IS6052: PREDICTIVE ANALYTICS

Predicting Homebuyer Decisions

Sajin Siyad
Sajinsiyad77@gmail.com

CORK UNIVERSITY BUSINESS SCHOOL | UCC University College Cork, Ireland Coláiste na hOllscoile Corcaigh

1. Introduction

The purpose of this report is to analyse the factors influencing the decisions of Irish homebuyers in Dublin when purchasing a house and to find the most accurate predictive analytics model to determine the likelihood of a property being sold. The analysis includes data cleaning, feature engineering, exploratory data analysis (EDA), and predictive modelling using various techniques. The Irish housing market, especially in Dublin, faces challenges like Brexit impacts, fluctuating mortgage regulations, and varying demand influenced by the city's growing tech industry and job opportunities. These factors make predicting housing decisions both complex and vital for developers and policymakers.

2. Dataset Description and Initial Exploration

2.1 Dataset Overview:

The dataset includes property attributes such as:

- Location: Categorical data indicating the local authority (DCC, Fingal, Dun Laoghaire, South Dublin).
- Property scope: Categorical data that gives us the type of property.
- Availability: Attribute specifying when the buyer can move in.
- Size: Specifying number of bedrooms.
- Total square feet: Numerical data showing total square feet of each.
- price per square feet: Numerical data showing price per square feet.
- Buying or not buying: Categorical data indicating the buying decision of the homebuyers.

These key attributes along with other factors are included in the dataset, that could influence homebuying decisions. The dataset has a total of 13320 rows and 12 columns.

```
print('Number of rows and columns: ', df_house_price.shape)
```

```
Number of rows and columns:  (13320, 12)
```

Figure 1: Number of rows and columns

2.2 Exploratory Data Analysis

To gain an initial understanding of the dataset, key descriptive functions such as info(), shape(), and describe() were utilised. These provided insights into the structure of the data, including data types, the number of entries, missing values, and summary statistics (e.g., mean, median, and standard deviation) for numerical variables. To further explore the distribution of key variables, frequency distributions were plotted, enabling a visual understanding of patterns, trends, and potential anomalies in the data. This step provided a clearer view of the variable distributions, aiding in identifying skewness, outliers, and overall data characteristics.

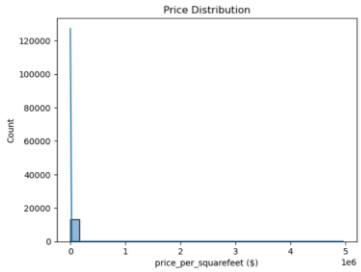

Figure 2: Price Distribution

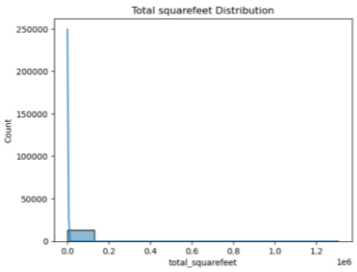

Figure 3: Total Square feet Distribution

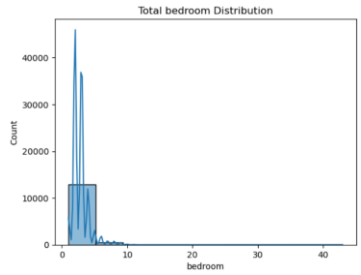

Figure 4: Bedroom Distribution

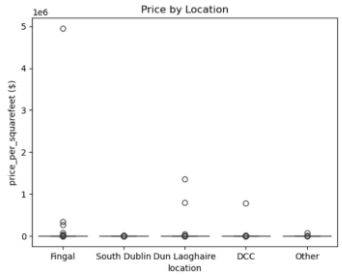

Figure 5: Box plot for Price by location

From the Exploratory Data Analysis (EDA), it was evident that outliers were present in key variables. However, most of the data appeared to be concentrated around non-extreme values. Despite this, the presence of outliers could significantly impact the cleaning process and the accuracy of the predictive models. To address this, boxplots were generated for the variables of interest to visualise the extent and distribution of outliers. For handling outliers, median imputation was identified as a suitable approach, particularly for variables with extreme outliers, as the median is less sensitive to extreme values compared to the mean. For example, when plotting boxplots of price by location, it was observed that in Fingal, there was an extreme outlier for price that could heavily distort the mean. Summary statistics obtained

