Data Mining Project using CRISP-DM

E-commerce classification Model Performance Evaluation

Hakim Shaibu 1641405

Contents

[Business Understanding 3](#_Toc195272589)

[Project Objectives 3](#_Toc195272590)

[Background 3](#_Toc195272591)

[Success Criteria 3](#_Toc195272592)

[Primary Problem 3](#_Toc195272593)

[stakeholders 3](#_Toc195272594)

[Constraints 4](#_Toc195272595)

[Data Understanding 4](#_Toc195272596)

[Data Collection 4](#_Toc195272597)

[Data Description 4](#_Toc195272598)

[Key features include 4](#_Toc195272599)

[Initial Exploration 5](#_Toc195272600)

[Data Preparation 5](#_Toc195272601)

[Modelling 6](#_Toc195272602)

[Random Forest 6](#_Toc195272603)

[Parameter tuning for Random Forest 6](#_Toc195272604)

[XGboost 6](#_Toc195272605)

[Hyperparameters for XGboost 6](#_Toc195272606)

[KNN 7](#_Toc195272607)

[Hyperparameters for KNN 7](#_Toc195272608)

[Dummy classifier model 8](#_Toc195272609)

[Hyperparameters of Dummy classifier 8](#_Toc195272610)

[Training and Validation 8](#_Toc195272611)

[Summary of split 8](#_Toc195272612)

[Model evaluation metrics 9](#_Toc195272613)

[Comparison with Baseline 9](#_Toc195272614)

[Evaluation of Performance Metrics 10](#_Toc195272615)

[KNN Performance 10](#_Toc195272616)

[Random Forest Performance 11](#_Toc195272617)

[XGboost Performance 12](#_Toc195272618)

[Dummy Classifier Results 13](#_Toc195272619)

[Strengths and Weaknesses of Models 14](#_Toc195272620)

[Strengths 14](#_Toc195272621)

[Weakness 14](#_Toc195272622)

[Biases 14](#_Toc195272623)

[Deployment 14](#_Toc195272624)

[Deployment Strategy 14](#_Toc195272625)

[Monitoring 14](#_Toc195272626)

[Future Work 15](#_Toc195272627)

[Conclusion 15](#_Toc195272628)

# Business Understanding

## Project Objectives

The goal of this project is to predict whether a visitor to an e-commerce website will make a purchase. This is done by analysing various session-related variables, such as time spent on specific types of pages, clickstream data, traffic source, and visitor behaviour.

## Background

Understanding customer behaviour is important to be able to navigate in the world of e-commerce. Investing in the wrong marketing or website optimisation strategy can be very costly. And even if done right they often struggle to convert visitors into buyers. By building a predictive model that can estimate purchase intent based on online behaviour, businesses can better target potential customers and reduce bounce rates.

 The dataset used for this project consists of over 12,000 session records from an e-commerce site, with both numerical and categorical features. The class label (Revenue) indicates whether a purchase was made.

## Success Criteria

Success will be measured using the accuracy, precision, f1-score and recall. This will ensure that it makes correct predictions and correctly identify true purchasing sessions without misclassifying non-purchasing ones.

## Primary Problem

The main business problem is increasing conversion rates on e-commerce websites. By predicting whether a session is likely to end in a purchase, businesses can make smarter decisions, such as targeting users with personalized offers or improving the user journey.

## stakeholders

The key stakeholders are e-commerce business owners who want to increase sales, digital marketers and data analysts who interpret user behaviour to guide decision-making.

## Constraints

Before conducting the analysis, the dataset presented clear constraints. One major issue was the imbalance in the target variable revenue. with class 0 being the majority (no purchase) and class 1 the minority(purchase). This imbalance will likely restrict the model’s capacity to effectively learn patterns for class 1, leading to potential underperformance in minority class predictions.

Another significant constraint was the quality of the features. The data collection process could introduce challenges such as missing values, outliers, or irrelevant variables, which may negatively impact the model’s accuracy and reliability.

# Data Understanding

## Data Collection

The dataset was sourced from Kaggle, where it was published for academic and analytical purposes. It consists of anonymized session data from an e-commerce website over a one-year period. Each row represents a unique user session to avoid biases from repeated visits by the same user.

## Data Description

The dataset consists of feature vectors belonging to 12,330 sessions. The dataset was formed so that each session would belong to a different user in a 1-year period to avoid

any tendency to a specific campaign, special day, user profile, or period.

