Chapter 1
A Comparative Methodology of Supervised Machine Learning Algorithms for Predicting Customer Churn Using Neuromarketing Techniques

Martin Muduva
Midlands State University, Zimbabwe

Thanks Hondoma
Midlands State University, Zimbabwe

Ronald Chiwariro
University of Zimbabwe, Zimbabwe

Fungai Jacqueline Kiwa
Chinhoyi University of Technology, Zimbabwe

ABSTRACT

This chapter presents an approach to using supervised machine learning and neuromarketing techniques to predict customer churn. It explores how combining neuromarketing strategies with machine learning algorithms improves churn forecast accuracy. The chapter highlights the significance of choosing the most appropriate techniques for churn prediction by contrasting several algorithms combined with various neuromarketing methodologies, such as biometric analysis and neuroimaging. It discusses the connection between customer attrition and neuromarketing, highlighting studies on customer relationship characteristics, neuroscience methods, and the role of emotions in churn prediction. Marketers can leverage machine-learning algorithms and evaluation metrics while adhering to privacy regulations, conducting algorithm testing, ensuring interpretability and practicing responsible use to create predictive models, minimize biases, and maintain trust in customer relationships.

DOI: 10.4018/979-8-3693-2165-2.ch001

1. INTRODUCTION

Predicting customer churn is a critical task for businesses seeking to retain their valuable customers. The combination of neuromarketing strategies with supervised machine learning algorithms has surfaced as a viable method to improve churn forecast accuracy in recent years. Using knowledge from psychology and neuroscience, neuromarketing offers a more profound comprehension of consumer behavior and decision-making processes. Using state-of-the-art neuromarketing strategies, this chapter provides an extensive comparative study of supervised machine learning algorithms for churn prediction. The aim of this study is to assess how well different algorithms perform in properly projecting customer attrition when combined with different neuromarketing approaches. This chapter examines various algorithms and their integration with biometric, neuroimaging, eye tracking, neurophysiological and facial expression analysis data in an attempt to offer insightful information about how well neuromarketing techniques can enhance churn prediction. Marketers may choose the best methods for customer churn prediction in their individual businesses by being aware of the advantages and disadvantages of different algorithms.

2. OVERVIEW OF CUSTOMER CHURN PREDICTION IN NEUROMARKETING

Customer churn refers to the occurrence when customers discontinue their association with a business or service provider and it holds significant importance in marketing as it affects organisations across various industries (Bhattacharyya, 2021). The connection between customer attrition and neuromarketing has been highlighted in relevant literature and discussions, although direct research specifically focused on this topic remains limited. Notably, studies such as Chaush (2022) on the examination of the impact of customer relationship characteristics on profitable lifetime duration and Kumar (2022) on analysis of customer relationship management offer valuable insights into customer churn, encompassing its definition, measurement and factors influencing customer retention. These studies emphasise the significance of comprehending client relationships and the duration of profitable interactions, which are essential considerations when implementing neuromarketing strategies.

Recent studies provide a critical analysis and outlook on “branding the brain” in the context of neuromarketing and consumer behavior (Chaush, 2022). These studies explore the use of neuroscience methods to comprehend consumer behavior, emphasising the use of cognitive processes, physiological measurements and neuroimaging to reveal subconscious reactions and guide marketing tactics. Although research on the importance of emotions in customer churn prediction do not explicitly incorporate neuromarketing approaches, they do highlight the relevance of these techniques. Chaush (2022) investigates how relationship constructs affect client referrals and the quantity of services acquired from a multiservice provider. Huang employs a machine learning approach to study how customer emotions affect churn prediction (Huang, 2020). These researches highlight the significance of emotional reactions in comprehending consumer behavior and their impact on attrition, offering information that may be investigated further through the application of neuromarketing techniques.

While there is not much direct overlap between research on customer turnover and neuromarketing, integrating knowledge from both fields provides a basis for comprehending the applicability of neuromarketing strategies in anticipating and resolving customer attrition. Additional studies spanning these domains may enhance our comprehension of the brain foundations of churn-related customer behaviors and guide the creation of more potent customer retention tactics (Chaush, 2022). Neuromarketing tech-

niques have gained significant attention in recent years for their ability to provide insights into customer behavior and aid in predicting customer churn (Duque-Hurtado, 2020). Previous researches examined brain activity and find neural responses connected to consumer behavior using neuroimaging techniques like functional magnetic resonance imaging (fMRI) (Casado-Aranda, 2022). Implementing these methods to study brand perception, emotional reactions and customer preferences has yielded insightful data for churn prediction.