using describe() were also analysed before and after handling the outliers, revealing the impact of outliers on the dataset. This process ensured the data was cleaned effectively without losing its core characteristics.

```
df_house_price.describe()
```

	bedroom	total_squarefeet	bath	balcony	price_per_squarefeet ($)
count	13315.000000	1.331500e+04	13315.000000	13315.000000	1.331500e+04
mean	2.803605	1.822601e+03	2.688321	1.603455	1.415381e+03
std	1.293012	1.428388e+04	1.337417	0.803059	4.569528e+04
min	1.000000	1.000000e+00	1.000000	0.000000	3.040000e+01
25%	2.000000	1.100000e+03	2.000000	1.000000	4.862700e+02
50%	3.000000	1.277000e+03	2.000000	2.000000	6.194200e+02
75%	3.000000	1.680000e+03	3.000000	2.000000	8.326300e+02
max	43.000000	1.306800e+06	40.000000	3.000000	4.953333e+06

Figure 6: Summary statistics with outliers

```
df_house_price_cleaned.describe()
```

	bedroom	total_squarefeet	bath	balcony	price_per_squarefeet ($)	total_price ($)	price_per_bedroom ($)
count	10737.000000	10737.000000	10737.000000	10737.000000	10737.000000	1.073700e+04	1.073700e+04
mean	2.418366	1304.111959	2.273726	1.567756	608.510279	8.108031e+05	3.361141e+05
std	0.653768	375.794334	0.643704	0.771546	198.937036	4.041129e+05	1.399420e+05
min	1.000000	276.000000	1.000000	0.000000	56.750000	9.080048e+04	3.193200e+04
25%	2.000000	1082.000000	2.000000	1.000000	465.100000	5.220988e+05	2.383520e+05
50%	2.000000	1239.000000	2.000000	2.000000	572.130000	7.082375e+05	3.121240e+05
75%	3.000000	1513.000000	3.000000	2.000000	711.850000	9.988026e+05	4.085988e+05
max	4.000000	2550.000000	4.000000	3.000000	1351.190000	2.553757e+06	2.270000e+06

Figure 7: Summary statistics without outliers

Also, correlations between the different variables in the dataset where checked. The obstained correlations have been put into correlation matrix for easy analysis. There were some good correlations observed between some of the variables in the dataset.