### Key features include

1. Administrative: Count of administrative pages visited by the user.
2. Administrative\_Duration: Total time spent on administrative pages.
3. Informational: Count of informational pages visited by the user.
4. Informational Duration: Total time spent on informational pages.
5. Product Related: Count of product-related pages visited by the user.
6. ProductRelated\_Duration: Total time spent on product-related pages.
7. BounceRates: Percentage of visitors leaving without interacting further on the entry page.
8. Exit Rates: Percentage of pageviews where users exited the site on that specific page.
9. Page Values: Average monetary value assigned to a page based on eCommerce success.
10. Special Day: Measure of proximity to a special holiday or event impacting user transactions.
11. Month: Month in which the pageview occurred (stored as a string).
12. Operating Systems: Integer representation of the user's operating system.
13. Browser: Integer representation of the user's browser choice.
14. Region: Integer representation of the user’s geographic region.
15. Traffic Type: Integer categorizing the user's type of traffic.
16. Visitor Type: Categorization of the user (New Visitor, Returning Visitor, or Other).
17. Weekend: Boolean indicating if the session occurred on a weekend.
18. Revenue: Boolean showing whether the user generated revenue (e.g., made a purchase).

## Initial Exploration

Summary statistics were generated, revealing no missing values but there were class imbalances and categorical features that required encoding for modelling. Visualizations such as histograms and count plots helped identify feature correlations.

The dataset was downloaded to Kaggle from UCI Machine Learning Repository. It was last updated 5 years ago

## Data Preparation

I first imported the necessary packages then loaded the data frame. After viewing the summary statistics of the data. Not much cleaning was needed. There were no missing values

The only cleaning that was required was changing objects into floats and integers. I also had changed the target column from Boolean to integers to keep everything uniform.  
Hence after cleaning the data frame was all numerical data types.

Features like Month and weekend were encoded into numerical format using Categorical type from pandas library.

The feature columns also had to be standardized because there were huge ranges within their values. After Using Standard scaler to normalise the values, the mean of all columns was closer to 0 and the standard deviation close to 1.

For the target column revenue Smote was used to address the issue of class imbalance in datasets, particularly for classification tasks which are going to be performed.  
SMOTE increases the number of samples in the minority class which is in this case Revenue column.

# Modelling

For this analysis I used the following models

* Random Forest model
* XGboost Model
* KNN Model

## Random Forest

Random Forest was used because of the huge number of features in the dataset. Random Forest is known to work well with high dimensional datasets. Random Forests are less prone to overfitting than single decision trees.

### Parameter tuning for Random Forest

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Value | Purpose |
| n\_estimators = | [50, 100, 200] | Number of trees |
| max\_depth | [10, 20, 30, none] | |  | | --- | |  |  |  | | --- | | Maximum tree depth | |
| Class-weight | ‘balalanced | Handle class imbalance automatically |
| Cross validation cv | 5 | |  | | --- | |  |   evaluate each hyperparameter combination. |
| random\_state | 42 | Reproducibility |
| Scoring metric | recall | Focus on correctly identifying Class 1 |

## XGboost

XGboost was selected because it delivers strong performance on tabular data and allows fine-grained control over model complexity. XGBoost builds an ensemble of decision trees sequentially, where each tree corrects the errors of the previous ones.

### Hyperparameters for XGboost

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Value | Purpose |
| n\_estimators = | [50, 100, 200] | Number of trees |
| max\_depth | [10, 20, 30, none] | |  | | --- | |  |  |  | | --- | | Maximum tree depth | |
| learning\_rate | | [0.01, 0.1, 0.2] | Shrinks the contribution of each tree |
| cv | 5 | evaluate each hyperparameter combination. |
| random\_state | 42 | reproducibility |
| Scoring metric | recall | Focus on correctly identifying Class 1 |
| scale\_pos\_weight | Calculated as (negative class count / positive class count) | Balances class imbalance |
| eval\_metric | 'logloss' | Evaluation metric for training |

## KNN

The dataset was chosen to revolve around the compatibility with the KNN since it was a model which had to be used. KNN was used for this because of its ease of implementation and interpretability. I plan to use the KNN as a baseline for the more complex models I plan to use.

### Hyperparameters for KNN

|  |  |  |
| --- | --- | --- |
| Hyper parameter | Value | Purpose |
| n \_neighbours | 19 | Number of neighbours to consider when predicting |

## Dummy classifier model

If our model is only slightly better than Dummy Classifier it means I will need better features or model tuning.

Even Though I have decided to use KNN as a baseline I have also written a code for a dummy classifier model to help us understand just how well our 3 models are doing. This was created by importing the Dummy classifier from the Sklearn package.

