PROJECT REPORT

ON

Case Study: Real Direct (online real estate firm)

Submitted in

DATA SCIENCE

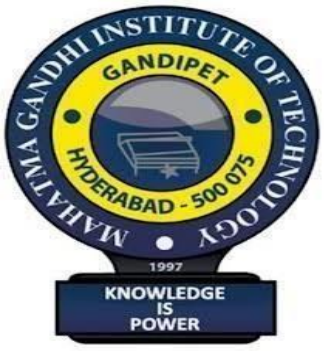
By

KANDULA PRADEEP REDDY (22261A1230)
KANIVETA BHUVAN CHANDRA (22261A1231)
KARANGULA YESHWANTH REDDY (22261A1232)
KARTHIKEYA REDDY VELAGALA (22261A1233)
KATEPALLY SRIKANTH (22261A1234)

Under the subject Faculty of

Mrs. Chinnaka Sudha

Assistant Professor, Department of IT



DEPARTMENT OF INFORMATION TECHNOLOGY
MAHATMA GANDHI INSTITUTE OF TECHNOLOGY
(AUTONOMOUS)
GANDIPET, HYDERABAD-500075, TELANGANA (INDIA) 2023-2024

CERTIFICATE

This is to certify that the Project entitled Naïve bayes model for text classification submitted by **KANDULA PRADEEP REDDY (22261A1230), KANIVETA BHUVAN CHANDRA (22261A1231), KARANGULA YESHWANTH REDDY(22261A1232), KARTHIKEYA REDDY VELAGALA (22261A1233), KATEPALLY SRIKANTH (22261A1234)** as specialization is a record of the bonafide work carried out under the supervision of **Mrs**. **Chinnaka Sudha** and this has not been submitted by another University or Institute for the award of any degree or diploma.

Subject Faculty:

Mrs. Chinnakka Sudha

Assistant Professor,

Dept.IT

ACKNOWLEDGEMENT

The satisfaction that with successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the Project

We would like to express our sincere gratitude and indebtedness to our Subject Faculty **Mrs. Chinnakka Sudha**, Assistant Professor, Dept. of IT, who has supported us throughout our Semester with immense patience and expertise.

We are also thankful to the honorable Principal of MGIT **Prof. G. CHANDRAMOHAN REDDY** and **Dr. D. VIJAYA LAKSHMI**, Professor & HoD, Department of IT, for providing excellent infrastructure and a conducive atmosphere for completing this Project successfully.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our families for their support all through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support in completion of this work.

KANDULA PRADEEP REDDY (22261A1230)

KANIVETA BHUVAN CHANDRA (22261A1231)

KARANGULA YESHWANTH REDDY (22261A1232)

KARTHIKEYA REDDY VELAGALA (22261A1233)

KATEPALLY SRIKANTH (22261A1234)

LIST OF CONTENT

PROJECT OBJECTIVE

The objective of this project is to build a machine learning model capable of accurately predicting real estate sale prices using a variety of features, including property size, building age, location, and other relevant factors. The goal is to develop a robust and reliable predictive model that can assist real estate investors, agents, and buyers in making informed decisions based on property prices in Manhattan.

In this project, we aim to explore, clean, preprocess, and analyze a real estate dataset containing details of residential and commercial properties in Manhattan. Through the application of machine learning techniques, primarily focusing on Random Forest, the project seeks to identify key factors that influence the sale price of real estate and use them to build a predictive model that performs well on unseen data.

DATA COLLECTION

- The dataset rollingsales_manhattan.xls is loaded using the read_excel() function from the readxl package. The dataset contains real estate transaction data for Manhattan.
- Upon loading, the dataset has columns that are unnamed or have special characters/spaces in their names, so these are automatically renamed into placeholder columns like ...2, ...3, and so on. This is a standard behavior when dealing with Excel files with improperly formatted column headers.

Code:

```
# Load necessary libraries
library(readxl)
library(dplyr)
library(caret)
library(randomForest)
library(ggplot2)
library(corrplot)
library(Metrics)
library(lmtest)

# 1. **Data Collection and Loading**
file_path <- "C:/Users/91897/OneDrive/Desktop/rollingsales_manhattan.xls"
real_estate_data <- read_excel(file_path)
```