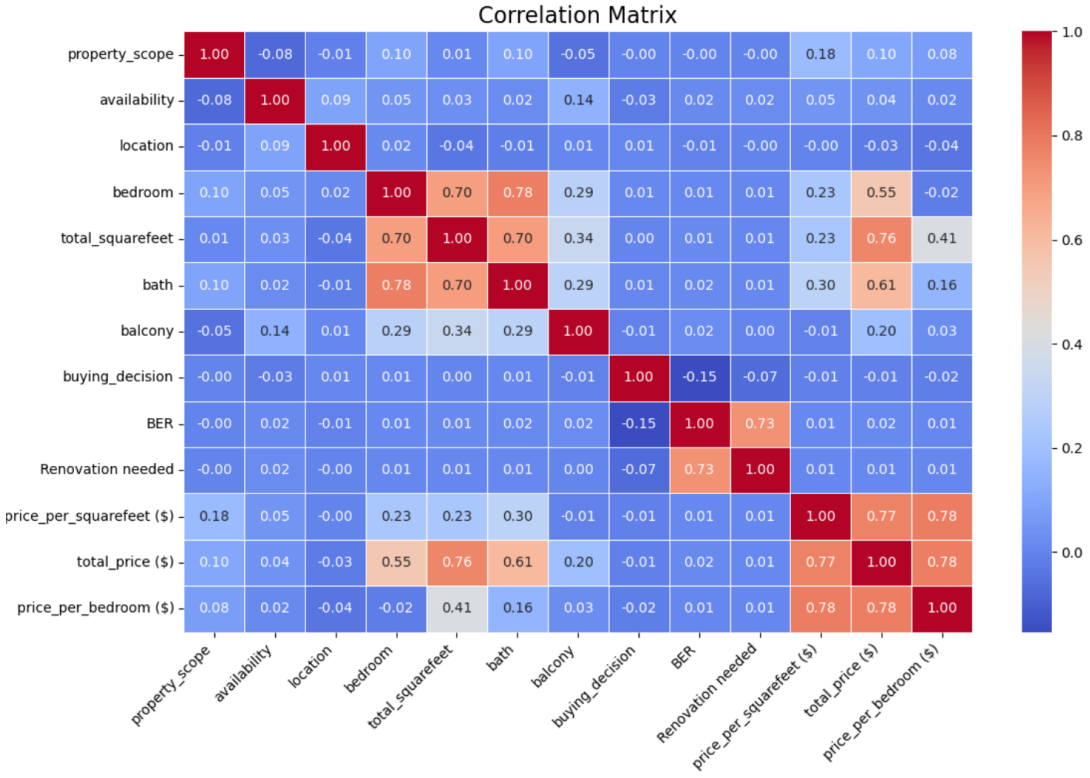

Figure 8: Correlation Matrix

The exploratory data analysis revealed critical insights into the dataset, including the presence of outliers, missing values, and skewed distributions in key variables. Boxplots highlighted extreme values that could impact modelling, while the correlation matrix identified strong relationships between variables like bedroom, total_squarefeet, and total_price. These findings provide a clear direction for the next step: data cleaning, which will address these issues to ensure the dataset is accurate, consistent, and ready for predictive modelling.

3. Data Preparation and Feature Engineering

3.1 Data Cleaning

For the data cleaning phase, the column names have been changed for the columns price-per-sqft-$, total_sqft and buying or not buying to price_per_squarefeet ($), total_squarefeet and buying_decision to ensure consistency and better readability. Used the duplicate() of pandas to identify duplicate rows in the dataset provided and no duplicates were identified.

```
#checking for duplicates
print("Number of duplicate rows in the dataset: ", df_house_price.duplicated().sum())
```

```
Number of duplicate rows in the dataset:  0
```

Figure 9: Number of duplicates in the dataset

Several inconsistencies were identified in the total_squarefeet column, including values represented in different units such as square yards, acres, and even ranges of values. To address these issues, appropriate code was written to standardize all values by converting them into a single consistent unit: square feet. All the dates within the availability column have been converted to 'Future Availability' to make the column to have categorical data.

For the purpose of identifying presence of missing values in the dataset, isna().sum() has been used and this revealed that there were missing values in the columns, bedroom, bath, balcony and price_per_squarefeet ($).

```
# Display columns with null values
df_house_price.isna().sum()
```

```
ID                          0
property_scope              0
availability                0
location                    0
bedroom                    16
total_squarefeet            0
bath                       73
balcony                   608
buying_decision             0
BER                         0
Renovation needed           0
price_per_squarefeet ($)  242
dtype: int64
```

Figure 10: Checking for null values

For the columns, bedroom, bath and balcony the missing values were filled using the median. For price_per_squarefeet ($), the dataset was grouped by property_scope, location, and bedroom, and missing values were filled using the median of each combination. Remaining missing values were addressed by grouping over property_scope and location, then filling with the median of these groups. As a final step to handle edge cases, the overall median of price_per_squarefeet ($) was used for any remaining missing values. This made sure that all the missing values got imputed.

```
ID                          0
property_scope              0
availability                0
location                    0
bedroom                     0
total_squarefeet            0
bath                        0
balcony                     0
buying or not buying        0
BER                         0
Renovation needed           0
Price_per_squarefeet ($)    0
dtype: int64
```

Figure 11: Checking for null values

Outliers were detected using box plots, which revealed the presence of extreme values in several numerical columns. To address these outliers, the interquartile range (IQR) method was applied (Aggarwal, C. C., 2017). For each column, the lower bound was calculated as Q1 - 1.5 * IQR, and the upper bound as Q3 + 1.5 * IQR. Values outside these bounds were identified as outliers and removed from the dataset. This approach ensures the dataset retains meaningful variability while reducing the influence of extreme values that could skew predictive models. Outliers were dropped because they likely represent erroneous or highly unusual data points that do not align with typical patterns, which could negatively impact the accuracy and generalizability of predictive models.

