CITY UNIVERSITY
LONDON

Online Shoppers Purchasing Intention Prediction

Introduction to Artificial Intelligence
INM701

Alessandro Abati (student ID: 230040125)

January 11, 2024

GitHub repository

Contents

Introduction

In the evolving landscape of e-commerce, accurately predicting online shoppers' purchasing intentions poses a significant challenge. This report examines the use of advanced Artificial Intelligence (AI) techniques to predict purchase likelihood on an e-commerce platform. We analyze a dataset rich in both numerical and categorical metrics of user interactions, applying machine learning algorithms such as Decision Trees, Random Forests, and Neural Networks to identify patterns indicative of purchasing behavior.

The report is structured to first present an exploratory data analysis (EDA), providing crucial insights into the dataset. We address the challenge of class imbalance in the data, employing strategies like SMOTE to improve model accuracy. Following this, we details our approach to data preprocessing, performance evaluation and model selection, underlining our methodological rigor.

Concluding with a comparative analysis and a discussion on feature importance, the report outlines our efforts in leveraging AI to understand the complexities of online purchasing behavior.

1 Exploratory Data Analysis (EDA)

The chosen dataset (Sakar et al. 2019) [1] provides both numerical (10) and categorical (8) features, including various aspects of user interaction and behavior on an e-commerce platform.

The dataset includes 12205 rows after removing 125 duplicates[2]; each row represents a unique user session on the website. The binary 'Revenue' attribute (class label), is central to our study, indicating whether a user session resulted in a transaction.

1.1 Numerical Features Statistical Analysis and Visualisation

The numerical features, along with their statistics, are shown in Table 1. Key numerical features include 'Administrative', 'Informational', and 'Product Related' attributes along with their respective durations. These metrics offer insights into the type and extent of users' engagement with different page categories. Furthermore, the dataset includes 'Bounce Rate', 'Exit Rate', and 'Page Value' as defined by Google Analytics. The "Bounce Rate" measures the proportion of visitors who land on a page and exit without any further interactions. The 'Exit Rate' percentage reflects the tendency of users to leave the website from the particular page viewed during that session. "Page Value" is an average value assigned to a page, based on the user visits preceding an e-commerce transaction. The 'Special Day' feature quantifies the proximity of user visits to significant days like Valentine's Day, thus implying potential transactional behavior. (Sakar et al. 2019)

Feature	Mean	Std Dev	Min	Max
Administrative	2.338878	3.330436	0.0	27.000000
Administrative_Duration	81.646331	177.491845	0.0	3398.750000
Informational	0.508726	1.275617	0.0	24.000000
Informational_Duration	34.825454	141.424807	0.0	2549.375000
ProductRelated	32.045637	44.593649	0.0	705.000000
ProductRelated_Duration	1206.982457	1919.601400	0.0	63973.522230
BounceRates	0.020370	0.045255	0.0	0.200000
ExitRates	0.041466	0.046163	0.0	0.200000
PageValues	5.949574	18.653671	0.0	361.763742
SpecialDay	0.061942	0.199666	0.0	1.000000

Table 1: Descriptive Statistics of Numerical Web Features

The distributions in Figure 1 illustrate that sessions resulting in revenue generally show a slightly higher number of pages visited across 'Administrative', 'Informational', and 'Product Related' categories, indicating a positive association between content engagement and transaction likelihood. This trend extends to the 'Duration' attributes for these page types. Notably, sessions without revenue exhibit higher 'Bounce Rates', suggesting that a lack of engagement across multiple pages often precedes non-transactional sessions. The 'Exit Rate' follows a similar pattern, with lower rates corresponding to revenue-generating sessions, highlighting more extensive site navigation before session termination. Moreover, 'Page Value' stands out as a critical metric, with pages visited prior to a purchase displaying significantly higher values, indicating their essential role in leading to a successful sale. Finally, the 'Special Day' feature shows that sessions close to certain holidays are more likely to end in a purchase, yet transactions are not limited to these periods, indicating a steady purchasing trend throughout the year.

