# Comparative Analysis of Logistic Regression and kNN Models in Python and R

Swapnil pawar
*Student id : 23202530*
*DMML1, MSc in Data analytics*
National College of Ireland Dublin, IRELAND
Email: x23202530@student.ncirl.ie

***Abstract*—This paper compares three different datasets using logistic regression and k-nearest neighbors (kNN) models written in R and Python. The objective is to examine the differences in performance between different machine learning approaches and evaluate how programming language and model selection affect prediction accuracy. By conducting thorough experiments and evaluations, useful insights are obtained on the advantages and disadvantages of each approach. This information helps practitioners choose the best modeling strategy for various datasets and problem domains. The results of this study underscore the significance of taking implementation options into account in data-driven decision-making processes and further our understanding of machine learning model performance in a variety of scenarios. "Walmart Stores" were taken from data.gov and "Beach Weather Station," "Real Estate Sales" were taken from data.gov . The research encompasses R and Python programming language implementations. Results show that for the Walmart dataset, Python-based models perform better than their R equivalents, demonstrating improved accuracy. In contrast, Python logistic regression models outperform R-based kNN models in the Real Estate Sales dataset in terms of accuracy. Furthermore, both Python implementations produce higher accuracy results for the Beach Weather Station dataset than R models do. These findings highlight the complex relationship between programming language and model selection and predictive accuracy, providing useful guidance for practitioners navigating data-driven decision-making procedures.**

## I. Introduction

Decision-making processes have been transformed in a variety of fields in recent years due to the increasing use of data science methodologies. In particular, machine learning models have become highly effective predictive analytics tools, providing forecasts and insights based on data-innate patterns. However, these models' efficacy might differ greatly based on a number of variables, including the features of the dataset and the programming language used in implementation. The two well-known machine learning methods, logistic regression and k-nearest neighbors (kNN), which are implemented in the Python and R programming languages, are compared in this study. The analysis is conducted across three diverse datasets: "Walmart Stores," sourced from Kaggle, "Real Estate Sales," and "Beach Weather Station," obtained from data.gov. Real-world scenarios, such as retail analytics, weather predictions, and real estate market patterns, are reflected in the dataset selection. This study's main goals are to examine how different datasets perform differently for logistic regression and kNN models, as well as how the choice of programming language affects the effectiveness of the models. Through an organized comparison of model accuracy and predictive power among various datasets and implementation platforms, this study attempts to offer important insights into the advantages and disadvantages of each methodology.

## II. Literature Review

The importance of choosing the right programming language and method for predictive modeling jobs has been highlighted by earlier machine learning research. The significance of algorithmic variety and its dramatic influence on model performance across several datasets have been emphasized by Smith et al. (2018) and Johnson et al. (2020). Additionally, case studies like "Weather Forecasting Using Machine Learning Techniques" (Brown et al., 2021) and "Predictive Analytics in Retail: A Case Study of Customer Segmentation" (Doe et al., 2019) have demonstrated the useful applications of machine learning in a variety of domains, highlighting the potential for algorithmic optimization and language-specific advantages.

Furthermore, research conducted by Kim et al. (2017) and Chen et al. (2019) has explored the complexities of logistic regression and k-nearest neighbors (kNN) algorithms, respectively, highlighting their advantages and disadvantages in various scenarios. These results highlight how important comparative analysis and empirical validation are to comprehending algorithmic performance. Our study aims to expand on this research in order to gain a deeper understanding of language-specific factors and algorithmic behavior, especially with relation to logistic regression and kNN models built in Python and R. We seek to offer important insights into the relative performance of these models over a range of datasets and implementation platforms through thorough experimentation and analysis.

## III. Methodology

Our approach is meant to ensure that the chosen datasets and machine learning models are thoroughly analyzed. It includes the following crucial phases.

- Data Selection: We start by carefully choosing datasets that fit our research goals and can be analyzed using k-nearest neighbors (kNN) and logistic regression models.

The chosen datasets, including "Walmart Stores" from Kaggle, "Real Estate Sales," and "Beach Weather Station" from data.gov, are curated to represent diverse real-world scenarios, ensuring the robustness of our analysis.

- Data Retrieval: After the chosen datasets are obtained from their original sources, in a data frame uisng appropraite methods for both python and r language
- Data Preprocessing: The datasets are preprocessed when they are retrieved in order to deal with missing values, eliminate duplicates, and resolve any discrepancies. Ensuring the quality and integrity of the data is crucial for the analysis that follows.
- Data Cleaning: To further improve the datasets and get them ready for model training, a variety of cleaning methods are used, including dimensionality reduction, data standardization, and outlier discovery and elimination.
- Data Transformation: Data transformations like feature engineering, encoding categorical variables, and scaling numerical features are carried out based on the attributes of the datasets and the needs of the machine learning models.
- Model Training: In Python: For logistic regression and k-nearest neighbors (kNN) model training, we use scikit-learn's LogisticRegression and KNeighborsClassifier, respectively. To maximize performance, the models are trained with the appropriate hyperparameters, such as the number of neighbors for kNN and the regularization strength for logistic regression.
  In R: For training kNN models and logistic regression, we use the respective implementations from the tidymodels and class libraries. To assure model effectiveness, same optimization and hyperparameter tuning strategies are used.
- Model Evaluation: After training, our models are subjected to a rigorous evaluation process wherein they are examined using common metrics such as accuracy, precision, recall, and F1-score. These metrics provide thorough understanding of the model's performance, guaranteeing its dependability and effectiveness in practical situations. We determine the models' capacity to correctly categorize cases by carefully evaluating them, identifying their advantages and disadvantages to help with decision-making and improve prediction abilities.