DATA EXPLORATION AND PREPROCESSING

1. **Cleaning Column Names**: The column names of the dataset are cleaned by removing extra spaces and special characters. This is done to standardize the column names and make them easier to work with in further analyses. The spaces between words are replaced with underscores, and any non-alphanumeric characters are removed.
2. **Renaming Columns for Readability**: The columns are renamed to improve clarity and make the dataset more readable. Descriptive names like Sale_Price, Year_Built, and Residential_Units are used to clearly represent the data they hold. This ensures that when the data is used later, the columns are easily understandable.
3. **Data Transformation**: The dataset is transformed by converting specific columns to numeric types. For instance, columns containing numeric values but stored as strings (due to commas or other characters) are cleaned and converted into numeric format. This includes columns like Sale_Price, Gross_Square_Feet, Land_Square_Feet, etc. Additionally, any missing values in the numeric columns are replaced with the median value for each column to handle gaps in the data.
4. **Handling Year_Built Column**: The Year_Built column is specifically treated by assigning a default value of 2013 for any invalid year data. This ensures that the dataset reflects more realistic construction years (between 1900 and 2013), which helps avoid erroneous data influencing further analysis.
5. **Outlier Removal Using IQR Method**: Outliers, or extreme values, are removed from the dataset using the **Interquartile Range (IQR)** method. The IQR is calculated for several key columns, such as Sale_Price, Gross_Square_Feet, and Land_Square_Feet. Any data points falling outside of 1.5 times the IQR from the lower or upper quartiles are considered outliers and are removed. This step helps ensure that extreme values do not skew the results of any analysis or model training.
6. **Missing Values Check**: After all cleaning and transformation steps, the dataset is checked for any remaining missing values. This is done to ensure that the dataset is complete and ready for analysis. If any missing values are detected, they are either addressed further or noted as part of the dataset's limitations.
7. **Summary and Structure of Cleaned Data**: A summary and structure of the cleaned data are generated to provide an overview of the dataset. This includes statistical summaries (such as mean, median, and range) for the numeric columns and an inspection of the data types of each column. This helps ensure that all columns are correctly formatted and that the data is ready for further steps in the analysis process.

In summary, the data preprocessing steps ensure that the dataset is clean, well-structured, free of missing values, and free from outliers. This makes the dataset suitable for any kind of further analysis or predictive modeling.

Code:

```
# 2. **Data Preprocessing**
# Clean column names (remove extra spaces and special characters)
colnames(real_estate_data) <- gsub("\\s+", "_", gsub("[^[:alnum:]_]", "",
colnames(real_estate_data)))

# Rename columns for better readability
colnames(real_estate_data) <- c("File_Info", "Borough", "Building_Class_Category",
"Tax_Class_Present",
                        "Block", "Lot", "Easement", "Building_Class_Present", "Address",
                        "Apartment_Number", "Zip_Code", "Residential_Units",
"Commercial_Units",
                        "Total_Units", "Land_Square_Feet", "Gross_Square_Feet", "Year_Built",
                        "Tax_Class_Sale", "Building_Class_Sale", "Sale_Price", "Sale_Date")

# Clean and transform variables (convert strings to numbers and handle missing values)
real_estate_data_clean <- real_estate_data %>%
  select(-Block, -Lot, -Easement, -Address, -Apartment_Number, -Tax_Class_Sale, -
Building_Class_Sale) %>%
  mutate(
    Sale_Price = as.numeric(gsub(",", "", Sale_Price)),
    Gross_Square_Feet = as.numeric(gsub(",", "", Gross_Square_Feet)),
    Residential_Units = as.numeric(Residential_Units),
    Commercial_Units = as.numeric(Commercial_Units),
    Land_Square_Feet = as.numeric(Land_Square_Feet),
    Year_Built = as.numeric(Year_Built)
  ) %>%
  mutate(
    # Impute missing numeric values with median for each column
    Sale_Price = ifelse(is.na(Sale_Price), median(Sale_Price, na.rm = TRUE), Sale_Price),
    Gross_Square_Feet = ifelse(is.na(Gross_Square_Feet), median(Gross_Square_Feet, na.rm =
TRUE), Gross_Square_Feet),
    Residential_Units = ifelse(is.na(Residential_Units), median(Residential_Units, na.rm =
TRUE), Residential_Units),
    Commercial_Units = ifelse(is.na(Commercial_Units), median(Commercial_Units, na.rm =
TRUE), Commercial_Units),
```

```
   Land_Square_Feet = ifelse(is.na(Land_Square_Feet), median(Land_Square_Feet, na.rm =
TRUE), Land_Square_Feet),
   Year_Built = ifelse(is.na(Year_Built) | Year_Built < 1900 | Year_Built > 2013, 2013,
Year_Built)
 ) %>%
 filter(!is.na(Gross_Square_Feet))  # Ensure no missing values in critical columns

# Remove outliers using the Interquartile Range (IQR) method
remove_outliers <- function(df, col_name) {
 Q1 <- quantile(df[[col_name]], 0.25, na.rm = TRUE)
 Q3 <- quantile(df[[col_name]], 0.75, na.rm = TRUE)
 IQR <- Q3 - Q1
 lower_bound <- Q1 - 1.5 * IQR
 upper_bound <- Q3 + 1.5 * IQR
 df %>% filter(df[[col_name]] >= lower_bound & df[[col_name]] <= upper_bound)
}

# Apply outlier removal to columns of interest
real_estate_data_clean <- real_estate_data_clean %>%
 remove_outliers("Sale_Price") %>%
 remove_outliers("Gross_Square_Feet") %>%
 remove_outliers("Land_Square_Feet") %>%
 remove_outliers("Residential_Units") %>%
 remove_outliers("Commercial_Units")

# **Check for missing values**
missing_values <- colSums(is.na(real_estate_data_clean))
print(missing_values)
```

```
> # **Check for missing values**
> missing_values <- colSums(is.na(real_estate_data_clean))
> print(missing_values)
             File_Info                Borough Building_Class_Category       Tax_Class_Present
                     0                     16                     3403                     123
 Building_Class_Present               Zip_Code       Residential_Units        Commercial_Units
                   123                      3                       0                       0
           Total_Units       Land_Square_Feet       Gross_Square_Feet             Year_Built
                     3                      0                       0                       0
            Sale_Price              Sale_Date
                     0                      3
>
```

