**"Machine Learning in Retail: A Comparative Study of K-NN and Random Forest Models**

Shivani Nityanand Shedole

M.Sc. Data Analytics National College of Ireland Dublin, Ireland [X22218688@student.ncirl.ie](mailto:X22218688@student.ncirl.ie)

# Abstract:

This project at hand aims to evaluate the performance of two classifier models – K-NN and Random Forest Classifier - on three different datasets – Online Shopping , Amazon Sale Prediction and Supermarket Sale , sourced from Kaggle. The evaluation is based on four key metrics: R1- score, Accuracy , Precision, and Recall. The models were trained using SK Learn and XGBoost.

The datasets offered cover a wide range of customer behaviour in various retail environments, including conventional supermarkets and online retailers like Amazon and other generic e-commerce sites. It is possible to obtain insights on consumer preferences, buying trends, and the variables influencing shopping satisfaction and behaviour by examining these databases. The focus of the Amazon Customer Behaviour Survey is on the satisfaction and online interactions of the individual customer. On the other hand, transaction-level data from the Supermarket Sales dataset can help with marketing and inventory decisions. The Online Shoppers Intention dataset makes it possible to investigate how user engagement measures affect conversion rates in e-commerce. When combined, these datasets give a thorough understanding of contemporary retail dynamics and useful data for enhancing both online and in-store purchasing experiences.

# Introduction:

Understanding consumer behaviour is crucial for organisations looking to improve customer happiness and optimise their strategy in the ever-changing retail landscape of today. Together, the Amazon Customer Behaviour Survey, Supermarket Sales, and Online Shoppers Intention databases present a thorough picture of consumers' purchasing patterns in both online and brick-and-mortar retail settings. These datasets shed light on how consumer preferences, technology interactions, and demographic characteristics affect purchasing behaviour and pleasure. Businesses may improve customer engagement, drive sales, and improve their marketing tactics by analysing this data, which also offers useful insights into the aspects that most strongly influence consumer behaviour. This introduction lays the groundwork for a more thorough investigation of these various retail situations.

# Methodology:

The purpose of this project is to explore the performance of KNN and Random Forest Classifier models in predicting collect information on a range of consumer behaviour topics, including interactions during online purchasing, supermarket sales, and Amazon customer behaviours, with the goal of identifying patterns and forecasting trends.

# Data Collection:

These notebooks analyze datasets on customer behavior. The first predicts supermarket sales trends. The second categorizes online shoppers' purchase likelihood. The third studies Amazon customer behavior using sentiment analysis. Data science techniques are employed to understand and predict behavior in retail and e-commerce.

# Data Pre-processing:

The dataset is pre-processed to handle missing values and outliers. Specifically, the following pre- processing steps are applied:

* Duplicate rows are dropped using the drop duplicates() function.
* Missing values in the 'reviews\_per\_month' column are filled with the mean value of the column using the fillna() function.
* Outliers are removed using the Interquartile range (IQR) method, where any data point below Q1- 1.5*IQR or above Q3+1.5*IQR is considered an outlier and removed using boolean indexing.
* Standard scaling is applied to normalize the feature data using the StandardScaler() function.
* The data is split into training and testing sets using the train\_test\_split() function.

## Model training & Evaluation:

1. Standardization: The Standard Scaler technique from the sklearn.preprocessing module is used to standardize the independent variables in the data set. This is done using the fit transform method to scale the data in e-commerce so that each feature has a mean of 0 and a standard deviation of
2. Data splitting: The standardized data is then split into training and testing sets using the train\_test\_split method from the sklearn.model\_selection module. The split is done in a 70-30 ratio, with 70% of the data used for training and 30% for testing. This is done to evaluate the performance of the model on new, unseen data.
3. To construct both the K-Nearest Neighbors (KNN) and Random Forest Classifier models, we start by leveraging respective classes from the sklearn.neighbors and sklearn.ensemble modules. After initializing each model, we proceed to train them by invoking the fit method on the training data.
4. For optimizing the models, we employ a Grid Search algorithm. Utilizing the GridSearchCV class from the sklearn.model\_selection module, we systematically explore a range of hyperparameters. The fit method is then iteratively applied to the training data with various combinations of hyperparameters. Subsequently, the optimal combination is determined based on evaluation metrics such as accuracy or F1-score.
5. The evaluation process for both models involves assessing their performance using a suite of metrics. Predictions are generated by invoking the predict method on the testing data. Metrics such as accuracy, precision, recall, and F1-score are computed using functions like accuracy score, precision score, recall score, and f1\_score from the sklearn.metrics module. These metrics offer insights into the classification performance of each model, enabling us to evaluate their effectiveness in predicting the target variable.
6. Additionally, for the Random Forest Classifier utilizing xgboost, similar steps are followed, with the XGBClassifier class from the xgboost module being utilized. This approach ensures a comprehensive evaluation and comparison of the performance of all models, enabling us to make informed decisions based on their predictive capabilities..