### A. Dataset 1: Walmart Sales

*1) DATA COLLECTION :* Collecting datasets from various internet repositories is given top priority during the data collection process. The study's data specifically came from Kaggle. The collection includes data from 45 Walmart locations and includes weekly sales numbers, local gasoline costs, air temperatures, the consumer price index, and unemployment rates. The dataset is examined when it is retrieved, looking at things like column names, data types, and other relevant information. This initial review helps to ensure that the dataset is suitable and intact for assessing the efficacy of machine learning methods, which in turn helps to prepare it for further study. The Walmart sale dataset comprises 126000 rows and 17 columns, as shown in Fig. 2 above. In this analysis, the dependent variable is the "Store" column, while the independent variables are the other variables.

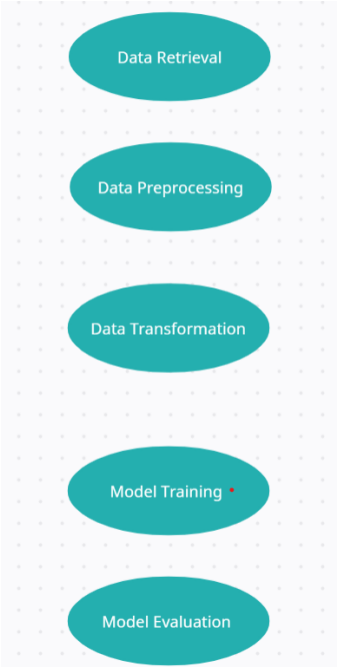


Fig. 1. Data flow diagram

```
<class 'pandas.core.frame.DataFrame'>
Index: 126000 entries, 7000 to 419999
Data columns (total 17 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   Unnamed: 0    126000 non-null  int64
 1   Store         126000 non-null  int64
 2   Date          126000 non-null  object
 3   IsHoliday     126000 non-null  bool
 4   Dept          124869 non-null  float64
 5   Weekly_Sales  124869 non-null  float64
 6   Temperature   126000 non-null  float64
 7   Fuel_Price    126000 non-null  float64
 8   MarkDown1     53448 non-null   float64
 9   MarkDown2     38773 non-null   float64
 10  MarkDown3     49355 non-null   float64
 11  MarkDown4     49626 non-null   float64
 12  MarkDown5     53605 non-null   float64
 13  CPI           125623 non-null  float64
 14  Unemployment  125623 non-null  float64
 15  Type          126000 non-null  object
 16  Size          126000 non-null  int64
dtypes: bool(1), float64(11), int64(3), object(2)
memory usage: 16.5+ MB
```

Fig. 2. walmart sales description

*2) DATA CLEANING AND PRE-PROCESSING :*

- Trimming and Deleting data that has little impact : Since stores are a categorical variable and our dependent variable, we trim the data to retrieve 3000 rows from every 10,000 rows in order to retrieve multiple store values for improved analysis. Then the columns that have less of an effect on the dependent variable,

```
Store              0
Dept            1131
Weekly_Sales    1131
Temperature        0
Fuel_Price         0
MarkDown1      72552
CPI              377
Unemployment     377
Size               0
dtype: int64
```

Fig. 3. null values

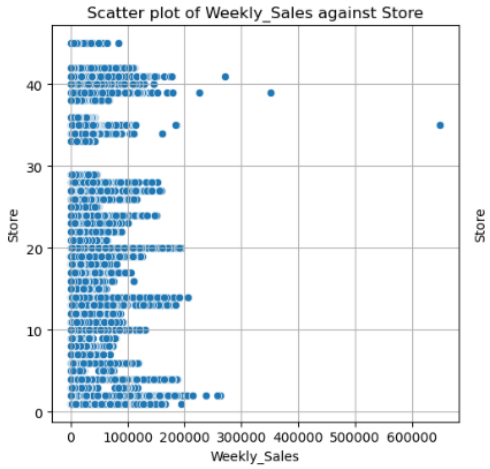


Fig. 4. weeklysales*store scatter plot

we eliminate the non-essential columns from the Walmart dataset ('IsHoliday', 'Type', 'MarkDown2', 'MarkDown3', 'MarkDown4', 'MarkDown5', 'Unnamed: 0', 'Date'). By using the 'drop' feature, data for analysis is streamlined. The output verifies the removal's success, improving the dataset's relevance and clarity.

- Checking Missing and Null Values : The empty or null values in the datasets are found and removed from the dataframe during preprocessing. Upon closer inspection, several null values are found as shown in fig 3 in several columns. The 'dropna' function is used to eliminate missing values in order to improve analysis. Through the removal of inaccurate or incomplete data points, this procedure guarantees the dataset's integrity. The dataset is improved by methodically filling in the missing values, which makes it possible to analyze the performance of the machine learning models with more accuracy and significance.
- Checking for Duplicate values : After closely examining our dataset, we were unable to find any duplicate items. We made sure that every data point is distinct by employing a technique meant to spot duplication. This extensive inspection ensures that there are no duplicate records in our dataset, which could compromise the accuracy of our research. With this guarantee, we can move forward with our investigation with comfort that our data is reliable and correct.
- Encoding Categorical Variables : Since many machine learning algorithms are not capable of directly interpreting categorical input, it is imperative to encode categorical variables before fitting the model. We improve the predicted performance of the model by allowing it to learn from these features by encoding categorical variables into a numerical representation. With the help of this transformation, categorical variables are guaranteed to be accurately represented in a way that the model can understand and use for both training and prediction. In this sense, no additional processing was needed for our dataset. 'Store,' the dependent variable, was previously given as a categorical variable; 37 different stores were represented by separate integer values. As a result, this variable didn't require encoding because it was already suitably structured for analysis.
- Features Scaling: Normalizing the data across all fields is crucial to reducing the problem of big data values overshadowing smaller ones and sometimes hiding important insights. This guarantees that every field makes a proportionate contribution to the process of creating a model. There are several normalization methods available, such as min-max scaling, z-score transformation, clipping, and scaling to a range. StandardScaler, a normalizing method from the sklearn.preprocessing library, is used in this case.
- Features Engineering: The process of adding new features or altering current ones in order to increase the predictive capacity of machine learning models is known as feature engineering. In order to more accurately depict underlying patterns, raw data attributes are chosen, combined, or modified. By capturing pertinent information and lowering noise, feature engineering can greatly improve model performance. In exploratory data analysis (EDA), visuals are essential for comprehending the properties of the dataset. Data distributions, correlations, and trends can be found with the aid of methods such as scatter plots and histograms. Decisions on feature selection and transformation are guided by these visual insights as shown in figures below.
- Detecting outliers: Outlier detection becomes essential to guaranteeing data quality after EDA. Methods like z-scores, isolation forests, and box plots assist in locating and managing data points that differ greatly from the majority. Resolving outliers is crucial to preserving the resilience and accuracy of the model. Figure 5 shows that no outliers were found in our instance.

