAPE)
MAPE_svm <- mean(abs((y_test - predicted_house_prices) / y_test)) * 100

# Print SVM model evaluation metrics


cat("Evaluation matrix for Support Vector Machine (SVM):\n\n")
cat("Accuracy for SVM model (R-squared):", R2_svm, "\n")
cat("Mean Absolute Error (MAE):", MAE_svm, "\n")
cat("Mean Squared Error (MSE):", MSE_svm, "\n")
cat("Root Mean Squared Error (RMSE):", RMSE_svm, "\n")
cat("Mean Absolute Percentage Error (MAPE):", MAPE_svm, "\n")



#4.Random_Forest------------------------------------------------------------------------------------------
------------------
```

```
# Train the Random Forest model
random_forest_model <- randomForest(y_train ~ ., data = X_train)

# Predict house prices for the test dataset
predicted_houseprice_rf <- predict(random_forest_model, X_test)

# Calculate R-squared
R2_rf <- 1 - sum((y_test - predicted_houseprice_rf)^2) / sum((y_test - mean(y_test))^2)

# Calculate Mean Absolute Error (MAE)
MAE_rf <- mean(abs(y_test - predicted_houseprice_rf))

# Calculate Mean Squared Error (MSE)
MSE_rf <- mean((y_test - predicted_houseprice_rf)^2)

# Calculate Root Mean Squared Error (RMSE)
RMSE_rf <- sqrt(mean((y_test - predicted_houseprice_rf)^2))

# Calculate Mean Absolute Percentage Error (MAPE)
MAPE_rf <- mean(abs((y_test - predicted_houseprice_rf) / y_test)) * 100

# Print Random Forest model evaluation metrics
cat("Evaluation matrix for Random Forest:\n\n")
cat("Accuracy for Random Forest model (R-squared):", R2_rf, "\n")
cat("Mean Absolute Error (MAE):", MAE_rf, "\n")
cat("Mean Squared Error (MSE):", MSE_rf, "\n")
cat("Root Mean Squared Error (RMSE):", RMSE_rf, "\n")
cat("Mean Absolute Percentage Error (MAPE):", MAPE_rf, "\n")
#------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------


#data visualization through graphs----------------------------------------------------------------------------------------------------------------------------------------------


# Create data frames for plotting
lr_plot_data <- data.frame(Actual = y_test, Predicted = predicted_houseprice, Model = "Linear Regression")
dt_plot_data <- data.frame(Actual = y_test, Predicted = predicted_houseprice_tree, Model = "Decision Tree")
svm_plot_data <- data.frame(Actual = y_test, Predicted = predicted_house_prices, Model = "SVM")
Rf_plot_data <- data.frame( Actual = y_test, Predicted = predicted_houseprice_rf, Model = "Random Forest")

# Combine the data frames
```

```
combined_plot_data <- rbind(lr_plot_data, dt_plot_data, svm_plot_data,Rf_plot_data)

# Create the plot
  predicted_actual_plot <- ggplot(combined_plot_data, aes(x = Actual, y = Predicted, color = Model)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, linetype = "dashed") +
  labs(title = "Predicted vs. Actual House Prices", x = "Actual Prices", y = "Predicted Prices") +
  theme_minimal()

# Convert the ggplot object to a plotly object
predicted_actual_plot <- ggplotly(predicted_actual_plot)


# Display the plot
print(predicted_actual_plot)

# Create a data frame for model accuracy comparison
accuracy_data <- data.frame(Model = c("Linear Regression", "Decision Tree", "SVM","Random Forest"),
                            Accuracy = c(R2, R2_tree, R2_svm,R2_rf))

# Create the accuracy comparison plot
accuracy_plot <- ggplot(accuracy_data, aes(x = Model, y = Accuracy)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Model Accuracy Comparison", x = "Model", y = "R-squared Accuracy") +
  theme_minimal() +
  ylim(0, 1)  # Set y-axis limits

# Display the accuracy comparison plot
print(accuracy_plot)

# so linear regression is found most accurate with 85% accuracy
#let's perform k-fold(10 fold) cross validation on linear regression model


#k-flod cross validation--------------------------------------------------------------------------------------------------------------------------------------



#k-fold(10 folds) cross validation to check validation of accuracy of linear regression model
cat("\n\n\n")
# Set the number of folds and test data size(0.2 = 20%)
k = 10
test_size = 0.2

#declaring empty vector to store cross validation results
```

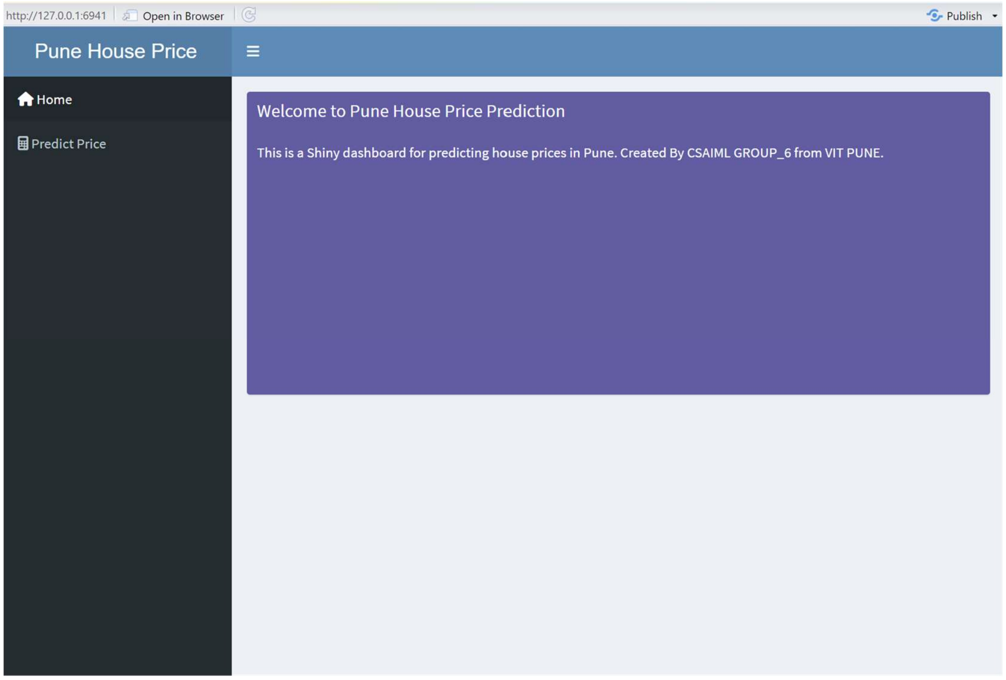
```
cv_results = numeric(k)