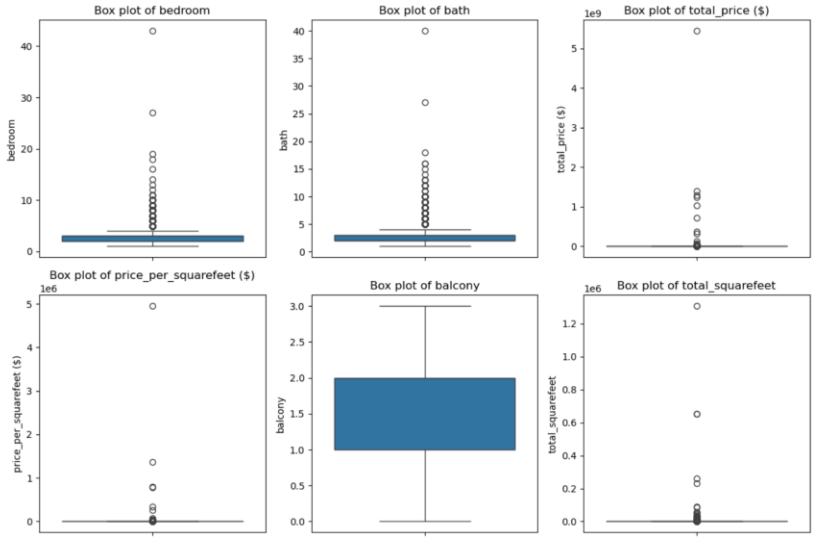

Figure 12: Boxplots to find outliers

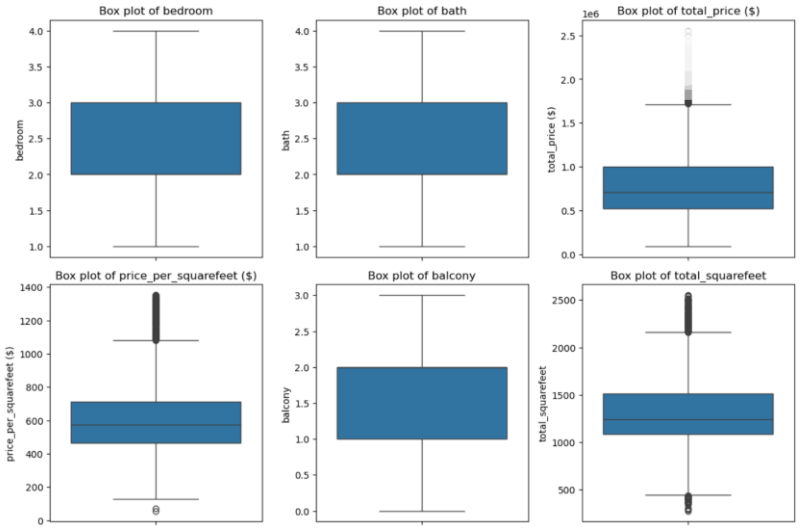

Figure 13: Boxplots after handling outliers

3.2 Feature Engineering

For the dataset, few news features were created (Zheng, A., & Casari, A., 2018):

- Total Price ($): Derived by multiplying the price_per_squarefeet ($) by the total square feet of the property. This feature provides an absolute cost metric that is essential for evaluating overall property affordability and desirability.
- Price per Bedroom: Total price divided by the number of bedrooms. This captures the affordability of the property per room, providing insights into pricing trends relative to property size.

4. Predictive Analysis

In this project, predictive analysis was conducted across two scenarios:

(i) Before handling outliers.

(ii) After handling outliers.