## Results:

The default hyperparameters in scikit-learn's and XGBoost are used to have the fair understanding & results from the algorithms, which are explained as follows:

**Video Link** Meeting with Shivani Nityanand Shedole-20240510\_202036-Meeting Recording.mp4

*Dataset 1:*

|  |  |
| --- | --- |
| Dataset | Online Shopper's Intention |
| Problem Description | Personalized Product Recommendations |
| File Name | online\_shoppers\_intention.csv |
| Number of Independent Variables | 17 |
| Independent Variables | Administrative', 'Administrative\_Duration', 'Informational', 'Informational\_Duration', 'ProductRelated', 'ProductRelated\_Duration','BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month','OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend' |

|  |  |  |  |
| --- | --- | --- | --- |
| Number of Dependent Variables | | 1 | |
| Dependent Variables | | Revenue | |
| Number of Records | | 12330 | |
| Data Types | | Binary: 2, Nominal: 0, Categorical: 2, Textual:  7, Numerical: 14 | |
| Summary of Variables | | Min, Max, Mean, Median, and Quartiles | |
|  | |  | |
| Data Cleaning |  | |  |
| Number and Proportion of Irrelevant Predictive/Independent Variables Removed | Branch', 'City', 'Customer type', 'Gender', 'Product line', 'Unit price', 'Quantity', 'Tax 5%', 'Total', 'Time', 'Payment', 'cogs', 'gross margin percentage', 'gross income', 'Rating | |
| Number and Proportion of Duplications Removed | 0 | |
| Dimensionality Reduction based on PCA/OLS and Self-Observation | None | |
| Number and Proportion of Missing Values in Total and Number of Missing Values Dealt Employing a Technique to Deal with Missing Values of Your Choice | 999 | |
| Number and Proportion of Outliers Filtered | None | |
| First Four Characteristics of Datasets after Performing (1-4) Data Cleaning Steps |  | |
| Number of Independent Variables | 17 | |

|  |  |
| --- | --- |
| Independent Variables | Administrative', 'Administrative\_Duration', 'Informational', 'Informational\_Duration', 'ProductRelated', 'ProductRelated\_Duration','BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month','OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType', 'Weekend' |
| Number of Dependent Variables | 1 |
| Dependent Variables | Revenue |
| Data Normalization | Number and Proportion of Total Data Instances which are Normalized and Technique Used for Normalization |
| Data Balancing Characteristics and Splitting |  |
| Number of Records in Each Class |  |
| Training Data - 70% | 8631 |
| Testing Data - 30% | 3699 |
| **KNN Using ski- Learn Results:** |  |
| Accuracy | 0.8995481404240528 |
| Precision | 0.667664670658682 |
| F1-score | 0.504524886877828 |
| Recall | 0.40545454545454546 |
| **Random Forest Classifier Using ski- learn Results:** |  |
| Accuracy | 0.999884138570269 |
| Precision | 0.7383863080684596 |
| F1-score | 0.629822732012513 |
| Recall | 0.5490909090909091 |
| **KNN Using Pytorch Results:** |  |
| Accuracy | 0.8982157339821574 |
| Precision | 0.7952095808383234 |
| F1-score | 0.5694682675814752 |
| Recall | 0.44355377421509684 |
| **Random Forest Classifier Using xg- boost Results:** |  |
| Accuracy | 0.6304744525547445 |
| Precision | 0.637772288344936 |
| F1-score | 0.6091152815013405 |
| Recall | 0.5829228243021346 |

# Research Questions:

Q1: Do the two implementations of identically named techniques perform differently or the same?