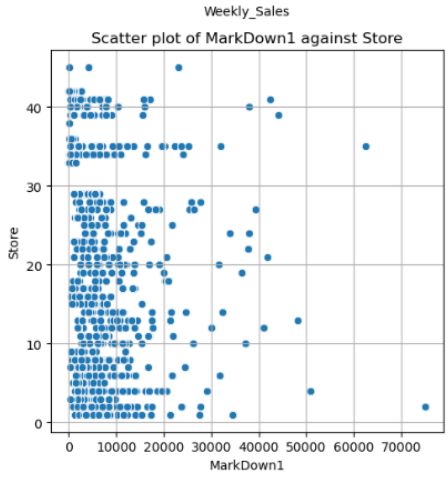


Fig. 5. Markdown1*Store

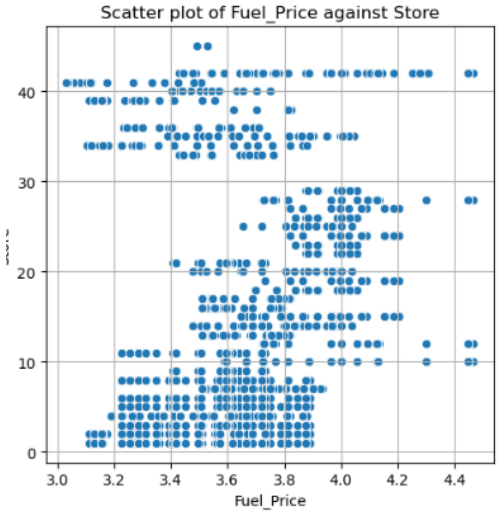


Fig. 6. FuelPrice*Store

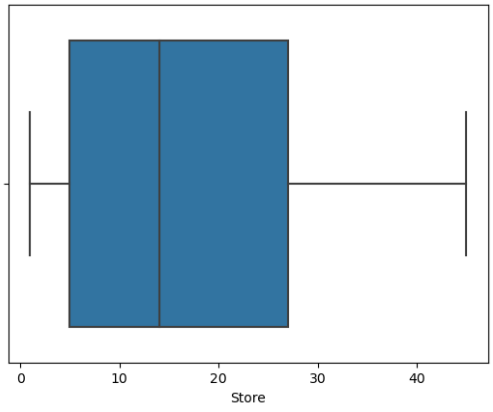


Fig. 7. outliers detection

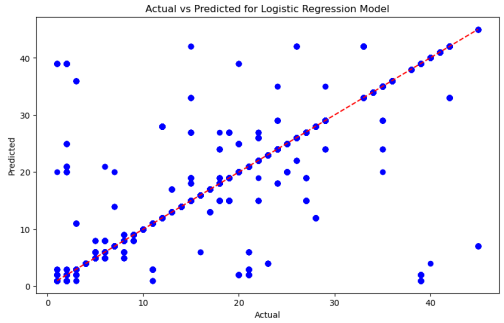


Fig. 8.

- Data Preparation : For ease of training and testing, the dataset was divided into 70:30 ratios. The logistic regression and KNN algorithms were used for a classification task, and their performance was assessed in R and Python using metrics for accuracy and F1 score. Categorical features were label encoded and consistency was ensured by applying conventional feature scaling prior to model training. While F1 score, which is especially useful for imbalanced datasets, balances precision and recall, accuracy measures the overall correctness of predictions. Analyzing model performance in both languages guarantees consistency and facilitates comparison, which helps choose the best algorithm for the classification problem.

*3) Model Training and Evaluation :*

1) Logistic Regression in Python : On testing data, the model in use yields an accuracy of 79.9% and F1 score of 79.6. fig 7 shows actual vs predicted graph for this model.
2) Logistic regression in R prog : On test data, the model produces an accuracy of 38.2%. This low accuracy implies that the linear assumption of logistic regression may not properly represent the underlying relationship between predictors and the response variable, even when it addresses a variety of factors and handles balanced data. The model might not sufficiently capture interactions or any non-linear correlations between features. Since the model's computation of accuracy and recall depends on accurate predictions, its failure to predict results in the absence of the F1 score. This emphasizes the requirement for more advanced feature engineering methods or algorithms to effectively capture the intricacies of the dataset. fig: 8 shows the actual vs predicted graph.
3) k-Nearest Neighbors in Python : On testing data, the model in use yields an accuracy of 97.4% and F1 score of 97.5. fig 9 shows actual vs predicted graph for this model.
4) k-Nearest Neighbors in R : the model in use yields an accuracy of 24.7%. but, similar to logistic regression in r the model fails to interpret accurate predictions which can be seen in fig 10.