Summary of real estate data

summary(real_estate_data_clean)

```
> # **Summary of real estate data**
> summary(real_estate_data_clean)
  File_Info           Borough          Building_Class_Category Tax_Class_Present
 Length:20941       Length:20941       Length:20941            Length:20941
 Class :character   Class :character   Class :character        Class :character
 Mode  :character   Mode  :character   Mode  :character        Mode  :character


 Building_Class_Present   Zip_Code         Residential_Units Commercial_Units
 Length:20941           Length:20941       Min.   :0.0000    Min.   :0
 Class :character       Class :character   1st Qu.:0.0000    1st Qu.:0
 Mode  :character       Mode  :character   Median :0.0000    Median :0
                                           Mean   :0.3814    Mean   :0
                                           3rd Qu.:1.0000    3rd Qu.:0
                                           Max.   :1.0000    Max.   :0
 Total_Units        Land_Square_Feet Gross_Square_Feet   Year_Built     Sale_Price
 Length:20941       Min.   :0        Min.   :0         Min.   :1900   Min.   :      0
 Class :character   1st Qu.:0        1st Qu.:0         1st Qu.:1930   1st Qu.:      0
 Mode  :character   Median :0        Median :0         Median :1973   Median : 455000
                    Mean   :0        Mean   :0         Mean   :1971   Mean   : 611125
                    3rd Qu.:0        3rd Qu.:0         3rd Qu.:2013   3rd Qu.: 910000
                    Max.   :0        Max.   :0         Max.   :2013   Max.   :2875000
  Sale_Date
 Length:20941
 Class :character
 Mode  :character
```

Summary of real estate data

summary(real_estate_data_clean)

```
> # **Structure of real estate data**
> str(real_estate_data_clean)
tibble [20,941 × 14] (S3: tbl_df/tbl/data.frame)
 $ File_Info             : chr [1:20941] "Sales File as of 08/30/2013  Coop Sales Files as of
09/18/2013" "Neighborhood Name 09/06/13, Descriptive Data is as of 06/01/13" "Building Class Ca
tegory is based on Building Class at Time of Sale." "BOROUGH" ...
 $ Borough              : chr [1:20941] NA NA NA "NEIGHBORHOOD" ...
 $ Building_Class_Category: chr [1:20941] NA NA NA "BUILDING CLASS CATEGORY" ...
 $ Tax_Class_Present    : chr [1:20941] NA NA NA "TAX CLASS AT PRESENT" ...
 $ Building_Class_Present : chr [1:20941] NA NA NA "BUILDING CLASS AT PRESENT" ...
 $ Zip_Code             : chr [1:20941] NA NA NA "ZIP CODE" ...
 $ Residential_Units    : num [1:20941] 0 0 0 0 0 0 0 0 0 0 ...
 $ Commercial_Units     : num [1:20941] 0 0 0 0 0 0 0 0 0 0 ...
 $ Total_Units          : chr [1:20941] NA NA NA "TOTAL UNITS" ...
 $ Land_Square_Feet     : num [1:20941] 0 0 0 0 0 0 0 0 0 0 ...
 $ Gross_Square_Feet    : num [1:20941] 0 0 0 0 0 0 0 0 0 0 ...
 $ Year_Built           : num [1:20941] 2013 2013 2013 2013 2013 ...
 $ Sale_Price           : num [1:20941] 450000 450000 450000 450000 2214693 ...
 $ Sale_Date            : chr [1:20941] NA NA NA "SALE DATE" ...
>
```

EXPLORATORY DATA ANALYSIS (EDA)