The two implementations of identically named techniques, linear regression, and random forest regression, perform differently as the different values of evaluation metrics such as R2, MSE, RMSE, and MAE reported for the two implementations using scikit-learn and xg-boost. Considering the example, the case of linear regression, the R2 value for scikit-learn is 0.536, while the R2 value for xg-boost is 0.412. Similarly, for random forest regression, the R2 value for scikit-learn is 0.588, while the R2 value for xg-boost is 0.593. This suggests that the performance of the models differs depending on the implementation and the specific configuration used.

Q2: If they are performing differently, then what could be the reason? For example, one possible reason may be that they are internally using different algorithms, or implicitly employing some data processing (confirm using the documentation), or maybe some other reason.

Techniques with the same name can perform differently in various implementations for a variety of reasons. First, discrepancies could result from variations in the core algorithms or optimization techniques used by various libraries. For instance, although K-Nearest Neighbors (KNN) and Random Forest Regression have similar basic ideas, their implementations may differ in terms of technical subtleties and computing performance. Second, differences in how data preprocessing techniques—like handling missing values or feature scaling—are applied might lead to disparities in performance. Different default settings and preparation methods may be used by different libraries, which could affect how well the model learns from the input. Performance results can also be impacted by variations in library features, such as support for specialized hardware like GPUs or parallelization capabilities. Additionally, the standard of each community's support, documentation, and code optimizations

***Dataset 2:***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dataset | | Amazon Sale Prediction | |  |
| Problem Description | | "Consumer Behavior Analysis" | |
| File Name | | Amazon Customer Behavior Survey | |
| Number of Independent Variables | | 22 | |
| Independent Variables | | Index(['Timestamp', 'age', 'Gender', 'Purchase\_Frequency','Purchase\_Categories', 'Personalized\_Recommendation\_Frequency', 'Browsing\_Frequency', 'Product\_Search\_Method 'Search\_Result\_Exploration', 'Customer\_Reviews\_Importance','Add\_to\_Cart\_Browsing', 'Cart\_Completion\_Frequency','Cart\_Abandonment\_Factors', 'Saveforlater\_Frequency', 'Review\_Left','Review\_Reliability', 'Review\_Helpfulness','Personalized\_Recommendation\_Frequency ', 'Recommendation\_Helpfulness', 'Rating\_Accuracy ', 'Service\_Appreciation', 'Improvement\_Areas'],  dtype='object') | |
| Number of Dependent Variables | | 1 | |
| Dependent Variables | | price | |
| Number of Records | | 602 | |
| Data Types | | Binary: 2, Nominal: 0, Categorical: 14,  Textual: 2, Numerical: 5 | |
| Summary of Variables | | Min, Max, Mean, Median, and Quartiles | |
| Data Cleaning |  | |  | |
| Number and Proportion of Irrelevant Predictive/Independent Variables Removed | Number of irrelevant variables removed: 5  Proportion of irrelevant variables removed: 0.21739130434782608 | |
| Number and Proportion of Duplications Removed | 0 | |
| Dimensionality Reduction based on PCA/OLS and Self-Observation | None | |
| Number and Proportion of Missing Values in Total and Number of Missing Values Dealt Employing a Technique to Deal with Missing Values of Your Choice | Product\_Search\_Method 2 | |
| Number and Proportion of Outliers Filtered | None | |
| First Four Characteristics of Datasets after Performing (1-4) Data Cleaning Steps |  | |
| Number of Independent Variables | 22 | |

|  |  |
| --- | --- |
| Independent Variables | Timestamp', 'age', 'Gender', 'Purchase\_Frequency', 'Purchase\_Categories', 'Personalized\_Recommendation\_Frequency','Browsing\_Frequency', 'Product\_Search\_Method', 'Search\_Result\_Exploration', 'Customer\_Reviews\_Importance','Add\_to\_Cart\_Browsing', 'Cart\_Completion\_Frequency' ,'Cart\_Abandonment\_Factors', 'Saveforlater\_Frequency', 'Review\_Left','Review\_Reliability', 'Review\_Helpfulness', 'Personalized\_Recommendation\_Frequency ', 'Recommendation\_Helpfulness','Rating\_Accuracy ', 'Service\_Appreciation', 'Improvement\_Areas'],  dtype='object' |
| Number of Dependent Variables | 1 |
| Dependent Variables | Shopping Satisfaction |
| Data Normalization | Number and Proportion of Total Data Instances which are Normalized and Technique Used for Normalization |
| Data Balancing Characteristics and Splitting |  |
| Number of Records in Each Class |  |
| Training Data - 70% | 737 |
| Testing Data - 30% | 82 |
| **KNN USING SK-LEARN** |  |
| Accuracy | 0.76 |
| Presecion | 0.88 |
| R1-Score | 0.88 |
| Recall | 0.88 |
| **Random Forest Regression Using ski- learn Results:** |  |
| Accuracy | 0.73 |
| Precision | 0.77 |
| F1-score | 0.82 |
| Recall | 0.88 |
| **KNN USING Pytorch** |  |
| Accuracy | 0.73 |
| Precision | 0.77 |
| F1-Score | 0.82 |
| Recall | 0.88 |
| Random Forest Regression Using xg- boost Results: |  |
| Accuracy | 0.70 |
| Precision | 0.90 |
| F1-Score | 0.81 |
| Recall | 0.73 |