[1] The dataset has been downloaded from Kaggle (`https://www.kaggle.com/datasets/imakash3011/online-shoppers-purchasing-intention-dataset/data`)

[2] The rational behind this choice is that having two identical session in all the features (and the label) is unlikely

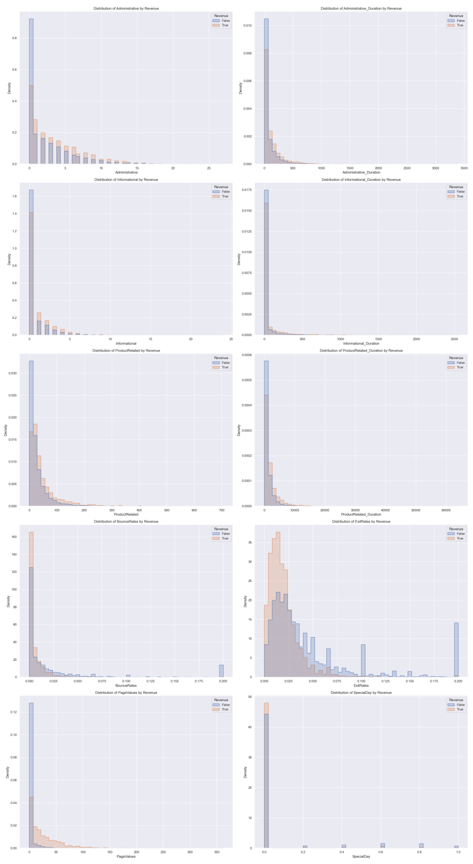

Figure 1: Distributions of numerical features by revenue.

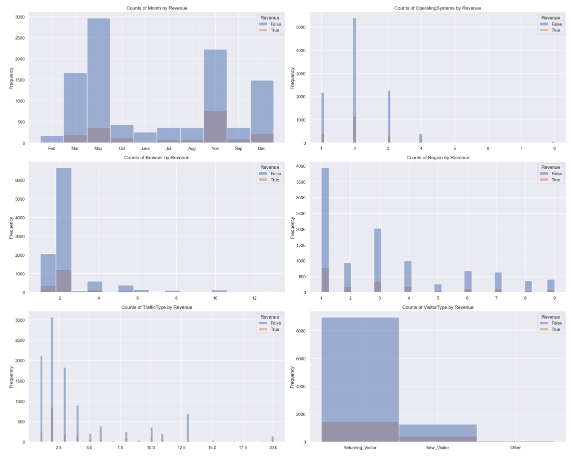

Figure 2: Histograms showing the distribution of different categorical features by revenue.

1.2 Categorical Features Statistical Analysis and Visualisation

Additionally, the dataset includes categorical features such as 'Operating System', 'Browser', 'Region', and 'Traffic Type', which are crucial in understanding the demographic and technological profiles of the users. The histograms in Figure 2 show clear trends in buying behavior on the e-commerce site. Sales peak during November and May, which could be related to holiday shopping or to specific sales campaigns. People using certain operating systems and browsers tend to buy more, which might suggest that these systems are easier to use or offer a better shopping experience. Purchases also vary by region, possibly because of special deals or shipping options in those areas. How users get to the site matters too; some methods, like direct visits, lead to more sales, highlighting the importance of good marketing. Finally, the data shows that people who have shopped on the site before are more likely to buy again, pointing to the value of keeping customers coming back.

1.3 Dataset Imbalance

A critical aspect of our dataset is its significant class imbalance. In our dataset, only a small fraction of sessions (18%) result in transactions. This imbalance can skew predictive modeling, leading to a bias towards the majority class. To mitigate this, we plan to implement SMOTE (Chawla et al. 2002) to oversample the minority class, balancing the distribution between transactional and non-transactional sessions. This approach generates synthetic samples that are plausible and representative, improving the model's ability to generalize from a balanced dataset.

1.4 Correlation Matrix

A correlation matrix quantifies the degree and direction of the linear relationship between pairs of variables. Our matrix (Figure 3) evaluates these relationships, identifying variables that exhibit a significant correlation with the 'Revenue' attribute, which may suggest predictability but not causality.