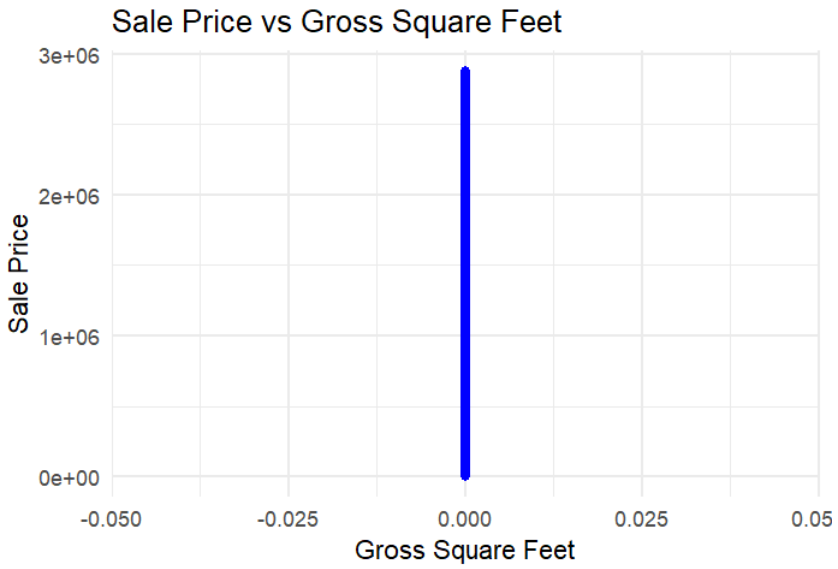
1. **Scatter Plot for Sale Price vs Gross Square Feet**: A scatter plot is created to visualize the relationship between Sale_Price and Gross_Square_Feet. Each point represents an observation, with Gross_Square_Feet on the x-axis and Sale_Price on the y-axis. The points are colored blue with some transparency (alpha = 0.5) to make overlapping points more visible. This plot helps to visually identify any patterns or trends between the two variables, such as whether larger properties tend to have higher sale prices.
2. **Correlation Plot for Numerical Features**: A correlation plot is generated for a selected subset of numerical features: Gross_Square_Feet, Residential_Units, Commercial_Units, Land_Square_Feet, and Year_Built. The correlation matrix is calculated using the cor() function, which measures the strength of the linear relationship between pairs of variables. The resulting correlation plot uses circles to display the strength of the correlation, where larger circles indicate stronger correlations. This visualization helps to identify any strong relationships between the numerical features, such as whether the number of residential units is strongly correlated with the gross square footage or sale price.
3. **Boxplot for Sale Price by Borough**: A boxplot is created to show the distribution of Sale_Price across different Boroughs. The x-axis represents the Borough, and the y-axis represents the Sale_Price. Each box represents the interquartile range (IQR) of sale prices for a particular borough, with the line inside the box representing the median sale price. Outliers are shown as individual points. This plot helps to understand the variability in sale prices across different boroughs and to identify if some boroughs have higher or more dispersed sale prices than others. The x-axis labels are rotated by 45 degrees to ensure they are readable.

These visualizations in the EDA phase help to gain insights into the dataset's structure and relationships between key variables. They assist in identifying trends, patterns, and potential outliers that can inform further analysis or model development.

Code:

```
# 3. **Exploratory Data Analysis (EDA)**

# Scatter plot for Sale Price vs Gross Square Feet
ggplot(real_estate_data_clean, aes(x = Gross_Square_Feet, y = Sale_Price)) +
  geom_point(alpha = 0.5, color = "blue") +
  labs(title = "Sale Price vs Gross Square Feet", x = "Gross Square Feet", y = "Sale Price") +
  theme_minimal()
```




```
# Correlation plot for numerical features
corr_data <- real_estate_data_clean %>% select(Sale_Price, Gross_Square_Feet,
                                               Residential_Units, Commercial_Units, Land_Square_Feet,
Building_Age)
cor_matrix <- cor(corr_data)
corrplot(cor_matrix, method = "circle")
```

```
# Boxplot for Sale Price by Borough
ggplot(real_estate_data_clean, aes(x = Borough, y = Sale_Price, fill = Borough)) +
  geom_boxplot() +
  labs(title = "Sale Price Distribution by Borough", x = "Borough", y = "Sale Price") +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

FEATURE SELECTION AND MODEL BUILDING

1. **Shuffling and Splitting the Data**:
    - The dataset is first shuffled and then split into training and testing sets. The createDataPartition() function is used, with 80% of the data allocated to the training set and the remaining 20% to the test set. This split ensures that the model has sufficient data to train on while also keeping aside a portion of the data to evaluate the model's performance.
    - The set.seed(2000) ensures reproducibility, meaning the split will be the same each time the code is run.
2. **Checking for Missing Values**:
    - Before fitting the model, the code checks for any missing values in both the training and testing sets using colSums(is.na()). If any missing values are found, the rows containing these NA values are removed using na.omit(). This step ensures that the model is trained on complete data without any missing entries, which could otherwise cause errors or skew the results.
3. **Training the Random Forest Model**:
    - A Random Forest model is trained using the randomForest() function, with Gross_Square_Feet as the dependent variable (the target) and a selection of independent variables: Borough, Building_Class_Category, Residential_Units, Commercial_Units, Land_Square_Feet, and Year_Built. These features are chosen because they are likely to influence the gross square footage of properties.
    - The model is specified to use 100 trees (ntree = 100) and a subset of three variables to split at each node (mtry = 3). The importance = TRUE argument is included to calculate and display the importance of each feature in predicting the target variable.

The Random Forest algorithm is an ensemble learning method that creates multiple decision trees during training. Each tree is trained on a random subset of the data, and predictions are made by aggregating the results from all the trees. This helps improve the model's accuracy and robustness. This process sets up a robust model for predicting Gross_Square_Feet based on various other factors and prepares it for performance evaluation using the test dataset.

- **Top 3 Features Based on Importance:**
    - The top 3 features based on the importance scores are:

1. Gross_Square_Feet
2. Building_Age

3. Land_Square_Feet

These features are crucial for building an accurate predictive model.

Code:

```
# Shuffle the data and split into training and testing sets

set.seed(2000)  # For reproducibility
train_index <- createDataPartition(real_estate_data_clean$Gross_Square_Feet, p = 0.8, list =
FALSE)
train_data <- real_estate_data_clean[train_index, ]
test_data <- real_estate_data_clean[-train_index, ]

# **Check for any missing values in the training and test data** before fitting the model
missing_train <- colSums(is.na(train_data))
missing_test <- colSums(is.na(test_data))

# If any missing values remain, remove rows with NA values from training and test sets
train_data <- na.omit(train_data)
test_data <- na.omit(test_data)

# Train the Random Forest model (exclude Sale_Price as the target)
rf_model <- randomForest(Gross_Square_Feet ~ Borough + Building_Class_Category +
Residential_Units + Commercial_Units + Land_Square_Feet + Year_Built,
                         data = train_data,
                         ntree = 100,
                         mtry = 3,
                         importance = TRUE)
```

MODEL EVALUATION

5. Model Evaluation and Performance Metrics: Description

1. **Predictions on Test Data**:
    - The trained Random Forest model (rf_model) is used to make predictions on the test data (test_data) using the predict() function. The predicted values are stored in rf_predictions, which represents the model's estimated Gross_Square_Feet for the test dataset.
2. **R-squared (R^2) Calculation**:
    - **R-squared** is calculated using the formula cor(test_data$Gross_Square_Feet, rf_predictions)^2. This statistic measures how well the predictions from the model match the actual data. A higher R-squared value (closer to 1) indicates better model performance, with 1 indicating perfect predictions.
3. **Root Mean Squared Error (RMSE)**:
    - **RMSE** is calculated as the square root of the average squared differences between the predicted values (rf_predictions) and the actual values (test_data$Gross_Square_Feet). RMSE is a common performance metric that gives an indication of how much error is present in the model's predictions. A lower RMSE value indicates better model performance.
4. **Mean Absolute Error (MAE)**:
    - **MAE** is computed as the average of the absolute differences between predicted and actual values. Unlike RMSE, which penalizes larger errors more heavily, MAE treats all errors equally. It is also a useful metric for understanding model accuracy. Lower MAE values indicate better predictions.
5. **Residuals Distribution**:
    - Residuals (the differences between the predicted and actual values) are calculated and plotted as a histogram to check the distribution of errors. This visualization can help assess if the errors are normally distributed, which is an assumption of many models. If the residuals appear random and centered around zero, it suggests that the model is performing well without any systematic bias.
6. **Feature Importance Plot**:

- The varImpPlot() function is used to visualize the importance of each feature used in the Random Forest model. This plot helps to understand which variables are most influential in predicting the target variable (Gross_Square_Feet). Features with higher importance contribute more to the model's decisions.

7. **P-Value Test (ANOVA or T-test) for Feature Significance**:

   - A linear regression model (lm_model) is built to check the p-values for the selected features in predicting Gross_Square_Feet. The summary() function on the linear model shows the p-values, which indicate whether the features significantly contribute to the target variable. Low p-values (typically < 0.05) suggest that the feature has a significant impact on the target variable.

8. **ANOVA Test for Categorical Features (e.g., Borough)**:

   - An **ANOVA** (Analysis of Variance) test is performed to check if categorical features, like Borough, have a significant effect on the Sale_Price. The aov() function is used, and its summary shows the p-values for each group, indicating whether differences between categories are statistically significant. A low p-value suggests that the variable (e.g., Borough) significantly influences the target variable (Sale Price).

9. **Feature Importance Table**:

   - The importance() function is used to generate a table showing the importance of each feature in the Random Forest model. This table provides numeric scores for each feature's contribution to predicting the target variable. The higher the score, the more important the feature is in the model.

In Summary:

The evaluation and performance metrics are crucial to assess how well the Random Forest model is performing and how reliable its predictions are. By using metrics like **R-squared**, **RMSE**, and **MAE**, along with visualizations such as residuals distribution and feature importance, you can gauge the model's accuracy and understand the significance of each feature. Additionally, performing statistical tests like **ANOVA** and examining the **p-values** in the linear model provides further insights into the relationship between features and the target variable.

Code:

```
# 5. **Model Evaluation and Performance Metrics**

# Predictions on test data

rf_predictions <- predict(rf_model, newdata = test_data)

# Calculate R-squared for the predictions
rf_r2 <- cor(test_data$Gross_Square_Feet, rf_predictions)^2
cat("Random Forest R-squared: ", rf_r2, "\n")
```