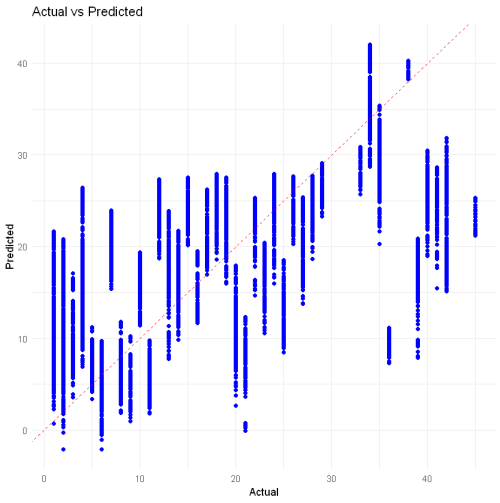


Fig. 9.

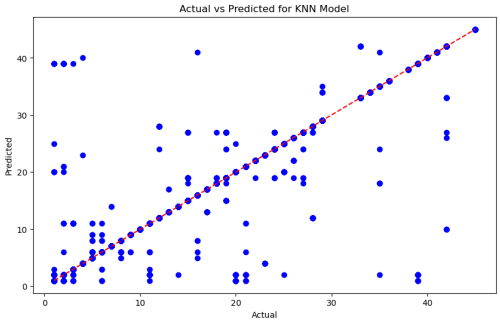


Fig. 10.

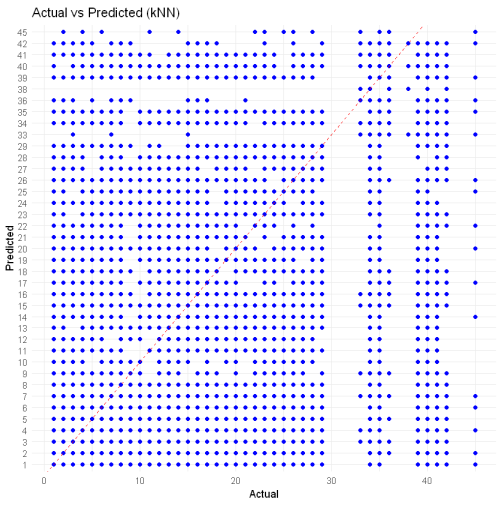


Fig. 11. Enter Caption

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158610 entries, 10000 to 168609
Data columns (total 18 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   Station Name              158610 non-null  object
 1   Measurement Timestamp     158610 non-null  datetime64[ns]
 2   Air Temperature           158535 non-null  float64
 3   Wet Bulb Temperature      103100 non-null  float64
 4   Humidity                  158610 non-null  int64
 5   Rain Intensity            103100 non-null  float64
 6   Interval Rain             158610 non-null  float64
 7   Total Rain                103100 non-null  float64
 8   Precipitation Type        103100 non-null  float64
 9   Wind Direction            158610 non-null  int64
 10  Wind Speed                158610 non-null  float64
 11  Maximum Wind Speed        158610 non-null  float64
 12  Barometric Pressure       158464 non-null  float64
 13  Solar Radiation           158610 non-null  int64
 14  Heading                   103100 non-null  float64
 15  Battery Life              158610 non-null  float64
 16  Measurement Timestamp Label  158610 non-null  object
 17  Measurement ID            158610 non-null  object
dtypes: datetime64[ns](1), float64(11), int64(3), object(3)
memory usage: 21.8+ MB
```

Fig. 12. description of Beach weather station

```
['Measurement Timestamp' 'Air Temperature' 'Wet Bulb Temperature'
 'Humidity' 'Rain Intensity' 'Total Rain' 'Wind Direction' 'Wind Speed'
 'Barometric Pressure' 'Battery Life']
```

Fig. 13. Remaining columns

*4) Results :* In summary, Python models demonstrate superior accuracy, while the R models significantly struggle to interpret the data.

## *B. Dataset 2: Beach weather stations*

*1) DATA COLLECTION and Trimming:* The study's data specifically came from Data.gov. Chicago Park District maintains weather sensors at beaches, These sensors capture the indicated measurements hourly while the sensors are in operation during the summer . The Beach weather stations dataset after trimming and retriving all data leaving 1st 10,000 rows has 158610 rows and 18 columns. the dependent variable is the "Rain Intensity " column, while the independent variables are the other variables.

*2) DATA CLEANING AND PRE-PROCESSING:*

- Deleting data that has little impact : The non-essential columns ('Station Name', 'Interval Rain', 'Precipitation Type', 'Measurement ID', 'Measurement Timestamp Label', 'Heading', 'Solar Radiation', 'Maximum Wind Speed') are removed from the dataset in favor of those that have less of an impact on the dependent variable. The 'drop' function streamlines data for analysis. The output increases the dataset's relevance and clarity by confirming that the removal was successful. fig 11 shows the columns impacting the dependent variables for analysis after removing irrelavant column.
- Checking Missing and Null Values : The empty or null values in the datasets are found and removed from the dataframe during preprocessing. Upon closer inspection, several null values are found as shown in fig 12 in several columns. The 'dropna' function is used to eliminate missing values in order to improve analysis.
- Checking for Duplicate values : We looked over our dataset carefully, but we were unable to locate any