#seed is set using seed value 0 to minimize randomness and provide accurate results
# Set seed for reproducibility
set.seed(20)

# Perform cross-validation by loop iterating from i=1 to 10
for (i in 1:k)
{
  # again creating linear regression model for cross validation,p--trained size=total-test_size)
  trainIndex = createDataPartition(Y, p = 1 - test_size, list = FALSE)
  X_train = X[trainIndex,]
  y_train = Y[trainIndex]
  X_test = X[-trainIndex,]
  y_test = Y[-trainIndex]
  linear_regression = lm(y_train ~ ., data = X_train)
  predicted_houseprice = predict(linear_regression, newdata = X_test)
  R2 = 1 - sum((y_test - predicted_houseprice)^2) / sum((y_test - mean(y_test))^2)

  # Store the result
  cv_results[i] = R2 # each r-squared value accuracy index is stored in ith index of cross validation
vector
}

# Print cross-validation results
cat("Cross-validation results:\n", cv_results, "\n")
cat("Mean R-squared:", mean(cv_results), "\n")
cat("\n\n\n")

#mean accuracy obtained on 10-fold cross validation of linear regression is 78%

#predicting house prices with user input

# Create a data frame with default values
custom_data <- data.frame(
  size = numeric(1),
  total_sqft = numeric(1),
  bath = numeric(1),
  balcony = numeric(1),
  location = character(1)
)


#GUI using shinny-----------------------------------------------------------------------------------------------
------------------------------

# Load necessary libraries
library(shiny)
```

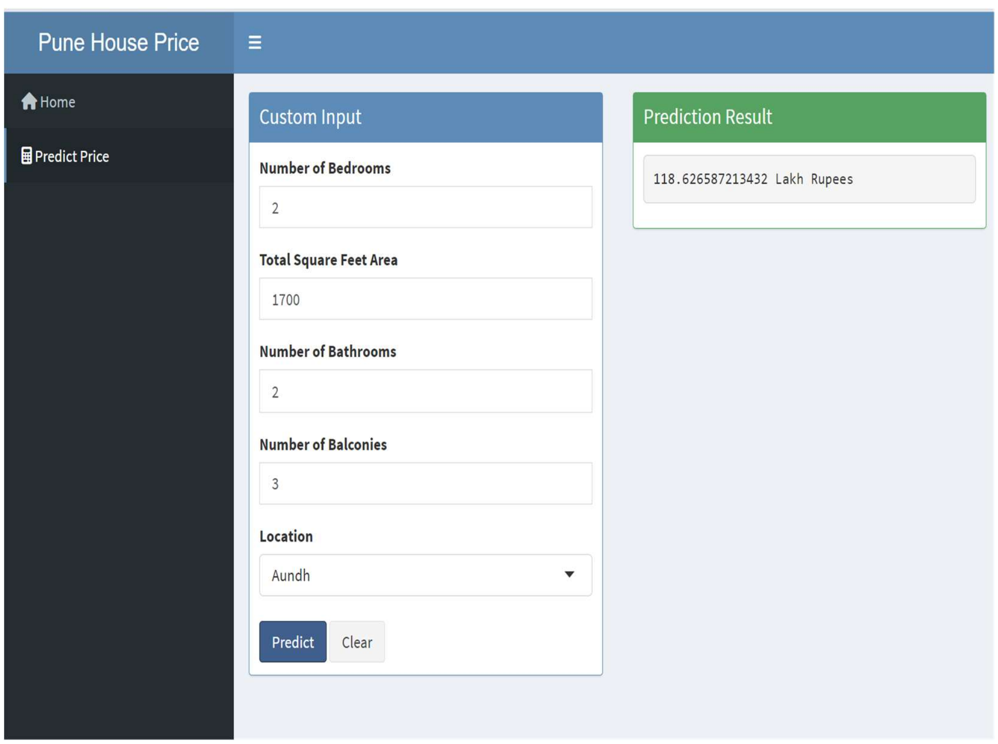
```
library(shinydashboard)

# Define UI for the Shiny app
ui <- dashboardPage(
  dashboardHeader(title = "Pune House Price Prediction"),

  dashboardSidebar(
    sidebarMenu(
      menuItem("Home", tabName = "home", icon = icon("home")),
      menuItem("Predict Price", tabName = "predict", icon = icon("calculator"))
    )
  ),

  dashboardBody(
    tabItems(
      tabItem(
        tabName = "home",
        fluidRow(
          box(
            title = "Welcome to Pune House Price Prediction",
            width = 12,
            height = 300,
            background = "purple",
            "This is a Shiny dashboard for predicting house prices in Pune.\n Created By CSAIML
GROUP_6 from VIT PUNE."