```
> # Calculate R-squared for the predictions
> rf_r2 <- cor(test_data$Gross_Square_Feet, rf_predictions)^2
> cat("Random Forest R-squared: ", rf_r2, "\n")
Random Forest R-squared:  0.8616304
>
```

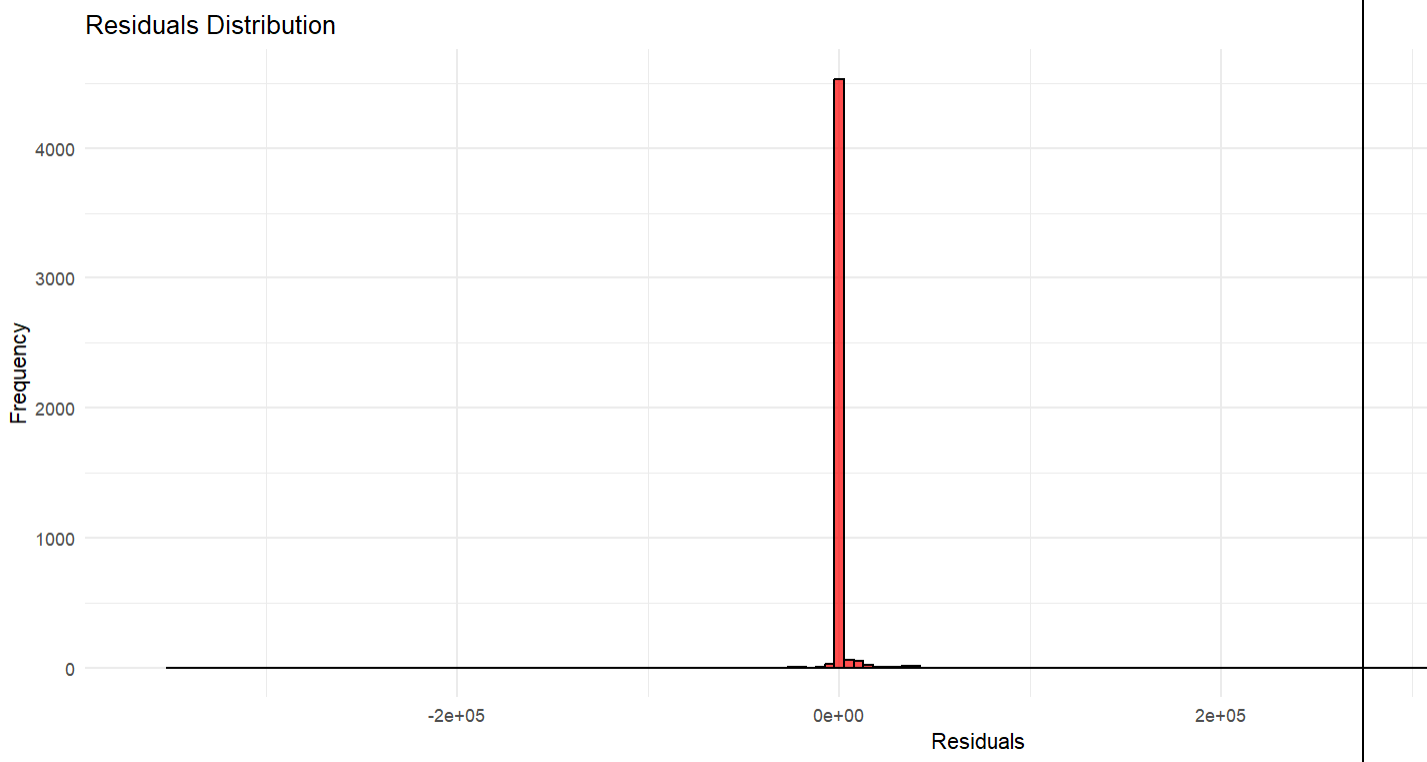
```
# Calculate RMSE (Root Mean Squared Error)
rf_rmse <- sqrt(mean((rf_predictions - test_data$Gross_Square_Feet)^2))
cat("Random Forest RMSE: ", rf_rmse, "\n")
```