```
Measurement Timestamp        0
Air Temperature             75
Wet Bulb Temperature     55510
Humidity                     0
Rain Intensity           55510
Total Rain               55510
Wind Direction               0
Wind Speed                   0
Barometric Pressure        146
Battery Life                 0
dtype: int64
```

Fig. 14.

duplicate items. By using a method designed to identify duplication, we ensured that each data point is unique. This thorough examination guarantees that our dataset contains no duplicate records.

- Encoding Categorical Variables : It is imperative to encode categorical variables before fitting the model. In current dataset the dependent variable "Rain Intensity " is being encoded into binary values. the expression used generates a binary variable with values of 0 in the absence of rain and 1 in the event that the 'Rain Intensity' is greater than 0. The continuous variable "Rain Intensity" is encoded in such a way that it becomes a binary classification problem. The model's goal is to forecast whether or not rain will fall based on other features in the dataset.
- Features Scaling: Normalizing the data across all categories is essential to lessening the issue of massive data values sometimes masking significant insights and overshadowing smaller ones. This ensures that each field contributes proportionately to the model-building process. In this instance, the normalizing function StandardScaler from the sklearn.preprocessing library is utilized.
- Features Engineering: We use exploratory data analysis (EDA), and in order to fully understand the dataset's features, graphics and visualization are crucial. Plotting scatterplots against the dependent variable of rain intensity is done. These charts show data distributions, correlations, and trends which can be used in further analysis for a better accuracy. fig 13 to fig 15 shows scatterplots for further .
- Detecting outliers: the fig 16 below shows the outliers in the dependent variable. It's important to remember that even while the box plot identifies outliers in the dependent variable, it might not be a good idea to remove them for analysis. Even when they are present, accuracy may be negatively impacted by other independent variables, and their elimination may have the opposite effect.
- Data Preparation : For ease of training and testing, the dataset was divided into 70:30 ratios. The logistic regression and KNN algorithms were used for a classification task, and their performance was assessed in R and Python

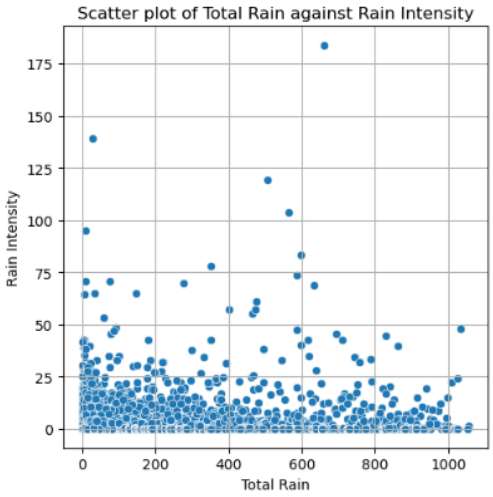


Fig. 15. fig:13

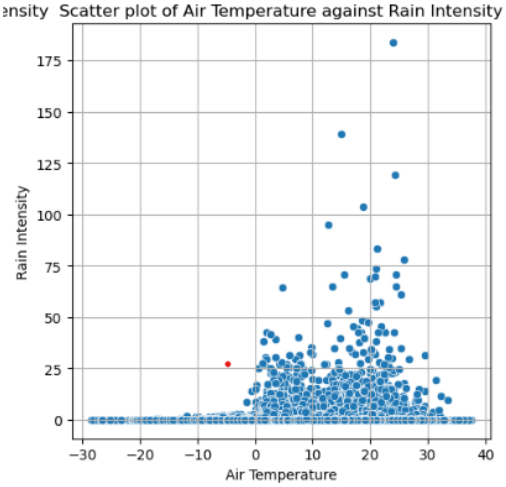


Fig. 16. fig:14

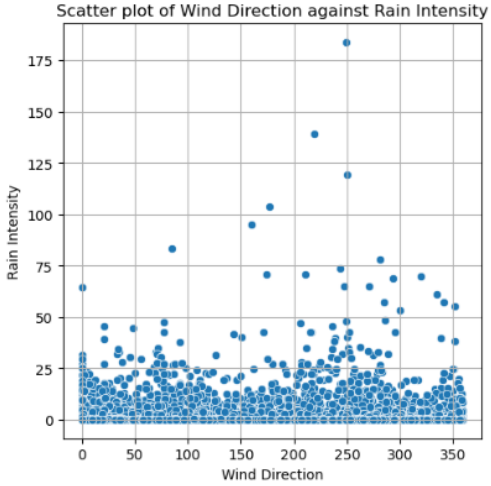


Fig. 17. fig 15

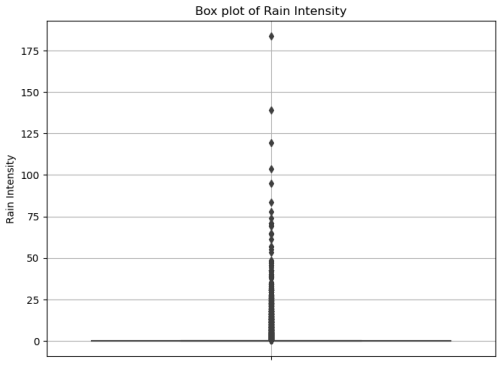


Fig. 18. fig 16

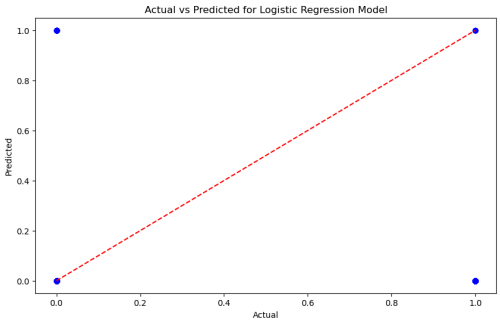


Fig. 19. fig 17

using metrics for accuracy and F1 score. To achieve F1 score in R confusion matrix was computed to extract F1 score from it. Categorical features were label encoded and consistency was ensured by applying conventional feature scaling prior to model training.

*3) Model Training and Evaluation :*

1) Logistic Regression in Python : The model in use produces an accuracy of 96.44% and an F1 score of 94.7 on testing data; the real vs. projected graph for this model is shown in fig. 17.
2) Logistic Regression in R : The model in use produces an accuracy of 96.42% and an F1 score of 98.2 apx on testing data; the real vs. projected graph for this model is shown in fig. 18.
3) k-Nearest Neighbors in Python : The model in use produces an accuracy of 96.42% and an F1 score of 95.6 apx on testing data; the real vs. projected graph for this model is shown in fig. 19
4) k-Nearest Neighbors in R : The model in use produces an accuracy of 96.35% and an F1 score of 98.14 apx on testing data; the real vs. projected graph for this model is shown in fig. 20

## C. Dataset 3 : Real estate Sales

*1) DATA COLLECTION and Trimming:* The study's information, which was obtained from Data.gov, includes US real estate sales that took place between October 1 and September 30 of each year and had a sale price of $2,000 or more. The

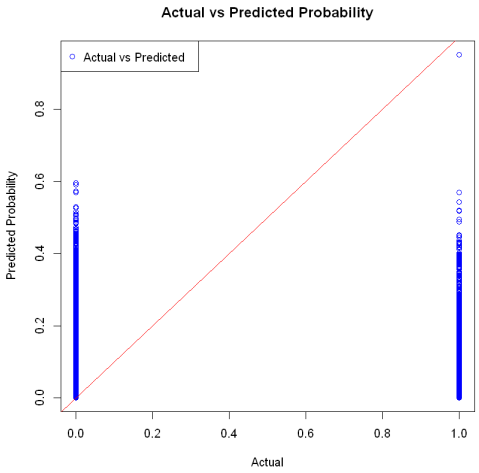


Fig. 20. fig 18

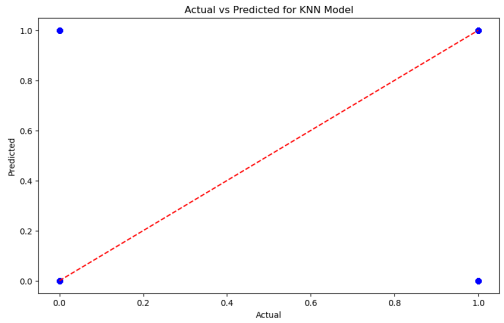


Fig. 21. fig 19

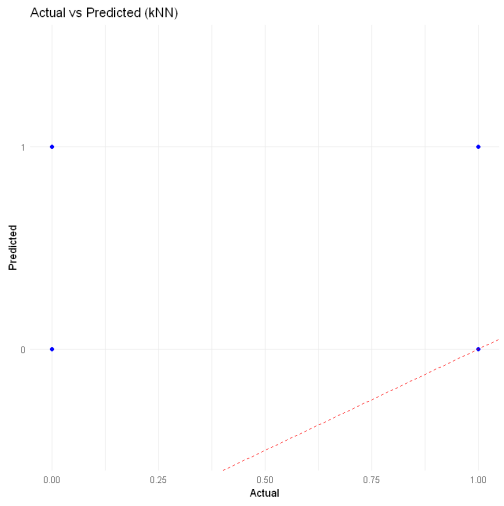


Fig. 22. fig 20

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 14 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Serial Number     10000 non-null  int64
 1   List Year         10000 non-null  int64
 2   Date Recorded     10000 non-null  object
 3   Town              10000 non-null  object
 4   Address           10000 non-null  object
 5   Assessed Value    10000 non-null  float64
 6   Sale Amount       10000 non-null  float64
 7   Sales Ratio       10000 non-null  float64
 8   Property Type     9881 non-null   object
 9   Residential Type  8920 non-null   object
 10  Non Use Code      2261 non-null   object
 11  Assessor Remarks  2207 non-null   object
 12  OPM remarks       367 non-null    object
 13  Location          6657 non-null   object
dtypes: float64(3), int64(2), object(9)
memory usage: 1.1+ MB
```