Various machine learning models, including Logistic Regression, Decision Tree, Random Forest, SVM, and KNN (Hastie T., Tibshirani R., & Friedman J., 2009), were employed. Logistic Regression serves as a reliable baseline with high interpretability, allowing clear insights into feature importance. SVM is effective for handling complex and non-linear patterns. Decision Tree provides intuitive decision-making pathways, while Random Forest enhances accuracy and reduces overfitting. KNN uses proximity-based classification, making it effective for handling non-linear relationships in the data. By leveraging these models' complementary strengths, the analysis ensures comprehensive evaluation and selection of the best-performing model for accurate and actionable predictions. Performance metrics for each model were calculated and compared across both scenarios to gain insights on how outliers influence model effectiveness and which model performs best in predicting the buying decision of homebuyers. Python libraries including pandas, scikit-learn, matplotlib, and seaborn were used for this project. To ensure compatibility with predictive models, categorical columns were identified and encoded using LabelEncoder. This transformation converted categorical variables into numerical representations suitable for analysis and to enhance model interpretability.

Encoded Dataset:

property_scope	availability	location	bedroom	total_squarefeet	bath	balcony	buying_decision	BER	Renovation needed	price_per_squarefeet ($)	total_price ($)	price_per_bedroom ($)
1	0	2	2.0	1056.0	2.0	1.0	0	0	1	419.93	443446.08	221723.04
0	2	1	3.0	1440.0	2.0	3.0	0	6	2	488.68	703699.20	234566.40
1	2	4	3.0	1521.0	3.0	1.0	0	6	2	708.91	1078252.11	359417.37
1	2	0	2.0	1200.0	2.0	1.0	0	5	2	482.38	578856.00	289428.00
1	2	1	2.0	1170.0	2.0	1.0	1	6	2	368.63	431297.10	215648.55

Figure 14: Encoded dataset

4.1 Predictive models implementation

- Logistic Regression: Logistic regression was implemented with 1000 iterations to ensure convergence. The model's accuracy was evaluated, and coefficients were analysed to interpret feature importance.

Accuracy: 0.6928276380022531

Classification Report:

	precision	recall	f1-score	support
0	0.69	1.00	0.82	1838
1	0.82	0.01	0.02	825
accuracy			0.69	2663
macro avg	0.76	0.50	0.42	2663
weighted avg	0.73	0.69	0.57	2663

Figure 15: Accuracy with outliers

Accuracy: 0.6866852886405959

Classification Report:

	precision	recall	f1-score	support
0	0.69	1.00	0.81	1462
1	0.74	0.03	0.06	686
accuracy			0.69	2148
macro avg	0.71	0.51	0.43	2148
weighted avg	0.70	0.69	0.57	2148

Figure 16: Accuracy without outliers

- Random Forest: A Random Forest Classifier was trained with 100 estimators. Feature importance was derived to highlight influential variables, and the model's accuracy was evaluated.

Accuracy: 0.7236199774690198

Classification Report:

	precision	recall	f1-score	support
0	0.74	0.92	0.82	1838
1	0.62	0.29	0.39	825
accuracy			0.72	2663
macro avg	0.68	0.60	0.61	2663
weighted avg	0.70	0.72	0.69	2663

Figure 17: Accuracy with outliers

Accuracy: 0.7239292364990689

Classification Report:

	precision	recall	f1-score	support
0	0.74	0.92	0.82	1462
1	0.64	0.31	0.41	686
accuracy			0.72	2148
macro avg	0.69	0.61	0.62	2148
weighted avg	0.71	0.72	0.69	2148

Figure 18: Accuracy without outliers

- Decision Tree: A Decision Tree Classifier with a maximum depth of 5 was used to prevent overfitting. Insights into the dataset's decision-making process were obtained via feature importance analysis. Performance matrix of the model has been obtained as well.

Accuracy: 0.7600450619601953

Classification Report:

	precision	recall	f1-score	support
0	0.74	0.99	0.85	1838
1	0.95	0.24	0.38	825
accuracy			0.76	2663
macro avg	0.85	0.62	0.62	2663
weighted avg	0.81	0.76	0.71	2663

Figure 19: Accuracy with outliers

Accuracy: 0.7602420856610801

Classification Report:

	precision	recall	f1-score	support
0	0.74	1.00	0.85	1462
1	0.99	0.25	0.40	686
accuracy			0.76	2148
macro avg	0.86	0.63	0.63	2148
weighted avg	0.82	0.76	0.71	2148

Figure 20: Accuracy without outliers

- Support Vector Machines (SVM): The RBF kernel was applied to capture non-linear patterns in the data. Performance matrix was realised and feature scaling was performed to normalize the data, ensuring fairness in margin calculations.