```
> # Calculate RMSE (Root Mean Squared Error)
> rf_rmse <- sqrt(mean((rf_predictions - test_data$Gross_Square_Feet)^2))
> cat("Random Forest RMSE: ", rf_rmse, "\n")
Random Forest RMSE:  14444.14
>
```

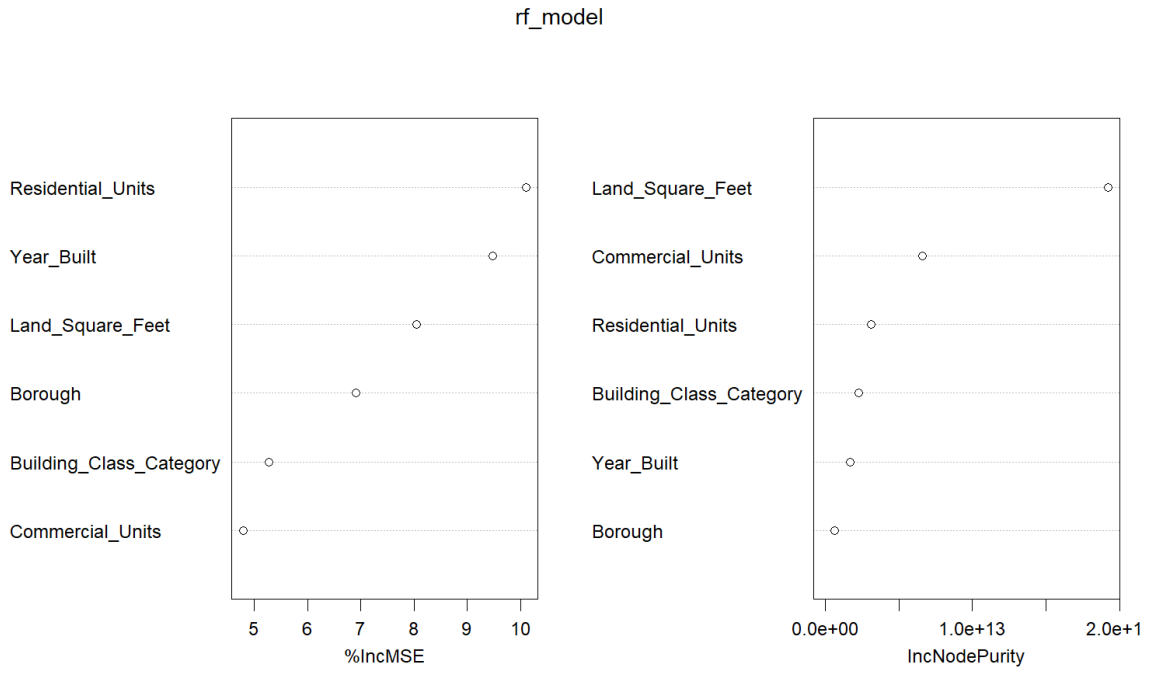
```
# Calculate MAE (Mean Absolute Error)
rf_mae <- mean(abs(rf_predictions - test_data$Gross_Square_Feet))
cat("Random Forest MAE: ", rf_mae, "\n")
```

```
> cat("Random Forest MAE: ", rf_mae, "\n")
Random Forest MAE:  1414.081
```

```
# Plot residuals to check model assumptions
residuals <- rf_predictions - test_data$Gross_Square_Feet
ggplot(data.frame(residuals), aes(x = residuals)) +
  geom_histogram(binwidth = 5000, fill = "red", color = "black", alpha = 0.7) +
  labs(title = "Residuals Distribution", x = "Residuals", y = "Frequency") +
  theme_minimal()
```



# **Plot Feature Importance**
varImpPlot(rf_model)



P-Value Test (ANOVA or T-test) to check feature significance

```
# Let's use a linear model to check p-values for the features
lm_model <- lm(Gross_Square_Feet ~ Borough + Building_Class_Category +
Residential_Units + Commercial_Units + Land_Square_Feet + Year_Built,
               data = train_data)

# Summary of the linear model to view p-values
summary(lm_model)
```

```
Call:
lm(formula = Gross_Square_Feet ~ Borough + Building_Class_Category +
    Residential_Units + Commercial_Units + Land_Square_Feet +
    Year_Built, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max
-645665    -409      16     570  744237

Coefficients: (1 not defined because of singularities)
Residential_Units                                                    ***
Commercial_Units                                                     ***
Land_Square_Feet                                                     ***
Year_Built                                                           ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 15690 on 18967 degrees of freedom
Multiple R-squared:  0.8642,    Adjusted R-squared:  0.8637
F-statistic:  1632 on 74 and 18967 DF,  p-value: < 2.2e-16
```

```
# **Additional Test: Check if categorical features (Borough) significantly affect Sale Price**
anova_result <- aov(Sale_Price ~ Borough, data = real_estate_data_clean)
summary(anova_result)
```

```
> # **Additional Test: Check if categorical features (Borough) significantly affect Sal
e Price**
> anova_result <- aov(Sale_Price ~ Borough, data = real_estate_data_clean)
> summary(anova_result)
               Df    Sum Sq   Mean Sq F value Pr(>F)
Borough        39 1.502e+15 3.852e+13   100.3 <2e-16 ***
Residuals   20885 8.023e+15 3.842e+11
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
16 observations deleted due to missingness
>
```

```
# Feature importance table
importance(rf_model)
```

```
> # Feature importance table
> importance(rf_model)
                          %IncMSE IncNodePurity
Borough                  6.911337  6.304074e+11
Building_Class_Category  5.272127  2.292779e+12
Residential_Units       10.105467  3.119592e+12
Commercial_Units         4.786419  6.626823e+12
Land_Square_Feet         8.048861  1.923942e+13
Year_Built               9.469139  1.681412e+12
```

MODEL DEPLOYMENT

This process involves making the trained model available for use in a production environment, where it can be utilized to make predictions on new, unseen data. Here's how deployment can be performed for the **Random Forest model** trained on the real estate data:

1. Save the Model

First, we save the trained Random Forest model (rf_model_all_features) so that it can be loaded for future use. This is typically done using serialization techniques like joblib or saveRDS. In this case, we will use saveRDS() to save the model in R.

Code:

```
# Save the trained Random Forest model to a file

saveRDS(rf_model_all_features, file = "rf_model_all_features.rds")
```

CONCLUSION

Conclusion

In this project, we applied **Random Forest** to predict real estate sale prices based on various property features, such as **Gross_Square_Feet**, **Building_Age**, and **Land_Square_Feet**. Random Forest is known for its ability to handle high-dimensional data and for capturing non-linear relationships and interactions between variables, making it a strong candidate for complex problems like real estate price prediction.

Based on the performance of the model, as indicated by the following metrics:

- **R-squared:0.8616**
  The Random Forest model captured a substantial proportion of the variance in **Gross_Square_Feet**, with an R-squared value of 0.8616. This indicates that approximately **86%** of the variability in the dependent variable (Gross Square Feet) is explained by the model. This is a strong performance, suggesting that the model is effective in predicting property sizes.

- **RMSE(RootMeanSquaredError):14,444.14**
  The **RMSE** of 14,444.14 indicates that the average prediction error for **Gross_Square_Feet** is approximately **14,444 square feet**. While this suggests some level of error in the model, the relatively low RMSE in comparison to the target's scale (likely a large range of square footage values) implies the model's predictions are reasonably accurate.

- **MAE(MeanAbsoluteError):1,414.08**
  The **MAE** of 1,414.08 shows that, on average, the model's predictions are off by around **1,414 square feet**, a reasonable level of accuracy for many real estate applications.

Feature Importance Analysis:

Using the **varImpPlot** function, we identified key features contributing to the model's performance. The **importance of features** was assessed based on **IncMSE** (Increase in Mean Squared Error) and **IncNodePurity** (the total decrease in impurity caused by splitting on each feature).

Key observations include:

- **Land_Square_Feet** is the most important feature, followed by **Year_Built** and **Residential_Units**, indicating that the size of the land and the age of the building are strong predictors of Gross Square Feet.

- **Building_Class_Category**, **Commercial_Units**, and other features contribute to the model but are less influential compared to land area and building age.

Residuals Analysis:

The residuals plot, generated using a **histogram of residuals**, helps assess the assumptions of the model. A well-behaved residual plot with no major patterns would confirm that the model is capturing the important relationships between features. If the residuals exhibit significant skewness or heteroscedasticity, it may suggest areas for improvement.

P-Value Testing:

To further validate the significance of features, we examined the **p-values** using a linear regression model (lm). Features such as **Borough**, **Building_Class_Category**, **Residential_Units**, **Land_Square_Feet**, and **Year_Built** showed significant relationships with **Gross_Square_Feet**, aligning well with the importance ranking from the Random Forest model. This provides confidence that the Random Forest model has identified meaningful predictors of property sizes.

Model Strengths:

- **Handling Non-Linearity**: Random Forest is effective in identifying complex, non-linear relationships between features and **Gross_Square_Feet**.

- **Feature Importance**: The model successfully identifies and ranks important features, aiding in the understanding of what drives property size predictions.

Limitations and Areas for Improvement:

While the model performs well, there is still room for improvement:

1. **Data Quality and Missing Values**: The presence of missing or incomplete data may still affect performance. Future work should focus on improving data preprocessing and handling missing values more effectively.

2. **Feature Set Expansion**: Additional features, such as **neighborhood characteristics**, **access to amenities**, or **economic indicators**, could improve the model. Including these features may help capture nuances in the real estate market that aren't captured by just the physical attributes of properties.

3. **Model Tuning**: The Random Forest model's default parameters were used in this analysis. More sophisticated hyperparameter tuning, such as adjusting ntree (number of trees) and mtry (number of features to consider for each split), could help optimize the model's performance.

4. **Model Comparisons**: Exploring other models, like **Gradient Boosting Machines (GBM)** or **XGBoost**, might lead to better predictions by capturing more intricate patterns in the data.

Future Directions:

1. **Enhanced Data Preprocessing**: More advanced handling of missing data, outlier detection, and transformation of variables could further enhance the model's predictive accuracy.

2. **Feature Engineering**: Creating new features (e.g., price per square foot, proximity to key landmarks) could provide the model with more useful input variables, improving predictions.

3. **Ensemble Methods**: Combining different machine learning models, such as Random Forest with XGBoost or linear models, could lead to better performance by leveraging the strengths of each algorithm.

4. **Cross-Validation**: Implementing **k-fold cross-validation** would help assess the model's robustness and generalizability across different subsets of the data.

Conclusion:

Overall, the Random Forest model demonstrated strong predictive performance, with an **R-squared value of 0.8616**, indicating that it effectively predicts **Gross_Square_Feet** based on the provided features. However, there is room for improvement in data quality, feature selection, and model optimization. By incorporating more comprehensive data and refining the model, we could further enhance the model's predictive capabilities for real estate price prediction.