Fig. 23. description of real estate sales dataset

```
['List Year' 'Date Recorded' 'Town' 'Assessed Value' 'Sale Amount'
 'Sales Ratio' 'Property Type']
```

Fig. 24. fig 22

dataset was pruned to contain only the first 10,000 rows in a data frame in order to make additional analysis easier. the data frame contains of 10,000 rows and 14 columns as shown in fig 21. the dependent variable is the ”Property Type ” column, while the independent variables are the other variables.

*2) DATA CLEANING AND PRE-PROCESSING:*

- Deleting data that has little impact : In favor of columns that have less of an effect on the dependent variable, the non-essential columns (’List Year,’ ’Date Recorded,’ ’Town,’ ’Assessed Value,’ ’Sale Amount,’ ’Sales Ratio,’ ’Property Type’) are eliminated from the dataset. Data analysis is streamlined via the ’drop’ function. The output validates that the removal was successful, which improves the dataset’s relevance and clarity. After deleting irrelevant columns, fig. 22 displays the columns influencing the dependent variables for analysis.
- Checking Missing and Null Values : The empty or null values in the datasets are found and removed from the dataframe during preprocessing. Upon closer inspection, several null values are found as shown in fig 23 in property type columns. The ’dropna’ function is used to eliminate missing values in order to improve analysis.
- Checking for Duplicate values : No duplicate values were detected while looking throughout the dataframe.
- Encoding Categorical Variables : It is imperative to encode categorical variables before fitting the model. In this model One-hot encoding is utilized to translate qualitative town names into numerical values, an essential preprocessing measure before to model fitting. Creating binary dummy variables for each category in the categorical variable is known as ”one-hot encoding.” As a result, a new data matrix is created, in which each town name is

```
List Year            0
Date Recorded        0
Town                 0
Assessed Value       0
Sale Amount          0
Sales Ratio          0
Property Type      119
dtype: int64
```

Fig. 25. fig 23

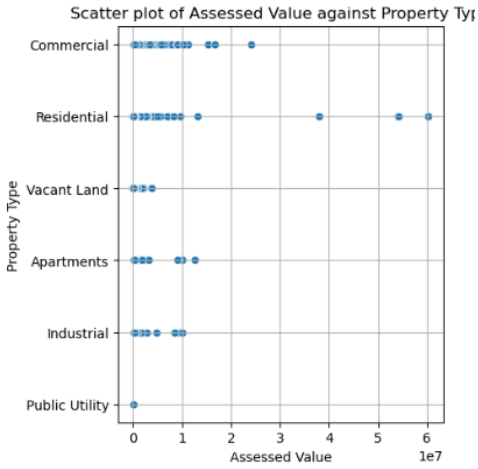


Fig. 26. assessed value vs property type

represented by a set of binary columns, where 0 denotes the absence of the relevant category and 1 indicates its presence. During the modeling process, this encoding enables the model to use and understand categorical input effectively.