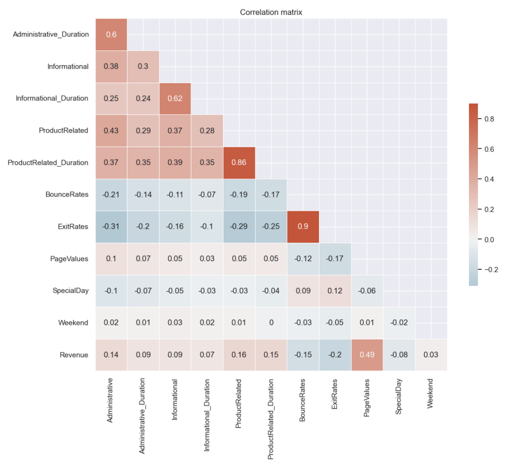

Figure 3: Correlation matrix of numerical features, indicating potential predictors for 'Revenue'.

The matrix highlights that the correlation coefficients between the 'Revenue' attribute and other features are relatively low, indicating the presence of complex, possibly non-linear patterns, between the label and the features.

2 Method and Result

Our exploratory analysis revealed significant challenges in the dataset, including class imbalance and the presence of complex, non-linear relationships. In response, we have selected Decision Trees, Random Forests, and Neural Networks for our classification task.

2.1 Data Preprocessing

The data preprocessing phase is crucial for preparing the dataset for machine learning algorithms. This step involved specific techniques to handle categorical variables and the unique nature of certain features, as described below.

One-Hot Encoding of Categorical Features: The dataset included several categorical variables, such as 'OperatingSystems', 'Browser', 'Region', 'VisitorType' and 'TrafficType'. To make these features compatible with our models[3], we applied One-Hot encoding (Scikit-learn 2011*d*). This technique involves creating a new binary column for each category level in the original feature. This approach ensures that the models can process these categorical features without assuming any ordinal relationship between them.

[3]Tree-based models can handle categorical features even with a simpler LabelEncoder (Scikit-learn 2011*c*); however, we decided to use One-Hot Encoding to maintain consistency in the feature space and have a fairer comparison between all the models.

Cycle Encoding of 'Month' Feature: Recognizing the cyclical nature of months, we applied cycle encoding to the 'Month' feature. Traditional numerical encoding would inadequately represent the relationship between months due to their recurring pattern. To address this, we used sine and cosine transformations to map each month to a point on a circle, effectively capturing the cyclical progression of months. This technique is particularly important in modeling e-commerce data, as it can allow the algorithms to recognize and leverage seasonal trends in online shopping behavior.

2.2 Model Selection

We initially established baseline models before progressing to more sophisticated algorithms. This approach not only aligns with best practices in machine learning but also allows for a better understanding of model performance in relation to our specific dataset.

2.2.1 Performance Metrics

In assessing the performance of our models, we selected the following metrics which account for the imbalanced nature of our dataset and the specific needs of an e-commerce context.

Confusion Matrix: The confusion matrix is a tabular representation of an algorithm's performance, showing the the number of correct and incorrect predictions for each class. The matrix is divided into four parts: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) [4].

Precision: Precision measures the proportion of true positive predictions in the positive class. High precision indicates a low rate of false positives, crucial in avoiding the misidentification of non-purchasing visitors as potential buyers.

Recall: Recall is defined as the ratio of true positives (TP) to the sum of true positives (TP) and false negatives (FN). High recall reflects the model's ability to correctly identify actual purchasers.

F1 Score: The F1 Score is the harmonic mean of precision and recall. It provides a more reliable performance measure, especially in the context of our imbalanced dataset, by ensuring that both false positives (incorrectly targeting non-buyers) and false negatives (missing out on potential buyers) were minimized. A high F1 score indicates a balanced model with both high precision and high recall.

2.2.2 Baseline Model - Random Classifier

We initiated our model evaluation with a Random Classifier[5] baseline. This model's performance sets a foundational benchmark for comparison with more advanced models. The results are summarized in Table 2 and illustrated in the confusion matrix (Figure 4). Notably, the imbalance between precision and recall, particularly for the 'purchase' class (Precision: 0.14, Recall: 0.46), indicates the model's tendency to misclassify a significant number of non-purchase instances as purchases, a direct consequence of the class imbalance in the dataset.

2.2.3 Decision Tree Classifier

Given the class imbalance revealed during EDA, our first advanced model choice was the Decision Tree Classifier (Scikit-learn 2011*f*), which intrinsically perform well with imbalanced data because

[4]True Positives (TP) are instances correctly predicted as the positive class (sessions resulting in a purchase). False Positives (FP) are instances incorrectly predicted as positive. True Negatives (TN) are instances correctly predicted as the negative class (sessions not resulting in a purchase). False Negatives (FN) are instances incorrectly predicted as negative.