Accuracy: 0.7108524220803605

	precision	recall	f1-score	support
0	0.71	0.99	0.82	1838
1	0.77	0.09	0.17	825
accuracy			0.71	2663
macro avg	0.74	0.54	0.50	2663
weighted avg	0.73	0.71	0.62	2663

Figure 21: Accuracy with outliers

Accuracy: 0.7057728119180633

	precision	recall	f1-score	support
0	0.71	0.97	0.82	1462
1	0.71	0.13	0.22	686
accuracy			0.71	2148
macro avg	0.71	0.55	0.52	2148
weighted avg	0.71	0.71	0.63	2148

Figure 22: Accuracy without outliers

- K-Nearest Neighbours (KNN): The KNN algorithm was configured with 5 neighbours to balance bias and variance. The proximity-based predictions were evaluated against test data for accuracy.

Accuracy: 0.7014645137063462

Classification Report:

	precision	recall	f1-score	support
0	0.74	0.87	0.80	1838
1	0.53	0.32	0.40	825
accuracy			0.70	2663
macro avg	0.64	0.60	0.60	2663
weighted avg	0.68	0.70	0.68	2663

Figure 23: Accuracy with outliers

Accuracy: 0.6750465549348231

Classification Report:

	precision	recall	f1-score	support
0	0.72	0.85	0.78	1462
1	0.49	0.31	0.38	686
accuracy			0.68	2148
macro avg	0.60	0.58	0.58	2148
weighted avg	0.65	0.68	0.65	2148

Figure 24: Accuracy without outliers

The results for these predictive models have been discussed in the section 5.

4.2 Techniques to improve model accuracy

To enhance the performance of predictive models, several techniques were applied, including hyperparameter tuning (IBM, 2024), feature selection and cross-validation (Scikit-learn, n.d.). Only 2 algorithms that demonstrated an initial accuracy of more than 70% were considered for further tuning and optimisation. This strategy ensured a focus on models with sufficient baseline performance, reducing time spent on less promising techniques while achieving valuable predictive results.

For Random Forest, accuracy was improved through hyperparameter tuning using GridSearchCV. Key parameters such as the number of estimators (n_estimators),

maximum depth (max_depth), and feature selection criteria (max_features) were optimised. Cross-validation ensured that the tuned model generalised well to unseen data. The tuned Random Forest achieved a significant accuracy improvement. Lasso regression identified the most important predictors by penalising irrelevant features, improving interpretability and model performance. Selected features were used to retrain the Random Forest, yielding an even higher accuracy. Decision Tree optimisation involved hyperparameter tuning using GridSearchCV. Parameters such as maximum depth (max_depth), minimum samples per split (min_samples_split), and splitting criterion (criterion) were fine-tuned. The best Decision Tree model achieved high accuracy, further validated through cross-validation.

5. Results, Evaluation, and Discussion

Each model was evaluated using key performance indicators such as accuracy, precision, recall and F1-score. The classification reports provided detailed insights into the models' predictive capabilities, while accuracy scores allowed for comparative benchmarking. This was done before handling outliers and after handling outliers to check for the effect of outliers on the predictive models.

The results first evaluated each of the predictive models for their accuracy to find which model performs well for the scenario under study. Based on their performance 2 best performing models were chosen for further enhancing of their accuracy by methods like hyper parameter tuning, feature selection using Lasso, PCA and cross validation. The results were as follows:

5.1 Before handling outliers:

The highest accuracy was achieved using the Decision Tree algorithm, which recorded a performance of 76%. This was closely followed by the Random Forest algorithm with an accuracy of 72%, while the SVM and KNN algorithms delivered accuracies of 71% and 70%, respectively.