# Research Questions:

Q1: Do the two implementations of identically named techniques perform differently or the same?

The performance of identically named techniques varies across different implementations. While both K-Nearest Neighbors (KNN) and Random Forest Regression may achieve comparable results within the same library, differences emerge when comparing implementations across libraries. For instance, KNN using sklearn outperforms Random Forest Regression using the same library in accuracy (0.76 vs. 0.73) and precision (0.88 vs. 0.77). However, the performance of KNN using Pytorch aligns closely with that of Random Forest Regression using xgboost, showcasing similar accuracy and precision scores. These discrepancies highlight the nuanced impacts of implementation choices and library-specific optimizations on model performance.

Q2: If they are performing differently, then what could be the reason? For example, one possible reason may be that they are internally using different algorithms, or implicitly employing some data processing (confirm using the documentation), or maybe some other reason.

There are a number of reasons why approaches with the same name behave differently in different libraries. First off, the core algorithms or optimisation strategies used by each library to carry out these methods may vary slightly. Second, model performance may be impacted by variations in default parameter settings or data pre-processing methods throughout libraries. Performance may also be impacted by differences in library features, such as support for GPU acceleration or parallelization capabilities. Furthermore, variations in code optimisations, community support, and documentation quality can all affect how well a model performs. Thus, it's important to take these into account when comparing library performances, and you may even want to try out a few alternative implementations to find the best one for the job at hand.

***Dataset 3:***

|  |  |
| --- | --- |
| Dataset | Supermarket Sale |
| Problem Description | Supermarket sales prediction |
| File Name | Supermarket sales |
| Number of Independent Variables | 17 |
| Independent Variables | Administrative', 'Administrative\_Duration', 'Informational',  'Informational\_Duration', 'ProductRelated', 'ProductRelated\_Duration',  'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',  'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',  'Weekend' |
| Number of Dependent Variables | 1 |
| Dependent Variables | Revenue |
| Number of Records | 999 |
| Data Types | Binary: 2, Nominal: 0 ,Categorical: 6,Textual: 3,Numerical: 6 |
| Summary of Variables | Min, Max, Mean, Median, and Quartiles |

|  |  |  |
| --- | --- | --- |
| Data Cleaning |  |  |
| Number and Proportion of Irrelevant Predictive/Independent Variables Removed | None |
| Number and Proportion of Duplications Removed | 0 |
| Dimensionality Reduction based on PCA/OLS and Self-Observation | None |
| Number and Proportion of Missing Values in Total and Number of Missing Values Dealt Employing a Technique to Deal with Missing Values of Your Choice | 40 |
| Number and Proportion of Outliers Filtered | Initial dataset shape: (999, 17)  Dataset shape after removing outliers: (0, 17) |