### Hyperparameters of Dummy classifier

|  |  |  |
| --- | --- | --- |
| Hyperparameter | Value | Purpose |
| Strategy | ‘most frequent ‘ | Always predict the class that appears most often in the training data |
| Random\_state | 42 | ensures that the results of random operations are reproducible |

# Training and Validation

Before Irun the chosen models, I first split the data for training and testing. 70% of the data is used for training the machine learning model so it can learn patterns from the features, and the remaining 30% is used for testing the model to check how well it performs on new, unseen data. The random\_state=42 is used to make sure the split is always the same every time you run the code, which helps in getting consistent results.

## Summary of split

|  |  |
| --- | --- |
| Term | Meaning |
| train\_test\_split() | Split data into training and testing sets. |
| test\_size=0.3 | |  | | --- | |  |  |  | | --- | | 30% data for testing, 70% for training. | |
| random\_state=42 | |  | | --- | |  |  |  | | --- | | Fixes the random shuffling for consistency. | |
| X\_train | Training the model (learning patterns) |
| X\_test | Testing the model (unseen data) |
| y\_train | |  | | --- | |  |  |  | | --- | | True labels for training set | |
| y\_test | |  | | --- | |  |  |  | | --- | | True labels for testing set | |

# Model evaluation metrics

I used evaluation metrics such as Accuracy, Precision, Recall, and F1-Score to assess model performance. Since the focus of this project was to improve the prediction of class 1 (the minority class), metrics like Recall and F1-Score were particularly important as they measure how well the model identifies positive cases and balances between precision and recall.

### Comparison with Baseline

The Dummy Classifier served as a baseline model, which simply predicted the most frequent class (class 0) for all instances. As expected, it achieved a high accuracy due to class imbalance but completely failed to predict class 1, resulting in a recall and F1-score of 0 for class 1. In comparison, the K-Nearest Neighbours (KNN) model showed a slight improvement over the baseline by correctly identifying some class 1 instances, although its performance was still limited. Both Random Forest and XGBoost significantly outperformed the Dummy and KNN models, achieving higher recall and F1-scores for class 1, which aligned with the project goal of improving minority class prediction.

# Evaluation of Performance Metrics

## KNN Performance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | 0.89 | 0.98 | 0.93 | 3124 |
| **1** | 0.75 | 0.32 | 0.45 | 575 |
| **Accuracy** |  |  | 0.88 | 3699 |
| **Macro avg** | 0.82 | 0.65 | 0.69 | 3699 |
| **Weighted avg** | 0.87 | 0.88 | 0.86 | 3699 |

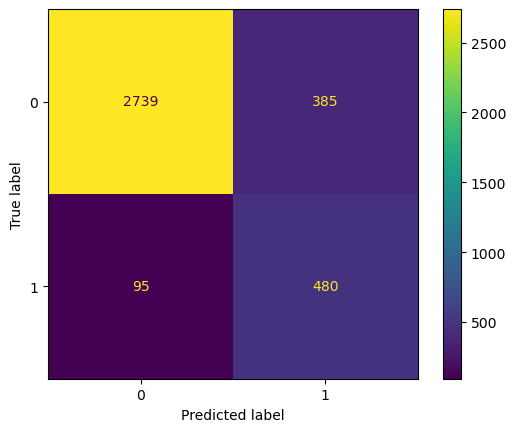
A yellow and purple squares with numbers

AI-generated content may be incorrect.

## Random Forest Performance

When I first run the Random Forest model it had high an overall accuracy of 90% but struggled to identify class 1 cases, with a low recall of 57%, meaning it missed many true positives. Having already run smote I decided to also apply class weight='balanced’ and change the decision threshold from 0.5 to 0.4 within the hyperparameters. These steps helped the model perform better for class 1, improving recall from 57% to 83% and the F1-score from 0.63 to 0.67. Although the overall accuracy dropped to 87%, this was worth it because the model became better at detecting the minority class.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | 0.97 | 0.88 | 0.92 | 3124 |
| **1** | 0.55 | 0.83 | 0.67 | 575 |
| **Accuracy** |  |  | 0.87 | 3699 |
| **Macro avg** | 0.76 | 0.86 | 0.79 | 3699 |
| **Weighted avg** | 0.90 | 0.87 | 0.88 | 3699 |



## XGboost Performance

The XGboost model achieved an overall accuracy of 86%, which looks good at first glance. However, because the dataset is imbalanced it’s important to look at other metrics too.

For class 0, the model performed very well, with a precision of 0.97 and a recall of 0.87, meaning it's good at correctly identifying non-buyers.