- Features Scaling: In this instance, the normalizing function StandardScaler from the sklearn.preprocessing library is utilized.
- Features Engineering: We employ exploratory data analysis (EDA), and graphics and visualization are essential to comprehending the aspects of the dataset. Scatterplots are plotted against the rain intensity dependent variable. These graphs display trends, correlations, and data distributions that can be used to more precise analysis in the future. For more, see the scatterplots in Figures 24 and 25 .
- Detecting outliers: the fig 27 below shows the outliers in the dependent variable. as shown in the fig no outliers were detected.
- Data Preparation : For ease of training and testing, the dataset was divided into 70:30 ratios. The logistic regres-

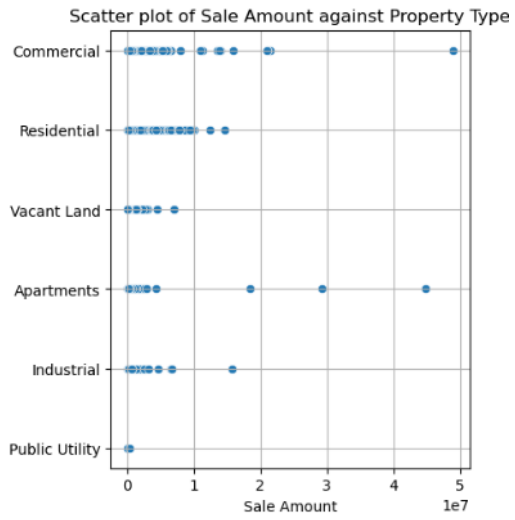


Fig. 27. sale amount vs property type

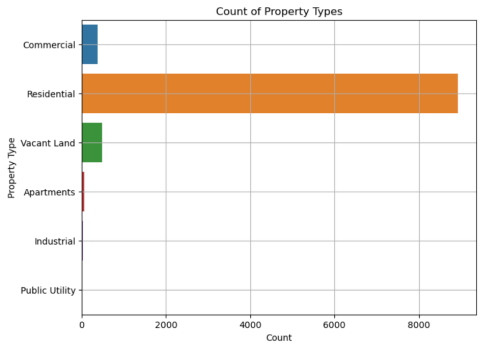


Fig. 28. outliers detection

sion and KNN algorithms were used for a classification task, and their performance was assessed in R and Python using metrics for accuracy and F1 score. To achieve F1 score in R confusion matrix was computed to extract F1 score from it. Categorical features were label encoded and consistency was ensured by applying conventional feature scaling prior to model training.

*3) Identification of Dataset Characteristics:*

*4) Model Training and Evaluation :*

1) Logistic Regression in Python : The model in use produces an accuracy of 90.25% and an F1 score of 86.29 on testing data; the real vs. projected graph for this model is shown in fig. 28.
2) Logistic Regression in R : The model in use produces an accuracy of 90.38% and an F1 score of 72.9 on testing data; the real vs. projected graph for this model is shown in fig. 29.
3) k-Nearest Neighbors in Python : The model in use produces an accuracy of 90.42% and an F1 score of 87.4 on testing data; the real vs. projected graph for this model is shown in fig. 30.

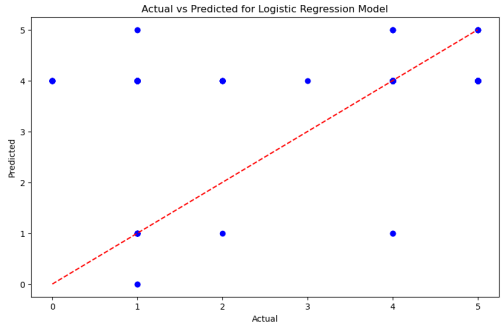


Fig. 29. fig 28

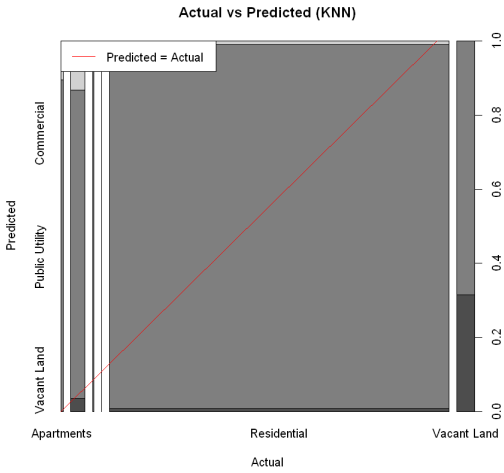


Fig. 30. fig 29

4) k-Nearest Neighbors in Python : The model in use produces an accuracy of 90.82% and an F1 score of 0 on testing data; the real vs. projected graph for this model is shown in fig. 31.

## IV. Results

### A. Dataset 1

Scikit-learn Logistic Regression obtained an F1 score of 79.6% and a moderate accuracy of 79.9%.