          )
        )
      ),

      tabItem(
        tabName = "predict",
        fluidRow(
          box(
            title = "Custom Input",
            width = 6,
            status = "primary",
            solidHeader = TRUE,
            textInput("size", "Number of Bedrooms", value = ""),
            textInput("total_sqft", "Total Square Feet Area", value = ""),
            textInput("bath", "Number of Bathrooms", value = ""),
            textInput("balcony", "Number of Balconies", value = ""),
            selectInput("location", "Location", choices = unique_location, selected = ""),
            actionButton("predictBtn", "Predict", class = "btn-primary"),
            actionButton("clearBtn", "Clear", class = "btn-default")
          ),
          box(
            title = "Prediction Result",
```

```
          width = 6,
          status = "success",
          solidHeader = TRUE,
          verbatimTextOutput("predictedPrice")
        )
      )
    )
  )
 )
)

# Define server logic
server <- function(input, output) {
 # Load your trained linear regression model here (replace with actual code)
 # Example: linear_regression <- lm(price ~ size + total_sqft + bath + balcony + location, data =
training_data)

 # Function to predict house price
 predictPrice <- function(size, total_sqft, bath, balcony, location) {
  custom_data <- data.frame(
   size = as.numeric(size),
   total_sqft = as.numeric(total_sqft),
   bath = as.numeric(bath),
   balcony = as.numeric(balcony),
   location = location
  )

  # Replace with your actual linear regression model
  predicted_price <- predict(linear_regression, newdata = custom_data)
  return(predicted_price)
 }

 # Predict and update the output when the "Predict" button is clicked
 observeEvent(input$predictBtn, {
  predicted_price <- predictPrice(
   input$size,
   input$total_sqft,
   input$bath,
   input$balcony,
   input$location
  )

  # Convert the predicted price to lakh rupees
  predicted_price_in_lakh <- predicted_price
  output$predictedPrice <- renderText({
   paste(predicted_price_in_lakh, "Lakh Rupees")
  })
# Run the Shiny app
```

shinyApp(ui = ui, server = server)

➢ OUTPUT/RESULTS :





➢ INTRODUCTION -:

A House Price Prediction Project is a data-driven endeavour that aims to predict the selling or buying prices of residential properties. This type of project is a common application of machine learning and data analysis, particularly in the real estate industry. The goal of such a project is to provide valuable insights and predictions for both homeowners and potential buyers, as well as for real estate professionals, investors, and policymakers.

- Real estate is a substantial part of the global economy, and buying or selling a house is a significant financial decision. Predicting accurate house prices is crucial for various stakeholders, including homebuyers, sellers, real estate agents, and property investors.

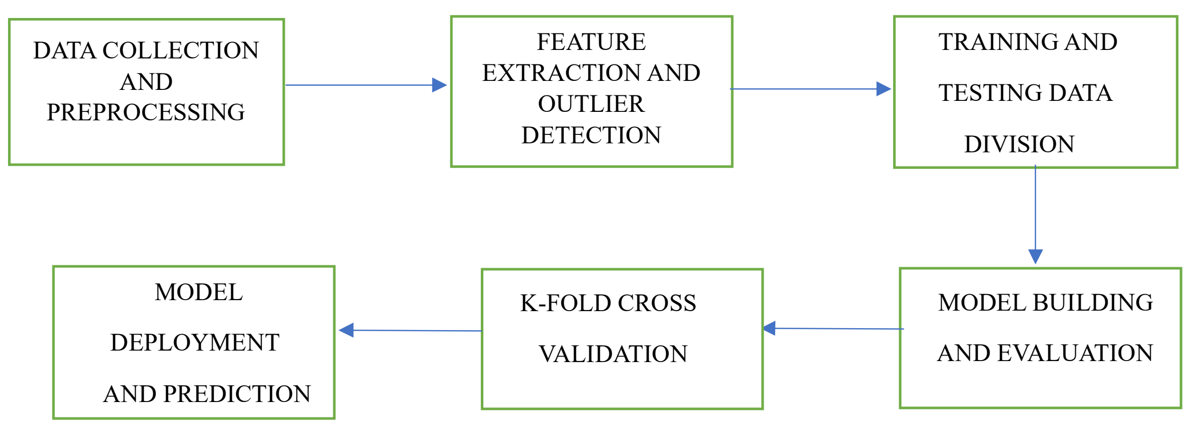
Objectives :

1. Predict future house prices to assist homebuyers in making informed decisions on when and where to purchase a property.

2. Support real estate investors in identifying profitable opportunities and assessing investment risks in the Pune housing market.

3. Provide market researchers and professionals with insights into Pune's real estate trends and price dynamics for strategic planning and development.

Methodology :



- **DATABASE DESCRIPTION :** The database that we have collected for our project is obtained from the machine learning online platform Kaggle. It consists of 13320 rows and 9 different columns. The data has one dependent column house price in lakh rupees and independent columns like house society, house building status, number of bedrooms, bathrooms, balconies, total area in sq. Feet and locations of Pune in character format. It means that data has one independent variable and two or more dependent variable and output is of regression type. so, supervised regression algorithms will be useful for our case. Therea were in total 97 unique locations including others which are listed below:

```
[1] "Alandi Road"              "Ambegaon Budruk"          "Anandnagar"
[4] "Aundh"                    "Aundh Road"               "Balaji Nagar"
[7] "Bhandarkar Road"          "Bibvewadi"                "Bopodi"
[10] "Budhwar Peth"            "Bund Garden Road"         "Camp"
[13] "Chandan Nagar"           "Dapodi"                   "Deccan Gymkhana"
[16] "Dehu Road"               "Dhankawadi"               "Dhayari Phata"
[19] "Dhole Patil Road"        "Erandwane"                "Fatima Nagar"
[22] "Fergusson College Road"  "Ganesh Peth"              "Ganeshkhind"
[25] "Ghorpade Peth"           "other"                    "Gokhale Nagar"
[28] "Gultekdi"                "Guruwar peth"             "Hadapsar"
[31] "Hadapsar Industrial Estate" "Jangali Maharaj Road"  "Kalyani Nagar"
[34] "Karve Nagar"             "Karve Road"               "Kasba Peth"
[37] "Khadaki"                 "Khadki"                   "Kharadi"
[40] "Kondhwa"                 "Kondhwa Khurd"            "Koregaon Park"
[43] "Kothrud"                 "Law College Road"         "Laxmi Road"
[46] "Lulla Nagar"             "Mahatma Gandhi Road"      "Mangalwar peth"
[49] "Manik Bagh"              "Market yard"              "Mukund Nagar"
[52] "Mundhawa"                "Nagar Road"               "Nana Peth"
[55] "Narayan Peth"            "Narayangaon"              "Navi Peth"
[58] "Padmavati"               "Parvati Darshan"          "Pashan"
[61] "Paud Road"               "Pirangut"                 "Prabhat Road"
[64] "Pune Railway Station"    "Rasta Peth"               "Raviwar Peth"
[67] "Sadashiv Peth"           "Sahakar Nagar"            "Salunke Vihar"
[70] "Sasson Road"             "Satara Road"              "Senapati Bapat Road"
[73] "Shaniwar Peth"           "Shivaji Nagar"            "Sinhagad Road"
[76] "Somwar Peth"             "Swargate"                 "Tilak Road"
[79] "Uruli Devachi"           "Vadgaon Budruk"           "Wadgaon Sheri"
[82] "Viman Nagar"             "Vishrant Wadi"            "Wagholi"
[85] "Wakadewadi"              "Wanowrie"                 "Warje"
[88] "Yerawada"                "Baner"                    "Baner road"
[91] "Bhavani Peth"            "Ghorpadi"                 "Hingne Khurd"
[94] "Katraj"                  "Kondhwa Budruk"           "Model colony"
[97] "Shukrawar Peth"
```

- **Data preprocessing :** The first process involved in house price prediction of Pune city is preprocessing of data. Data preprocessing refers to the process of cleaning the data, removing inconsistencies and extracting out relevant information. In the data preprocessing part following operations were performed on our dataset-:
  1) **Defining packages ,tools and libraries-:** shiny & Shiny dashboard (for UI design using shiny package), dplyr and tidyr (for data manipulations, performing operations on data and model fitting), ggplot2 (for plotting graphs) ,plotly (for generating moving graphs), caret (for accessing machine learning tools and functions) and glmnet(for regression model training).In addition with that e1071(for svm model), rpart (for decision trees model) and randomForest (for random forest model),boot(for k-fold cross validation) were also used.

  2) **Removing missing values-:** All the missing values or NA values containing rows were omitted from the data.

3) **Removing inconsistencies in Bhk values-:** The Bhk values containing inconsistencies like 2BHK, 4 bedroom etc were all converted to numeric format and stored in a new column size and old BHK column was removed. Rows with BHK >10 were also removed due to improper reliability
Using pipeline operator %in%.

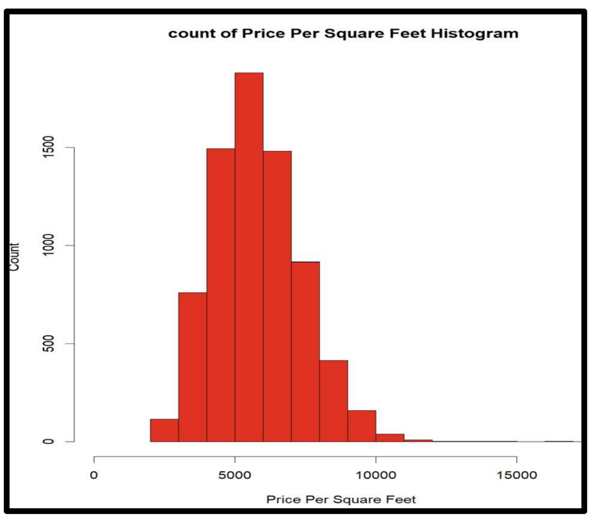
4) **Inconsistencies in Total_sqft column-:** Ambiguities in total_sqft column were removed and entire column was converted into numeric values. The outliers like 45 yards,56 metres, 678 sqft etc were omitted and ranged sqft values like 456-678 etc were converted to median/average value of Range using appropriate functions.

- **Feature Extraction:** Feature engineering refers to the process of extracting relevant features from the data that are useful for model training and accuracy enhancing. The feature Engineering and outlier removal processes used in the project are-:

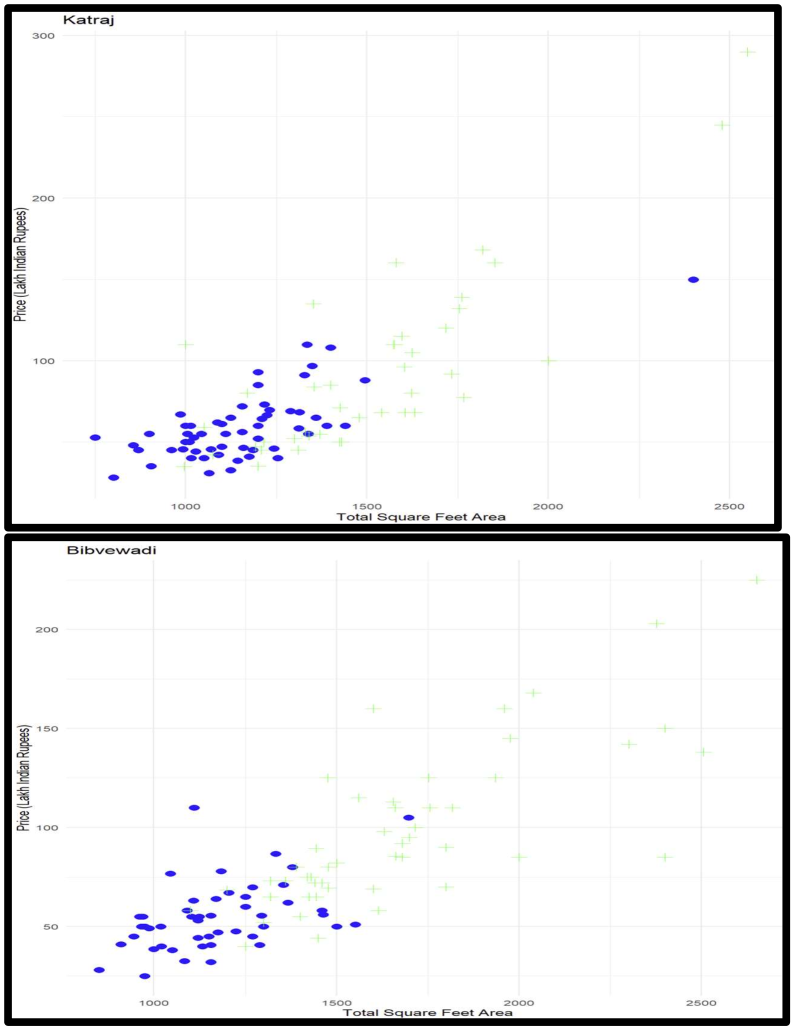
1) **Removing unnecessary columns -:** some of the features or column vectors shows more relationship with the output whereas few has no effect on output. Based on this relationship we can exclude those parameters whose presence or absence does not affect output. For that purpose, the Multiple-linear regression t-value test is used where low t-value and higher p-value. Hence , availability, society and area_type columns were found to be most useless by this test . Hence they were omitted from the table.