For class 1, which is the minority class, the model had a precision of 0.54 and a recall of 0.83. This means that while it's not always precise in predicting buyers, it's good at catching most of the actual buyers. The F1-score for class 1 is 0.65, which shows there's still room for improvement in the future.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | 0.97 | 0.87 | 0.91 | 3124 |
| **1** | 0.54 | 0.83 | 0.65 | 575 |
| **Accuracy** |  |  | 0.86 | 3699 |
| **Macro avg** | 0.75 | 0.85 | 0.78 | 3699 |
| **Weighted avg** | 0.90 | 0.86 | 0.87 | 3699 |

A chart of a blue yellow and purple color

AI-generated content may be incorrect.

## Dummy Classifier Results

Looking at the dummy classifier, it basically predicts only class 0 every time. That’s why the recall for class 0 is 1.00. It catches all the non-buyers. But for class 1 (purchase), the model does nothing, precision, recall, and F1-score are all 0.0

Compared to this, the XGboost model is clearly doing much better it actually tries to identify visitors who might purchase, with a recall of 0.83 for class 1. That’s a big improvement over the dummy, even if the precision for buyers is still not perfect.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Class** | **Precision** | **Recall** | **F1-score** | **Support** |
| **0** | 0.84 | 1.00 | 0.92 | 3124 |
| **1** | 0.00 | 0.00 | 0.00 | 575 |
| **Accuracy** |  |  | 0.84 | 3699 |
| **Macro avg** | 0.42 | 0.50 | 0.46 | 3699 |
| **Weighted avg** | 0.71 | 0.84 | 0.77 | 3699 |

A chart of a comparison of a number and a label

AI-generated content may be incorrect.

# Strengths and Weaknesses of Models

### Strengths

One clear strength of the approach used is that it doesn’t just rely on accuracy. Evaluation was done using multiple more meaningful metrics like precision, recall, and F1-score. This is very important for data like this which has significant class imbalance.

### Weakness

The KNN model struggled with precision and recall for class 1, This shows that there's still work to be done when it comes to tuning and selecting models that really balance both classes.

### Biases

I tried to handle biases by balancing out the classes and standardizing the features, which helped improve the model’s ability to treat both classes more equally. This was especially important given the imbalance in the dataset, where purchases were much less common than non-purchases. Standardizing the features also made sure that distance-based models like KNN performed more reliably, as all features were on the same scale. Overall, these steps helped reduce the risk of the model being biased toward the majority class.

# Deployment

## Deployment Strategy

To deploy the model in the real world, it could be added to a website’s backend. This way, it can study how visitors behave and predict if they might make a purchase. Businesses could use these predictions to give users personalized experiences, show special offers, or change their marketing strategies. The model can also be updated often with new data to stay useful and match how customer behaviour changes over time.

## Monitoring

As I have found out earlier accuracy isn’t the best measure for our models therefore to monitor performance after deployment, tracking key metrics like recall especially for the minority class, and F1-score over time would be important. If these metrics begin to drop, it might indicate that the data has shifted or that the model needs retraining. Logging false positives and false negatives in production could also provide insights into real-world errors the model makes.

## Future Work

In future work, it would be useful trying more advanced class balancing techniques or exploring deep learning models might help further improve performance, especially on the minority class.

Another next step could involve building a feedback loop where user actions such as purchases, or bounce rates are fed back into the model to help it continuously learn and adapt.

# Conclusion

In this project, I successfully built and evaluated predictive models to determine whether a visitor on an e-commerce website would make a purchase based on their browsing behaviour. Through data understanding and preparation, I addressed key challenges such as class imbalance using SMOTE and feature scaling for uniformity.

The models tested were Random Forest, XGBoost, and KNN. Random Forest and XGBoost models outperformed the KNN model, particularly in identifying purchasing sessions (class 1). Both Random Forest and XGBoost achieved higher recall and F1-scores, aligning with the project's objective to minimize false negatives and improve predictions of actual buyers.

The Dummy Classifier provided a useful baseline, confirming that the chosen models performed significantly better than random or naive predictions. This indicates that the models were able to learn meaningful patterns from user behaviour rather than relying on chance.

Overall, I were able to demonstrate machine learning models can effectively assist e-commerce businesses in predicting purchase intent. This can support strategies like targeted marketing, personalized offers, and website optimization aimed at increasing conversion rates.

Future improvements could involve feature engineering to capture more user-specific behaviours or using real-time session data for dynamic predictions. Additionally, testing more advanced ensemble methods or deep learning models could further enhance performance.