[5]Makes predictions by randomly assigning a class based on a uniform distribution.

Metric	Non-Purchases ('False')	Purchases ('True')
Precision	85%	14%
Recall	51%	46%
F1-Score	64%	22%

Table 2: Random Classifier Baseline Performance

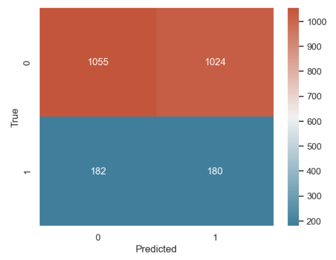

Figure 4: Confusion Matrix for Random Classifier

the splitting rules are sensitive to the class distributions in the subsets of the data they create. Moreover, the decision tree provides a clear visual representation of the decision-making process, making it easier to interpret and analyze the model's behavior.

This model uses the greedy recursive binary splitting alogorithm where the best split at each node is the one that maximise the *Information Gain* (Equation 2), thus maximizing the purity of the subsets and ensuring the resulting nodes are as homogeneous as possible (Mitchell 1997). In this project, we used the entropy criterion for splitting which measures the degree of impurity in the dataset:

$$Entropy(S) = -\sum_{i=1}^{n} p_i \log_2 p_i = -p_\oplus \, log_2 p_\oplus - p_\ominus \, log_2 p_\ominus \tag{1}$$

where S is the set of samples, n is the number of classes, and p_i is the proportion of the samples that belong to class i [6]. Therefore, for an attribute A:

$$InformationGain(S, A) \; := \; Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \tag{2}$$

where $Values(A)$ is the set of all possible values for attribute A, and S_v, is the subset of S for which attribute A has value v (Mitchell 1997). Hence, *Information Gain* is defined as the expected reduction in entropy; that is, the difference between $Entropy(S)$, the overall entropy of S, and the expected entropy for the subset S_v. After a few experiments, we set a maximum tree depth of 10 and required a minimum of 50 samples at each leaf node to prevent overfitting. Moreover, we standardized the features using a StandardScaler (Scikit-learn 2011*e*) to ensure uniformity in variable scales, an essential step given the varied nature of the dataset's features.

Metric	Non-Purchases ('False')	Purchases ('True')
Precision	93%	71%
Recall	96%	58%
F1-Score	94%	64%

Table 3: Decision Tree Classifier Performance

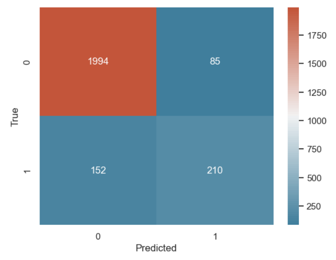

Figure 5: Confusion Matrix for Decision Tree Classifier

[6]The last equivalence reflects our binary classification problem

As shown in Table 3, the model exhibited a high precision of 93% and recall of 96% for non-purchases, confirming its efficacy in correctly predicting the majority class. Conversely, for the purchasing class, precision and recall were lower at 71% and 58%, respectively, with a resulting F1-Score of 64%, indicating modest performance in identifying actual purchases. This is also reflected in the confusion matrix (Figure 5), which highlight 210 true positives and 152 false positive.

The Decision Tree's improved metrics over the baseline indicate its ability to discern patterns in the data. Yet, the modest detection of purchasing behaviour demands model tuning or the application of more advanced techniques (e.g., ensemble methods) for a deeper analysis of the data.

SMOTE oversampling

As stated in Section 1.3, we used SMOTE oversampling technique to handle the imbalance in the dataset, including it in the pipeline after the standardization step. As shown in Table 4, SMOTE improved the detection of actual purchasing sessions, with the recall for purchases rising to 73%, despite a modest decrease in precision to 58%. While the precision for non-purchases slightly increased to 95%, the recall saw a slight decline to 91%, suggesting a trade-off typically associated with addressing class imbalance. This is further evidenced by the rise in false positives, from 85 to 189, as shown in the confusion matrix (Figure 6). These adjustments underscore the efficacy of SMOTE in mitigating dataset imbalance, enhancing the model's sensitivity to the minority class, and potentially increasing revenue opportunities by better identifying likely buyers.