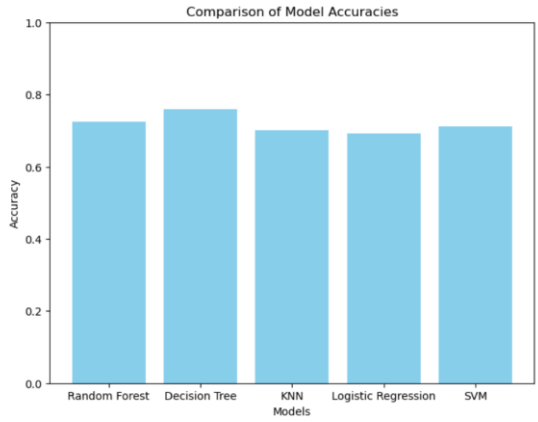

Figure 25: Comparing model accuracies

After analyzing the performance of various models, the Decision Tree and Random Forest algorithms were identified as the top performers, showing relatively high accuracies of 76% and 72%, respectively. Given their strong initial performance, these two models were selected for further enhancement through performance tuning.
When applying performance tuning techniques such as hyperparameter optimization (e.g., adjusting tree depth, the number of estimators, or learning rate), the Decision Tree model showed a marginal increase in accuracy, reaching 76.1%. While this improvement is positive, it remains minimal, suggesting that the Decision Tree model may have already reached its performance limit under the current conditions.
On the other hand, the Random Forest model experienced a more significant improvement in its accuracy, rising to 76.3%. This suggests that performance tuning was more effective for Random Forest, likely due to its ensemble nature, which benefits more from adjustments to parameters like the number of trees and the depth of individual trees. The improvement, though slight, reflects the model's ability to leverage performance tuning to capture more complex patterns in the data, ultimately enhancing its predictive power.
While both models showed potential, Random Forest benefited more noticeably from the tuning process, further solidifying its position as the preferred model for this particular dataset.

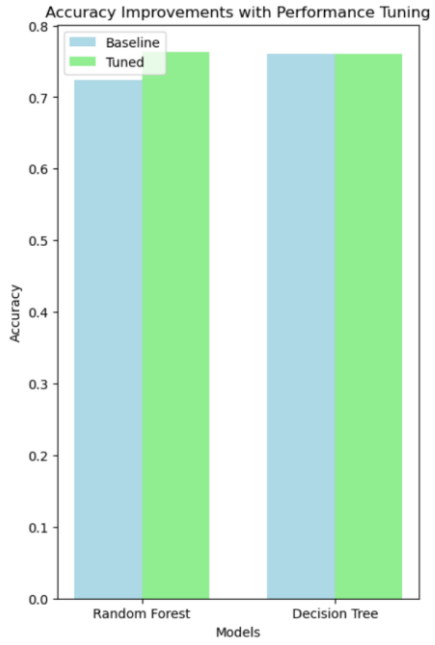

Figure 26: Comparing accuracies after performance tuning

5.2 After handling outliers:

The highest accuracy was achieved using the Decision Tree algorithm, which recorded a performance of 76%. This was closely followed by the Random Forest algorithm with an accuracy of 72%, while the SVM and KNN algorithms delivered accuracies of 70% and 67%, respectively.

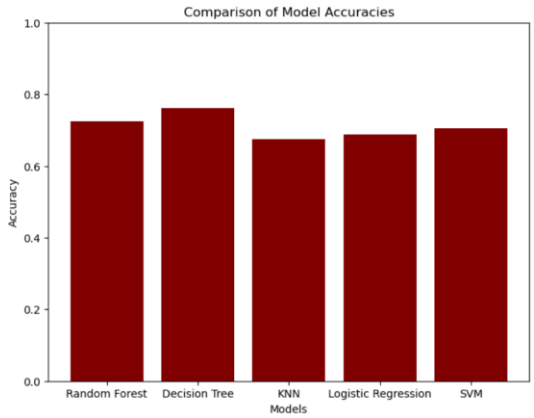

Figure 27: Comparing model accuracies

After analysing the performance of various models, the Decision Tree and Random Forest algorithms were identified as the top performers, showing relatively high accuracies of 76% and 72%, respectively. Given their strong initial performance, these two models were selected for further enhancement through performance tuning. When applying performance tuning techniques such as hyperparameter optimization (e.g., adjusting tree depth, the number of estimators, or learning rate), the Decision