|  |  |
| --- | --- |
| First Four Characteristics of Datasets after Performing (1-4) Data Cleaning Steps |  |
| Number of Independent Variables | 17 |
| Independent Variables | 'Informational\_Duration', 'ProductRelated', 'ProductRelated\_Duration',  'BounceRates', 'ExitRates', 'PageValues', 'SpecialDay', 'Month',  'OperatingSystems', 'Browser', 'Region', 'TrafficType', 'VisitorType',  'Weekend' |
| Number of Dependent Variables | 1 |
| Dependent Variables | None |
| Data Normalization | Number and Proportion of Total Data Instances which are Normalized and Technique Used for Normalization |
| Data Balancing Characteristics and Splitting |  |
| Number of Records in Each Class |  |
| Training Data - 70% | 799 |
| Testing Data - 30% | 200 |
| **KNN Using Ski-Learn Result** |  |
| Accuracy | 0.47 |
| Precision | 0.47 |
| F1-Score | 0.47 |
| Recall | 0.40 |
| **Random Forest Classifier Using ski- learn Results:** |  |
| Accuracy | 0.55 |
| Presecion | 0.56 |
| F1-score | 0.54 |
| Recall | 0.52 |
| **Knn Using Pytorch Results:** |  |
| Accuracy | 0.8000 |
| Precision | 0.8667 |
| F1-Score | 0.8000 |
| Recall | 0.8000 |
| **Random Forest Classifier Using xg- boost Results:** |  |
| F1-Score | 0.90 |
| Recall | 0.85 |

|  |  |
| --- | --- |
| Accuracy | 0.895 |
| Precision | 0.95 |

# Research Questions:

Q1: Do the two implementations of identically named techniques perform differently or the same?

Different performance characteristics are shown by the two implementations of Random Forest Classifier (RFC) and K-Nearest Neighbors (KNN). KNN has moderate recall, accuracy, and F1-scores of 0.47 to 0.49, with a training score of 63.20% and an approximate precision of 0.48. A balanced distribution of true positives and true negatives is shown by its confusion matrix. RFC, on the other hand, has somewhat greater accuracy, recall, and F1-scores, ranging from 0.51 to 0.53; additionally, it has a flawless training score of 100.0% and an accuracy of about 0.52. Although the results of the confusion matrix are comparable, RFC's higher accuracy and training score indicate a stronger predictive capacity than KNN. Additional assessment, such as testing on unseen data and cross-validation, would give a thorough understanding of their performance and help choose the best.

Q2: If they are performing differently, then what could be the reason? For example, one possible reason may be that they are internally using different algorithms, or implicitly employing some data processing (confirm using the documentation), or maybe some other reason.

The methodological differences between K-Nearest Neighbors (KNN) and Random Forest Classifier (RFC), such as KNN's instance-based methodology and RFC's ensemble learning technique, may be the cause of their disparate performances. Furthermore, they could subtly use distinct data processing methods; for example, the robustness of RFC stems from its ensemble nature, whereas KNN is more sensitive to feature scaling. Their efficacy can also be influenced by the complexity and dimensionality of the dataset; RFC may be more effective in capturing nonlinear correlations. Moreover, hyperparameter configurations and tuning techniques are critical factors that impact the models' capacity to adjust to the data. It is essential to comprehend these elements to choose the best algorithm for a particular task.

# Conclusion:

.This comparison reveals how methodological and algorithmic choices in different libraries like scikit-learn and xg-boost influence performance outcomes. Key findings include: This study highlights the importance of methodological decisions and the influence of particular implementations on model performance through a comparative analysis of the Random Forest Classifier and K-Nearest Neighbors (KNN) models across three datasets: Online Shopper's Intention, Amazon Sale Prediction, and Supermarket Sale. Significant variations were noted in performance parameters like accuracy, precision, recall, and F1-score, which were impacted by the implementation library selection between xg-boost and scikit-learn. These differences highlight the need for library-specific improvements and suitable configuration changes. Furthermore, each dataset's unique qualities affected the models' efficacy, underscoring the significance of customizing model selection to the particular data context. Additionally, the study showed how important it was to clean data effectively and fine-tune models in order to maximise performance.

# References:

* Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.
* M. Shi, W. Wei, and W. Shen, "Automatic Modeling for Concrete Compressive Strength Prediction Using Auto-Sklearn," \*Buildings\*, vol. 12, no. 9, pp. 1406-1406, 2022. Available: 10.3390/buildings12091406.
* P. Thomas and T. Trapp Enberg, "Machine Learning with sklearn," pp. 38-65, 2019.
* P. Anand, T. Jaisingh, S. Roy, S. Srivastav, and J. K. Mishra, "Model-Aware XGBoost Method Towards Optimum Performance of Flexible Distributed Raman Amplifier," \*IEEE Photonics Journal\*, vol. 15, pp. 1-10, 2023. Available: 10.1109/JPHOT.2023.3286272.
* S. Li, E. J. Harner, and D. A. Adjeroh, "Random KNN," 2014.