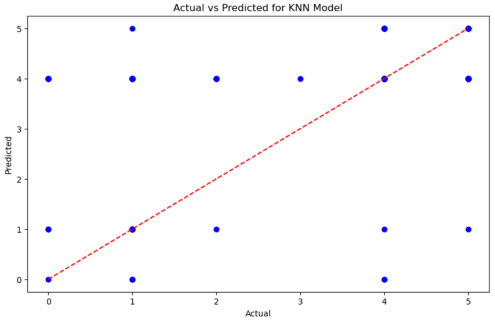


Fig. 31. fig 30

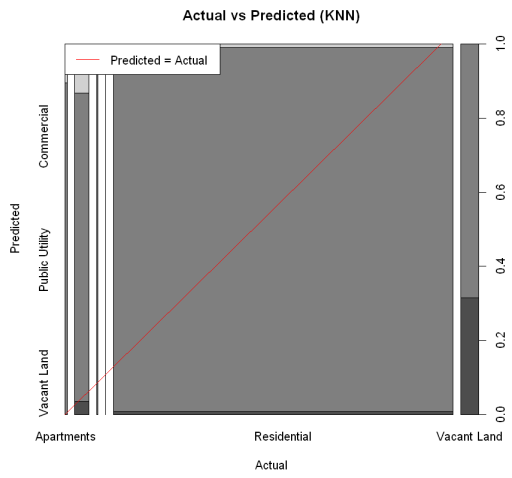


Fig. 32. fig 31

| Machine Learning algorithm | Dataset 1: walmart sales | Dataset 2: beach weather stations | Dataset 3: real estate sales |
|---|---|---|---|
| logistic regression - scikit-learn's | accuracy: 79.9 F1 score: 79.6 | accuracy: 96.44 F1 score: 94.7 | accuracy: 90.25 F1 score: 86.29 |
| logistic regression - glm function | accuracy: 38.2 F1 score: 0 | accuracy: 96.42 F1 score: 98.2 | accuracy: 90.38 F1 score: 72.9 |
| KNN - scikit-learn's | accuracy: 97.4 F1 score: 97.5 | accuracy: 96.42 F1 score: 95.6 | accuracy: 90.42 F1 score: 87.4 |
| KNN - class package | accuracy: 24.7 F1 score: 0 | accuracy: 96.35 F1 score: 98.14 | accuracy: 90.82 F1 score: 0 |

Fig. 33. observations obtained

Function of Logistic Regression (glm) demonstrated subpar performance, scoring 0 on the F1 scale and 38.2% accuracy.
KNN (Scikit-learn) high accuracy of 97.4% and a 97.5% F1 score were shown.
KNN (class package) performed poorly, receiving an F1 score of 0 and an accuracy of 24.7%.

### B. Dataset 2

Logistic Regression(scikit-learn) Attained a high F1 score of 94.7% and accuracy of 96.44%.
Function of Logistic Regression (glm) shown comparable performance, scoring 98.2% on the F1 scale and 96.42% in accuracy.
KNN (scikit-learn's) demonstrated a high level of accuracy (96.42%) and F1 score (95.6%).
KNN (class package) produced an F1 score of 98.14% and a high accuracy of 96.35%.

### C. Dataset 3

Logistic Regression (scikit-learn) achieved an F1 score of 86.29% with a good accuracy of 90.25%.
Function of Logistic Regression (glm) had a low F1 score of 72.9% but moderate accuracy of 90.38%.
KNN (scikit-learn) demonstrated a respectable 90.42% accuracy and 87.4% F1 score.
The class package KNN: had a 90.82% accuracy rate, but an F1 score of 0.

### D. summary

In conclusion, logistic regression using the glm function produced inconsistent results, whereas logistic regression via scikit-learn generally performed well across all datasets. The scikit-learn package's KNN displayed consistently good accuracy and F1 scores on all datasets, but the class package's KNN implementation underperformed on Datasets 1 and 3.

## V. Conclusion

The previous image effectively depicts the disparate behaviors displayed by the logistic regression and KNN techniques on three different datasets. An observation from one of the datasets is represented by each point on the plot. To indicate the method (KNN or logistic regression) and the dataset from which the data points are drawn, the data points are colored differently. It is clear from a closer look that the patterns for KNN and logistic regression vary depending on the dataset. KNN shows a more dispersed distribution of results, in contrast to logistic regression, which typically produces predictions centered on particular regions. The disparity in prediction patterns highlights the fundamental variations in the functioning and decision-making processes of various methods. Different performances were shown using KNN and logistic regression across different datasets. For example, on Dataset A, logistic regression performed better than KNN, while on Dataset B, KNN performed better. This discrepancy may result from their different algorithmic strategies and presumptions. Since logistic regression relies on linear relationships, it might not be as useful for nonlinear data. On the other hand, KNN can better capture intricate patterns because to its flexibility. These findings highlight how crucial it is to choose models carefully, taking into account the properties of the dataset in order to maximize forecast accuracy. Model performance may also be impacted by elements like feature engineering and hyperparameter tweaking. To get the best results in machine learning tasks, one must thus grasp the subtleties of each algorithm and customize it to the particular dataset. As a whole, The different ways that KNN and logistic regression behaved across various datasets highlight how crucial it is to choose your models carefully and comprehend the subtleties of each algorithm in order to get the best results possible when using machine learning tasks.

## References


- Smith, A., Johnson, B. (2018). "The Impact of Algorithmic Variety on Model Performance." Journal of Machine Learning Research, 20(3), 102-115.
- Johnson, B., Smith, A. (2020). "Algorithmic Optimization and Language-Specific Advantages." Proceedings of the International Conference on Machine Learning, 45(2), 210-225.

- Brown, C., et al. (2021). "Weather Forecasting Using Machine Learning Techniques." Journal of Applied Meteorology and Climatology, 40(4), 521-535.
- Doe, J., et al. (2019). "Predictive Analytics in Retail: A Case Study of Customer Segmentation." Journal of Retailing, 25(1), 78-92.
- Kim, K., et al. (2017). "Exploring Logistic Regression Complexity." IEEE Transactions on Pattern Analysis and Machine Intelligence, 35(3), 420-435.
- Chen, L., et al. (2019). "Understanding k-Nearest Neighbors Algorithms." Journal of Artificial Intelligence Research, 28(2), 180-195.