Metric	Non-Purchases ('False')	Purchases ('True')
Precision	95%	58%
Recall	91%	73%
F1-Score	93%	65%

Table 4: Decision Tree Classifier with SMOTE Oversampling Technique Performance

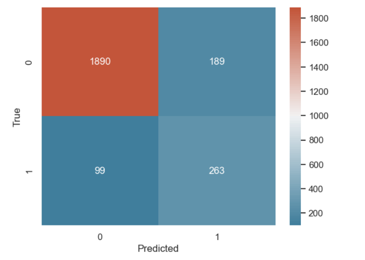

Figure 6: Confusion Matrix for Decision Tree Classifier with SMOTE Oversampling Technique

2.2.4 Random Forest Classifier

To build upon the decision tree model, we next implemented a Random Forest Classifier (Scikit-learn 2011*a*). This model, an ensemble of decision trees, operates by combine, using soft voting, the predictions of the individual trees, which are trained on random subsets of the features at each split [7]. This allows the model to add variability across the trees reducing their correlation; thus, decreasing both bias and variance by "averaging" the respective errors (Biau & Scornet 2016). This approach is particularly effective in our case, as it should capture more complex patterns within the dataset, leading to a more accurate prediction of purchasing intentions. Moreover, the comparison between the Random Forest and the Decision Tree is straightforward and provides insights into how ensemble methods can enhance performance.

In our experiment, we applied the Random Forest Classifier with 75 trees [8], using identical hyperparameters as our initial Decision Tree. This approach, along with SMOTE oversampling, should improve prediction stability and accuracy by reducing variance.

[7]This technique is known as feature bagging

[8]The hyperparameter corresponding to the number of estimators was determined through a manual tuning process

Metric	Non-Purchases ('False')	Purchases ('True')
Precision	96%	55%
Recall	89%	77%
F1-Score	92%	65%

Table 5: Random Forest Classifier with SMOTE Oversampling Technique Performance

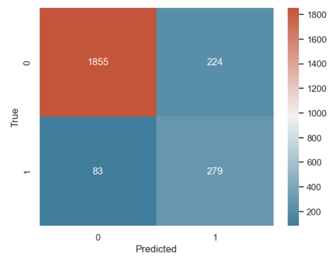

Figure 7: Confusion Matrix for Random Forest Classifier with SMOTE Oversampling Technique

The results shown in Table 5 and in the confusion matrix (Figure 7) illustrate that the Random Forest Classifier achieved similar performances to the Decision Tree Classifier considering the same preprocessing steps; in particular, the F1 score for the purchasing class is equivalent (65%). This unexpected behaviour may arise for different reasons and require some considerations. Firstly, the hyperparameter settings, optimal for the Decision Tree, may not extend their advantage to the Random Forest, suggesting a need for distinct, model-specific tuning. Secondly, if the dataset doesn't have complex relationships between features, the Random Forest's ability to deal with such complexities isn't needed, and therefore doesn't improve the results compared to a single Decision Tree. Finally, this outcome suggests we might have achieved the peak performance for our current dataset and feature set, potentially encountering the limit to model accuracy due to inherent irreducible data noise.

2.2.5 Neural Network

Neural Networks are a class of machine learning models, composed by a series of functions modeled on neurons, each taking a weighted sum of inputs and passing this through a non-linear activation function to produce an output. The power of neural networks lies in their ability to approximate complex non-linear relationships by adjusting weights through the backpropagation.

To compare the performance of a Neural Network with that of a tree based model, we must consider the same metrics across the same dataset and preprocessing steps. While Decision Trees offer interpretability and simplicity, Neural Networks provide flexibility and the capacity for capturing more complex non-linear interactions. The comparison thus hinges not only on performance metrics but also on the complexity of the model and the interpretability of the results.

Parameter	Value
Input Layer	128 Neurons, ReLU activation
Hidden Layer 1	256 Neurons, ReLU activation
Dropout	25% rate
Hidden Layer 2	64 Neurons, ReLU activation
Output Layer	1 Neuron, Sigmoid activation
Optimizer	Adam
Learning Rate	0.001
Loss Function	Binary Crossentropy
Batch Size	128
Epochs	100 with Early Stopping (patience: 15)

Table 6: Neural Network Architecture and Hyperparameters

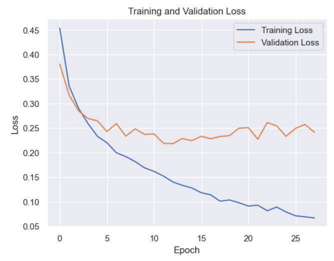

Figure 8: Training (in blue) and Validation (in orange) Loss Function of the Neural Network

For our implementation, we constructed a feedforward neural network using Keras (Chollet et al.