```
Built-up  Area          Carpet  Area           Plot  Area Super built-up  Area
          2418                    87                 2025                 8790
```

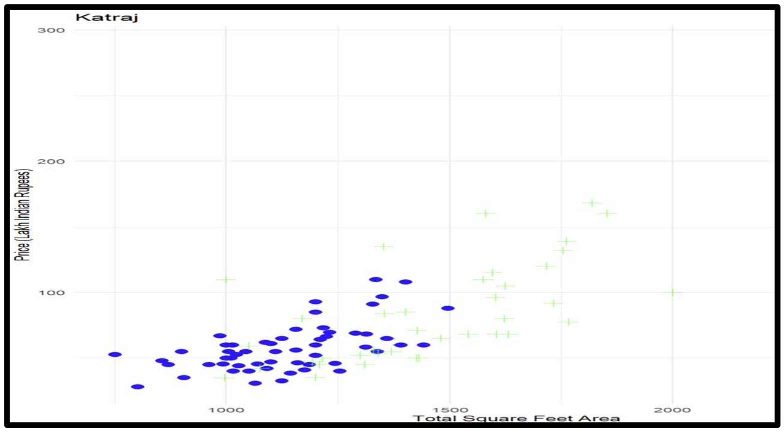
2) **Addition of price per_sqft and removal of non-feasible components**-: price per sqft column was added depending on total sq_ft /price and price of 1 bedroom area persqft was calculated and rows with price less than 300 were removed. Generally , for all price_per_sqft one mean normalization is applied, i.e they are kept in between the range (mean + standard deviation) and (mean - standard deviation). Others out of range were considered as outliers and omitted. also graphical visualization is done for price per_sqft and count of all those areas as follows-:

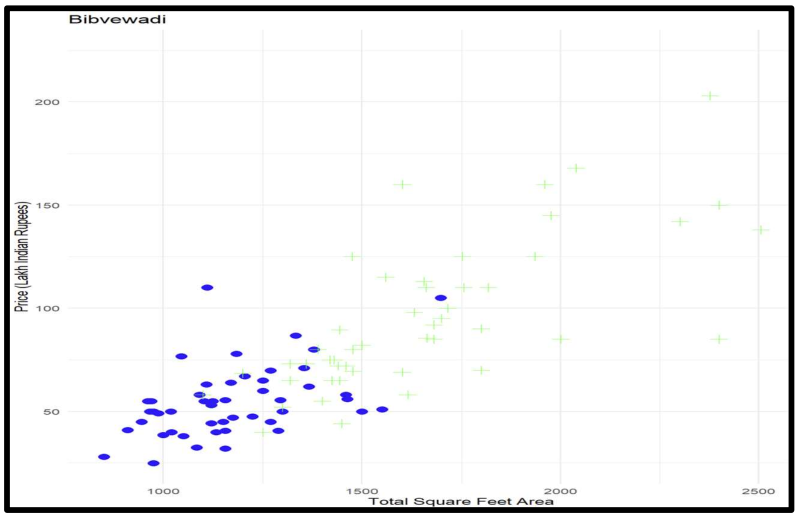


3) **Graphical analysis of outlier detection-:** for removal of outliers data visualization technique is used and through graphical analysis any inconsistencies were sorted out. For this purpose price per_sqft vs price graph was plotted for katraj and bibvewadi regions and outliers that were opposing the trend were eliminated using appropriate functions.



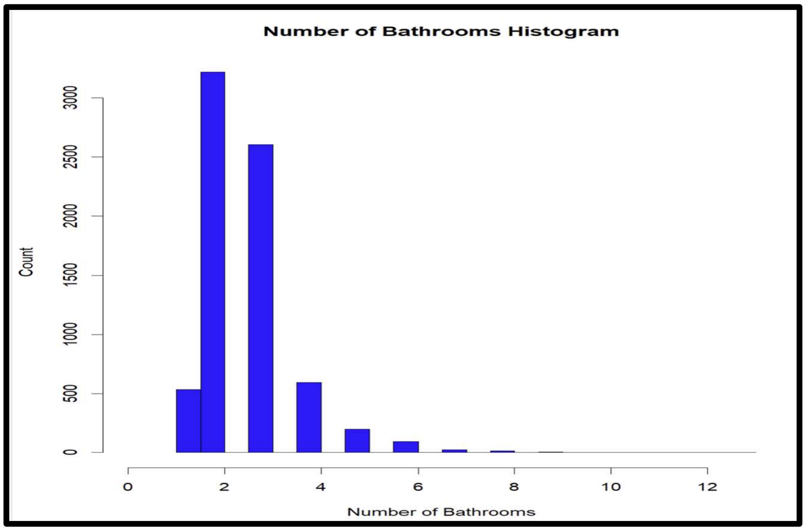
(katraj & bibwewadi—price vs sq_ft ---before outlier removal)





(katraj & bibwewadi—price vs sq_ft ---after outlier removal)

4) **Removing incorrect features of balcony column :** In general in any apartment system, the number of balconies cannot be more than the (number of bedrooms +1). So, considering this case and removing all such rows. Removing price_per_sqft, mean_pps and s.deviation_pps as **these features vectors are useless now. The number of flats associated with n-number of** bathrooms are plotted in the below graph-:



- **Model training and evaluation -:** for training and testing purpose the given model is divided into training and testing datasets. Out of 7000 left rows after processing 80% dataset were considered as training dataset and 20% dataset were considered as testing dataset. The four models mainly multiple linear regression, decision trees, support vector machine and random Forest were applied on training dataset for model training and evaluation of each of them is done on testing dataset and conclusions were drawn from evaluation metrices. It is found that the linear regression was proven to be most efficient model with highest accuracy of 85 % as compared to others. The machine learning models applied for our project is are as follows-:

1. **Multiple linear regression-:** Multiple linear regression is a statistical method used to model the relationship between a dependent variable and two or more independent variables by fitting a linear equation to the data. It extends simple linear regression to account for multiple predictors, allowing us to quantify how each independent variable contributes to the variation in the dependent variable while controlling for the output.

2. **Decision trees**-: Decision trees are a machine learning technique used for both classification a nd regression tasks. They create a tree-like structure where each internal node represents a decisi-on based on a specific feature, and each leaf node represents a predicted outcome. Decision trees are designed to recursively split the data into subsets based on the most informative features, mak ing them a powerful tool for data-driven decision-making and predictive modeling.

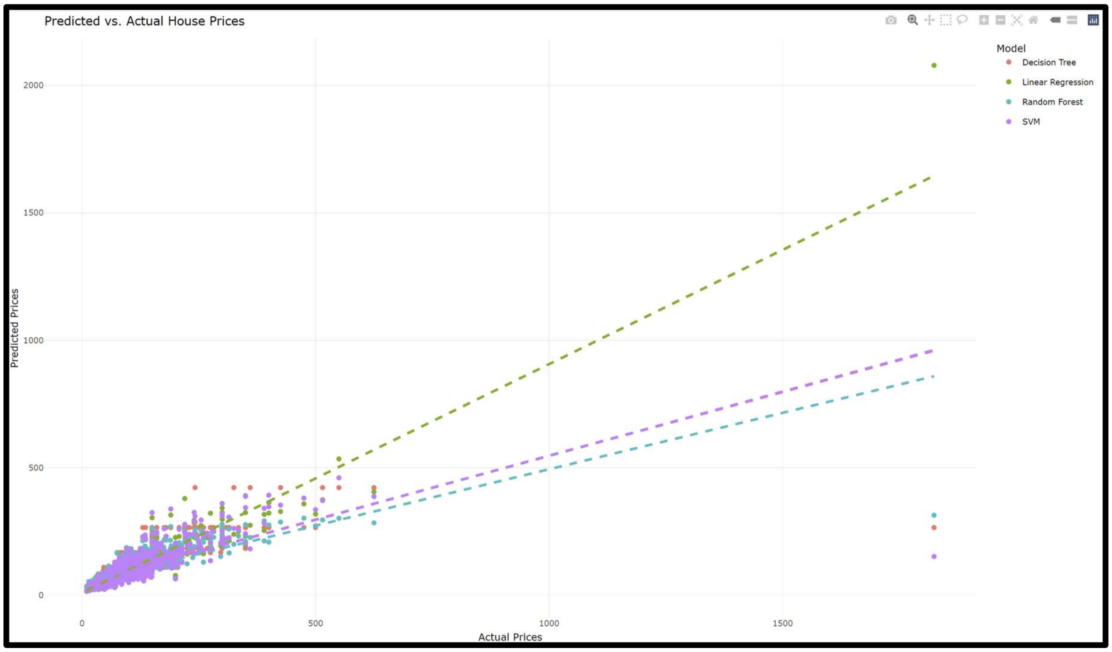
3. **Support vector machine** -: Decision trees are a machine learning technique used for both classification and regression tasks. They create a tree-like structure where each internal node represents a decision based on a specific feature, and each leaf node represents a predicted outcome. Decision trees are designed to recursively split the data into subsets based on the most informative features, making them a powerful tool for data-driven decision-making and predictive modelling.

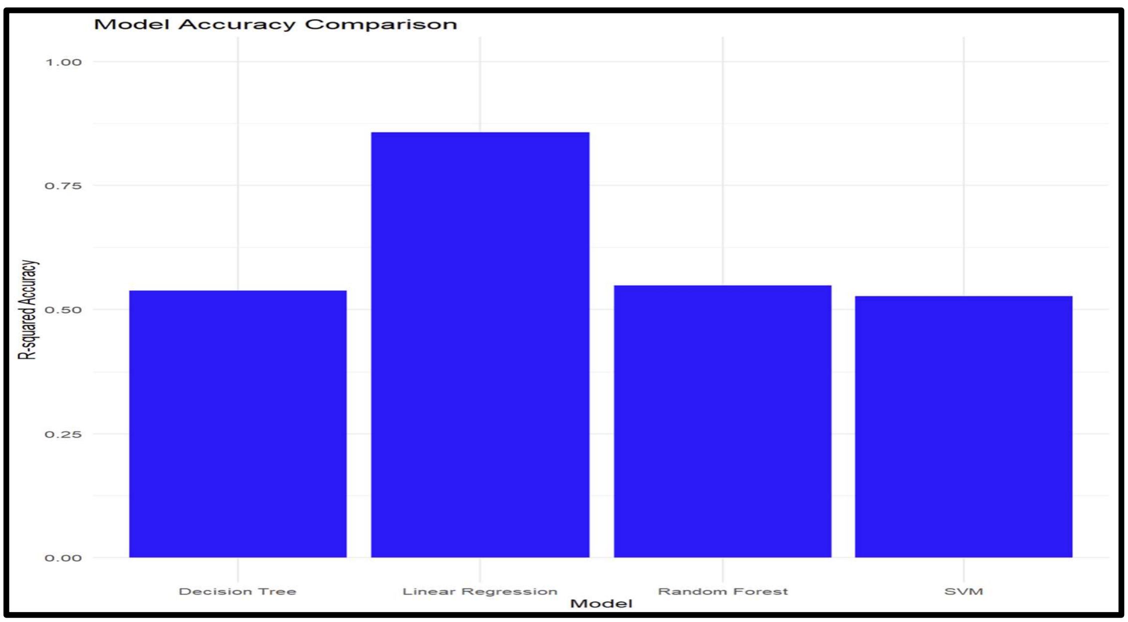
4. **Random Forest** -: Random Forest is an ensemble machine learning method that combines multiple decision trees to improve predictive accuracy and reduce overfitting. It works by constructing a multitude of decision trees during training and making predictions by aggregating the results from these individual trees. This ensemble approach enhances the robustness and

generalizability of the model, making Random Forest a popular choice for various classification and regression tasks in data science and machine learning.