Tree model showed a marginal increase in accuracy, reaching 76.1%. While this improvement is positive, it remains minimal, suggesting that the Decision Tree model may have already reached its performance limit under the current conditions.
On the other hand, the Random Forest model experienced a more significant improvement in its accuracy, rising to 75.7%. This suggests that performance tuning was more effective for Random Forest, likely due to its ensemble nature, which benefits more from adjustments to parameters like the number of trees and the depth of individual trees. Even though an improvement was observed to random forest model, for this scenario, decision tree proved to be more efficient than random forest. While both models showed potential, Random Forest benefited more noticeably from the tuning process, but decision tree position as the preferred model for this particular dataset considering the scenario.

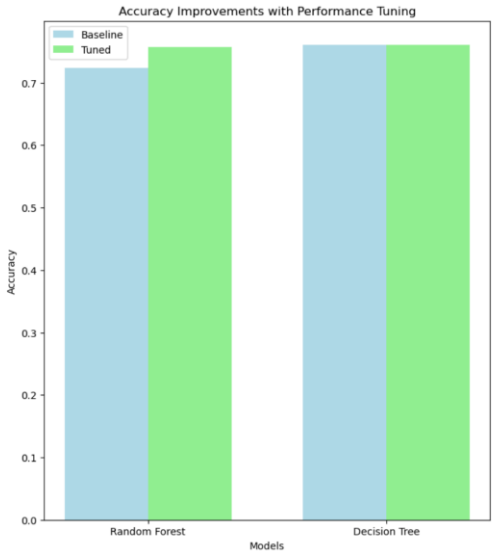

Figure 28: Comparing accuracies after performance tuning

In conclusion, the analysis and evaluation of predictive models before and after handling outliers in this dataset revealed valuable insights. While outliers were present in several key variables, their removal did not result in significant improvements in the accuracy of the models. In fact, for algorithms like Support Vector Machine (SVM) and K-Nearest Neighbours (KNN), the accuracy slightly decreased after the outliers were removed. This suggests that the models were somewhat robust to the presence of outliers and that their removal might not always lead to better performance.

For models like Decision Tree and Random Forest, which showed better performance with accuracies of 76% and 72% respectively, the removal of outliers did not result in notable changes. These models appear to be more resilient to the effects of outliers, highlighting their ability to generalize well to the underlying patterns in the data. Ultimately, the decision to handle outliers may not always lead to improved model performance, and the effectiveness of outlier removal can depend on the specific characteristics of the dataset and the models being used. Therefore, careful consideration should be given to whether outlier removal is necessary, as it might not always yield tangible benefits in terms of predictive accuracy. The findings suggest that more attention should be focused on other aspects of model tuning and feature engineering for improving model performance.

5.3 Model Recommendation:

Decision Tree: Due to its simplicity, interpretability, and consistently high accuracy, the Decision Tree model is recommended as the primary model for this dataset. It effectively captures key patterns without overfitting, making it suitable for scenarios requiring quick decision-making and straightforward explanations.

Random Forest: For scenarios requiring higher predictive power and robustness, especially when dealing with more complex data patterns, the Random Forest model is recommended. While slightly more computationally intensive, its ensemble approach provides a slight edge in capturing intricate relationships in the dataset.

Referencing

1. IBM (2024). *What is hyperparameter tuning? IBM Think.* Available at: https://www.ibm.com/think/topics/hyperparameter-tuning. (Accessed: 10 November 2024).
2. Hastie T., Tibshirani R., & Friedman J., (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://codowd.com/bigdata/misc/ESLII.pdf
3. Aggarwal, C. C. (2017). *Outlier analysis* (2nd ed.). Springer. https://doi.org/10.1007/978-3-319-47578-3_1
4. Zheng, A., & Casari, A. (2018). *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*. O'Reilly Media. https://www.oreilly.com/library/view/feature-engineering-for/9781491953235
5. Scikit-learn (n.d.). *Cross-validation: evaluating estimator performance*. Available at: https://scikit-learn.org/stable/modules/cross_validation.html#cross-validation (Accessed: 13 November)