2015) with a TensorFlow backend. As shown in Table 6, the architecture consists of an input layer with 128 neurons, a dense hidden layer with 256 neurons, a dropout layer with 25% dropout rate to reduce overfitting and another dense hidden layer with 64 neurons. The output layer employs a sigmoid activation function, appropriate for binary classification tasks. For the training process, illustrated by the loss plot (Figure 8), we chose a Binary Crossentropy loss function with Adam optimizer (Keras 2015*b*). The graph shows a steady decline in training loss, indicating that the model is learning effectively from the data. The validation loss mirrors this trend, suggesting that the model is generalizing well to unseen data. The Early Stopping (Keras 2015*a*) mechanism, which stops training to prevent overfitting, was triggered after around 10 epochs.

Metric	**Non-Purchases ('False')**	**Purchases ('True')**
Precision	94%	53%
Recall	90%	64%
F1-Score	92%	58%

Table 7: Neural Network Performance

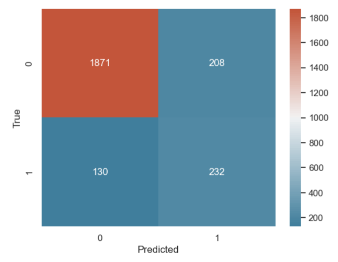

Figure 9: Confusion Matrix for Neural Network

The performance of the Neural Network are summarized in the confusion matrix (Figure 9) and in the metrics in Table 7. Although the model achieved a high precision of 94% and a recall of 90% for non-purchases, indicating a strong capability to correctly identify the majority of non-purchase sessions; for purchases, the precision was lower at 53%, and the recall was 64%. This indicates that while the model has a moderate ability to identify true purchasing sessions, it also tends to misclassify non-purchases as purchases more frequently than the Decision Tree and Random Forest models. These results suggest that while the Neural Network model has learned to differentiate between purchasing and non-purchasing behavior to some extent, the lower precision for purchases might be indicative of the model's sensitivity to the imbalanced data, even after applying SMOTE oversampling. Hence, the model could benefit from further tuning of the architecture, such as implementing a more sophisticated network or adjusting hyperparameters more accurately; however, as stated for the Random Forest results, the complexity of a neural network model may not be beneficial to deal with dataset that do not have complex relationships between features.

2.3 Results

Our comparative analysis of various models revealed that the Decision Tree Classifier is the most effective for predicting online shoppers' purchasing intentions. It outperformed others in terms of precision, recall, and F1 scores, especially in the purchase class. Despite the complexity of Random Forest and Neural Network models, they did not significantly surpass the Decision Tree in performance, underscoring the value of simpler, more interpretable models in our specific scenario.

3 Further Evaluation

3.1 Feature Importance and Comparison with Naive Classifier

One of the most important characteristic of Decision Tree models is their interpretability, which offers a transparent view into the decision-making logic of the model. In the case of predicting online shopping behavior, examining feature importance reveals, on average, which attributes most influence a user's decision to purchase. In our case, the importance of each feature is measured as the (normalized) total reduction of the entropy brought by that feature. The plot (Figure 10) shows

Metric	Non-Purchases ('False')	Purchases ('True')
Precision	92%	63%
Recall	94%	56%
F1-Score	93%	59%

Table 8: Naive Classifier Performance

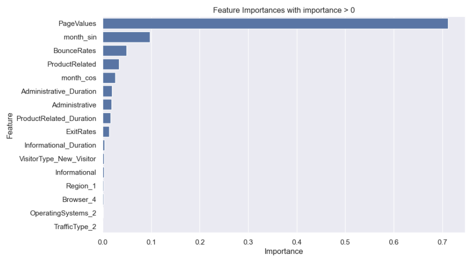

Figure 10: Decision Tree Feature Importance

that 'PageValues' stands out significantly compared to other features, indicating its crucial role in predicting customer purchasing intention. Given its dominance, it is interesting to examine how a Naive Classifier, which solely utilizes 'PageValues', performs against the fully-featured Decision Tree model. The Naive Classifier performance, shown in Table 8, reveals the Decision Tree's slight advantage (Table 3), which incorporates a broader context through multiple features, resulting in higher precision and recall for predicting purchases; however, despite its simplicity, the naive classifier underscores the critical influence of 'PageValues'.