```
Evaluation matrix for linear regression :


accuracy for linear regression model: 0.857423
Mean Absolute Error (MAE): 19.09531
Mean Squared Error (MSE): 813.94
Root Mean Squared Error (RMSE): 28.52965
Mean absolute percentage Error (MAPE): 21.74148


Evaluation matrix for Decision Trees :

accuracy for Decision Tree model: 0.538095
Mean Absolute Error (MAE): 22.04525
Mean Squared Error (MSE): 2636.915
Root Mean Squared Error (RMSE): 51.3509
Mean Absolute Percentage Error (MAPE): 24.65084


Evaluation matrix for Support vector machine :


accuracy for Support Vector machine model: 0.5262627
Mean Absolute Error (MAE): 19.44698
Mean Squared Error (MSE): 2704.462
Root Mean Squared Error (RMSE): 52.00444
Mean Absolute Percentage Error (MAPE): 20.1937




Evaluation matrix for Random Forest:

accuracy for Random Forest model (R-squared): 0.5483928
Mean Absolute Error (MAE): 21.08013
Mean Squared Error (MSE): 2578.127
Root Mean Squared Error (RMSE): 50.77526
Mean Absolute Percentage Error (MAPE): 23.53973
```

- **Plots for actual vs predicted values for all models and accuracy comparison for all trained models-:**





➢ K-fold cross validation on Multiple linear regression-:

The most accurate model among all tested ones is multiple linear regression. So, applying k-fold cross validation on linear regression for 10 folds (k=10).

K-fold cross-validation is a technique used in machine learning and statistics to assess the performance and generalizability of a predictive model. It involves dividing the dataset into K subsets of approximately equal size. The model is trained and tested K times, using a different subset as the test data in each iteration while the remaining subsets are used for training. This process helps estimate how well the model will perform on unseen data and reduces the risk of overfitting. The final evaluation is typically an average of the K individu al evaluations, providing a more reliable assessment of the model's performance. Common choices for K are 5 or 10, but it can vary depending on the specific application.

10-fold cross validation results on Multiple linear regression-:

Cross-validation results:

```
0.857423 0.7646136 0.7284056 0.8452887 0.7463447 0.7476483
0.8581502 0.7574972 0.7386272 0.788369
```

Mean R-squared: 0.7832368

So, the overall accuracy (mean-R squared) of our trained model was found to be **78.32368 %**

➢ Model deployment using GUI based on Shiny package-:

Model deployment is the process of taking a trained machine learning or statistical model and making it accessible for use in a production or operational environment. In other words, it's the step where the model is put into action to make predictions or classifications on real-world data.

- **Model Deployment using Shiny :**

Shiny App Setup: The code sets up a Shiny web application with a dashboard that has two tabs, "Home" and "Predict Price."

Home Tab: The "Home" tab provides information about the project and a brief introduction.

Predict Price Tab: The "Predict Price" tab allows users to input details about a house, including the number of bedrooms, square feet area, bathrooms, balconies, and location.

Model Integration: The server logic of the Shiny app loads the trained Linear Regression model and provides a function to predict house prices based on user inputs.

Prediction: When the user clicks the "Predict" button, the Shiny app calls the prediction function and displays the estimated house price.

Clear Button: Users can clear the input fields by clicking the "Clear" button.

Overall, our project code demonstrates how to preprocess data, train machine learning models, and deploy a predictive model using a user-friendly Shiny web application. Users can input house details, and the app predicts the house price based on the trained model. The Shiny framework allows for easy interaction and visualization of model results

➢ Applications of project:

- predicting house prices in Pune using machine learning and deploying it through a Shiny web application, can have several practical applications:

- **Real Estate Investment:** Potential real estate investors can use the application to estimate the market value of properties before making investment decisions. This helps in identifying properties that offer good return on investment.

- **Homebuyers:** Homebuyers can use the application to get an estimate of a property's value. This can assist them in making informed decisions and negotiations when purchasing a house.

- **Property Valuation Services:** Real estate agencies and property valuation services can utilize the model to provide accurate and quick property valuations to their clients.

- **Property Listing Websites:** Online property listing websites can integrate this application to provide estimated property values on their listings, helping both buyers and sellers.

- **Real Estate Agents:** Real estate agents can use the tool to provide data-backed recommendations to clients, enhancing their credibility and trustworthiness.

- **Market Research:** The model can be employed for real estate market research, allowing analysts to study trends and pricing patterns in different areas of Pune.

- **Government and Taxation:** Government agencies can use such models to assess property tax values and evaluate tax assessments more accurately.

- **Risk Assessment:** Banks and financial institutions can integrate the tool for risk assessment when providing loans for property purchases, ensuring the property's value is in line with the loan amount.

➢ Conclusion :

- This project successfully built a tool to predict house prices in Pune using machine learning. It offers valuable assistance to homebuyers, investors, and real estate professionals. The user-friendly web application simplifies the process, making informed property decisions more accessible.
- Graphical analysis is a key in data preprocessing process where it helps in significant amount to remove outlier and inconsistencies.
- Regression techniques like Multiple linear regression, linear regression etc overcomes the powerful algorithms like random forest, SVM and Decision trees in case of output-oriented problems like house price prediction, stock market prediction etc.
- K-fold cross validation approach validates the accuracy or efficiency of overall model and avoid any cases of overfitting.
- Overall, data science and machine learning are a versatile technological field too solve various industry related issues.

➢ **LINK For the spreadsheet of literature Survey:**

https://docs.google.com/spreadsheets/d/15rb6hCMLr_FIDFjoFagmwMKwUX1kpYbq5aVFRbuHZok/edit?usp=sharing

➢ References:

[1]Aishwarya Mujumdar and Dr. Vaidehi V. "Diabetics prediction using machine learning algorithm" ICRATC 2019

[2] Charith Silva, Charith Silva, Mahsa Saraee, Mahsa Saraee, "Data science in public mental health", 2019

[3]Elias Dritsas ,Maria Trigka,"Machine Learning Techniques for Chronic Kidney Disease Risk Prediction ", 2022

[4]Vivek Kumar,Brojo Kishore Mishra,Abhishek Verma,"Prediction of Malignant & Benign Breast Cancer: A Data Mining Approach in Healthcare Applications",2021

[5]Divya Krishnani, Anjali Kumari, Akash Dewangan, Aditya Singh, Nenavath Srinivas Naik,"Prediction of Coronary Heart Disease using Supervised Machine Learning Algorithm ",2019

[6]Tonmoy Hossain, Fairuz Shadmani Shishir , Mohsena Ashraf, MD Abdullah Al Nasim,Faisal Muhammad Shah"Brain Tumor DetectionUsing Convolutional Neural Network(CNN)",2019

[7]Praveen Khethavath,Yiu Chung Yau,Jose A. Figueroa,"Secure Pattern-Based Data Sensitivity Framework for Big Data in Healthcare",2019

[8] Ann-Marie Mallon, Dieter A. Häring , Frank Dahlke , Piet Aarden , Soroosh Afyouni ,Daniel Delbarre, "Advancing datascience in drug development through an innovative computational framework for data sharing and statistical analysis",2019

[9]Adriano Barbosa-Silva ,Milena Magalhães , Gilberto Ferreira da Silva , Fabricio Alves Barbosa da Silva , Flávia Raquel Gonçalves Carneiro , and Nicolas Carels ,"A Data Science Approach for the Identification of Molecular Signatures of Aggressive Cancers ",2022

[10]Sri Venkat Gunturi Subrahmany, Dasharathraj K. Shetty, Vathsala Patil, B. M. Zeeshan Hameed, Rahul Paul, Komal Smriti, Nithesh Naik,Bhaskar K. Somani,"The role of data science in healthcare advancements: applications, benefits, and future prospect",

[11]Samir S. Yadav1 and Shivajirao M. Jad ,"Deep convolutionary network based image classification for disease diagnosis",2019

[12]Fábio C. P. Navarro, Hussein Mohsen, Chengfei Yan, Shantao Li, Mengting Gu, William Meyerson and Mark Gerstein,"Genomics and data science: an application within an umbrella",2019

[13]kitoshi Shimazaki , Daiju Ueda, Antoine Choppin , Akira Yamamoto, Takashi Honjo, Yuki Shimahara & Yukio Miki,"Deep learning- based algorithm for lung cancer detection on chest radiographs using the segmentation method ",2022

[14]Anthony Pochini, Yitian Wu, Gongzhu Hu"Data Mining for Lifestyle Risk Factors Associated with Overweight and Obesity among Adolescents",2019

[15]Janssen R&D, Janssen Pharmaceutica,"A Machine Learning Approach to Predict HIV Viral Load Hotspots in Kenya Using RealWorld Data ",2023[11]Samir S. Yadav1 and Shivajirao M. Jad ,"Deep convolutionary network based image classification for disease diagnosis",2019

[18]Jun Liu and Xuewei Wang ,"Human diseases and diagnosis detection based on deep learning: a review",2021

[17]Vivek Kumar, Brojo Kishore Mishra, Manuel Mazzara, Dang N. H. Thanh, Abhishek Verma ,"Prediction of Malignant & Benign Breast Cancer: A Data Mining Approach in Healthcare Applications ",2020

[18]Hiroshi Takeuchi, Senior Member, IEEE, Naoki Kodama, Takeshi Hashiguchi and Doubun Hayashi,"automated h Healthcare Data Mining Based on a Personal Dynamic Healthcare System,2021

[19]ALI RIZWAN, AHMED ZOHA, RUI ZHANG, WASIM AHMAD, KAMRAN ARSHAD, NAJAH ABU ALI, AKRAM ALOMAINY, MUHAMMAD ALI IMRAN, AND QAMMER H. ABBASI "A Review on the Role of Nano-Communication in Future Healthcare Systems: A Big Data Analytics Perspective",2019

[20]Nina S. Godbole,"arch into Making Healthcare Green with Cloud, Green IT, and Data Science to Reduce Healthcare Costs and Combat Climate Change ",2023