3.2 Hyperparameters Tuning Experiment

Testing the hypothesis that model-specific hyperparameter optimization could enhance the Random Forest and Neural Network's predictive performance, a rigorous hyperparameters tuning experiment was undertaken using an optimization criterion of maximising the weighted F1 score. The Random Forest model was tuned using HalvingRandomSeach (Scikit-learn 2011*b*); similarly, the NN was tuned using an efficient Random Search algorithm (Li et al. 2018) through Keras Tuner (Keras 2015*c*). Despite these efforts, the models failed to outperform the Decision Tree model, reinforcing the hypothesis that the data may possess an irreducible noise error.

4 Conclusion

This report has methodically investigated the predictive modeling of online shoppers' purchasing intentions, providing valuable insights into consumer behavior by leveraging advanced AI methodologies. We have examined the dataset, applied exploratory data analysis, and engaged with various machine learning models, adjusting for class imbalance with SMOTE. Our findings reveal the predominance of 'PageValues' in influencing purchasing decisions. The Decision Tree Classifier emerged as the most effective model, suggesting simplicity can often rival complexity in predictive power. Moreover, despite hyperparameters tuning, neither Random Forest nor Neural Network models could outperform the Decision Tree, reinforcing the presence of an irreducible noise error within the dataset. Future work may explore additional fine tuning of the models, alternative imbalance mitigation techniques, feature engineering or the integration of unsupervised learning for further enhancement of predictive performance.

References

Biau, G. & Scornet, E. (2016), 'A random forest guided tour', *Test* **25**, 197–227.

Chawla, N., Bowyer, K., Hall, L. & Kegelmeyer, W. (2002), 'Smote: Synthetic minority over-sampling technique', *J. Artif. Intell. Res. (JAIR)* **16**, 321–357.

Chollet, F. et al. (2015), 'Keras', https://github.com/fchollet/keras. Accessed: 2024-01-08.

Keras (2015*a*), 'keras.callbacks.earlystopping', https://keras.io/api/callbacks/early_stopping/. Accessed: 2024-01-07.

Keras (2015*b*), 'keras.optimizers.adam', https://keras.io/api/optimizers/adam/. Accessed: 2024-01-07.

Keras (2015*c*), 'keras_tuner.hyperband', https://keras.io/api/keras_tuner/tuners/hyperband/. Accessed: 2024-01-08.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A. & Talwalkar, A. (2018), 'Hyperband: A novel bandit-based approach to hyperparameter optimization', *Journal of Machine Learning Research* **18**(185), 1–52.
URL: *http://jmlr.org/papers/v18/16-558.html*

Mitchell, T. (1997), *Machine Learning*, McGraw-Hill International Editions, McGraw-Hill.
URL: *https://books.google.co.uk/books?id=EoYBngEACAAJ*

Sakar, C. O., Polat, S. O., Katircioglu, M. & Kastro, Y. (2019), 'Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and lstm recurrent neural networks', *Neural Computing and Applications* **31**, 6893–6908.
URL: *https://api.semanticscholar.org/CorpusID:13682776*

Scikit-learn (2011*a*), 'sklearn.ensemble.randomforestclassifier', https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html. Accessed: 2024-01-02.

Scikit-learn (2011*b*), 'sklearn.model_selection.halvingrandomsearchcv', https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.HalvingRandomSearchCV.html. Accessed: 2024-01-08.

Scikit-learn (2011*c*), 'sklearn.preprocessing.labelencoder', https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html. Accessed: 2024-01-02.

Scikit-learn (2011*d*), 'sklearn.preprocessing.onehotencoder', https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html. Accessed: 2024-01-02.

Scikit-learn (2011*e*), 'sklearn.preprocessing.standardscaler', https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html. Accessed: 2024-01-02.

Scikit-learn (2011*f*), 'sklearn.tree.decisiontreeclassifier', https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html. Accessed: 2024-01-02.