Electroencephalography (EEG) and galvanic skin response (GSR) are two psychophysiological tests that offer insights into the emotional and cognitive processes that underlie consumer behavior (Val-Calvo, 2020). According to Saha (2022), these assessments aid in predicting patterns of brain activity and physiological reactions associated with consumer preferences, engagement and decision-making. These findings may be helpful in forecasting customer attrition. Analysing visual attention patterns and gaze behavior during client interactions by using eye-tracking techniques can be equally beneficial (Saha, 2022). Marketers can also forecast customer turnover by examining eye movements to learn more about what draws in customers, their visual preferences and how visual stimuli affect decision-making.

Facial expression analysis involves the process of recording and analysing facial muscle movements to infer emotional states and cognitive reactions (Xie, 2022). Based on facial expressions of contentment, dissatisfaction or apathy, this technique aids in assessing customers' emotional engagement, reactions to marketing stimuli and possibility of churn (Shadaydeh, 2021). According to Dsedsickis (2020), biometric data such as skin conductance, heart rate and face electromyography (EMG) offer physiological markers of emotional arousal and involvement. Based on physiological markers of interest, researchers can learn more about the emotional reactions, attention spans and likelihood of churn of their consumers by examining these metrics (Dzedzickis, 2020).

Incorporating neuromarketing strategies enables a deeper understanding of consumer behavior and attrition forecasting. These methods uncover subconscious reactions, affective involvement and cognitive processing, providing valuable insights for focused marketing campaigns and client retention. Combining supervised machine learning with neuromarketing insights enhances churn prediction accuracy by leveraging customer data, including demographics, purchase history and neuromarketing-derived features (Bhattacharyya, 2021). Techniques such as neuroimaging, psychophysiological measurements, eye tracking, facial expression analysis and biometric measurements facilitate a comprehensive understanding of customer behavior and emotions. Integrating neuromarketing insights and machine learning algorithms empowers marketers to develop targeted retention strategies and improve the precision of churn prediction models.

3. SUPERVISED MACHINE LEARNING ALGORITHMS IN NEUROMARKETING

Several supervised machine-learning algorithms are frequently used in neuromarketing and they can be combined with various neuromarketing strategies to forecast customer attrition. For binary classification problems, logistic regression is a popular algorithm that may be used with neurophysiological data like fNIRS signals or EEG data (Cao, 2022). Logistic Regression is a useful tool for churn prediction because it can predict patterns linked to consumer engagement, attention or emotional states by utilising neurophysiological data, which offers insights into cognitive and emotional reactions.

Using decision trees, which resemble trees and base choices on feature thresholds and splitting criteria, is an additional strategy. Eye tracking data can also be used as input features for Decision Trees in

neuromarketing (Mashrur, 2022). Decision Trees can predict significant visual stimuli or features that influence consumer behavior and anticipate churn based on gaze patterns thanks to eye-tracking data, which offers useful information about visual attention and gaze patterns.

Facial expression analysis is one of the input features that Random Forests, an ensemble learning technique that integrates many decision trees can use in neuromarketing. It is possible to deduce facial clues and emotional reactions using facial expression analysis (Aslam, 2021). By utilising these clues, Random Forests are able to generate a strong predictive model by capturing intricate correlations between facial expressions and client attrition.

Neuromarketing can make use of Support Vector Machines (SVM), a flexible method that can be applied to both regression and classification problems utilising EEG signals. Electrical brain activity is captured by EEG signals, which disclose emotional and cognitive states (Chaush, 2022). Churn prediction can be aided by SVM by employing EEG signals to predict patterns linked to consumer involvement or decision-making.

Neuromarketing can make use of the potent ensemble learning algorithms known as gradient boosting methods, including XGBoost and LightGBM, in conjunction with biometric data. Physiological measurements such as skin conductance or heart rate are examples of biometric data that shed light on physiological arousal and emotional reactions (Xie, 2022). Accurate forecasts can be obtained by combining gradient boosting methods with biometric data to capture intricate correlations between physiological parameters and churn.

Multilayer Perceptron designs, in particular, are a type of neural network that can recognise intricate patterns and correlations. Neuroimaging data, such fMRI or PET scans, can be linked into neural networks for use in neuromarketing (Val-Calvo, 2020). Neuroimaging data offers comprehensive insights into neural networks and brain activity related to cognitive functions. Neural Networks are able to forecast churn based on neural activation patterns and capture complex patterns in brain responses by combining neuroimaging data.

4. PERFORMANCE EVALUATION METRICS

When developing predictive models in neuromarketing for customer churn prediction, it is important to evaluate and assess the performance of these models. Evaluation metrics provide quantitative measurements that help determine how well a model is performing and how accurately it predicts churn. These metrics enable researchers and practitioners to compare different models and make informed decisions about the effectiveness of their predictive models in capturing customer churn. Table 1 shows the commonly used evaluation metrics in churn prediction.

These evaluation metrics provide valuable insights into the performance and effectiveness of predictive models in neuromarketing-based customer churn prediction. By considering these metrics, researchers and practitioners can make informed decisions about model selection, fine-tuning and optimisation to improve the accuracy and reliability of their predictions.

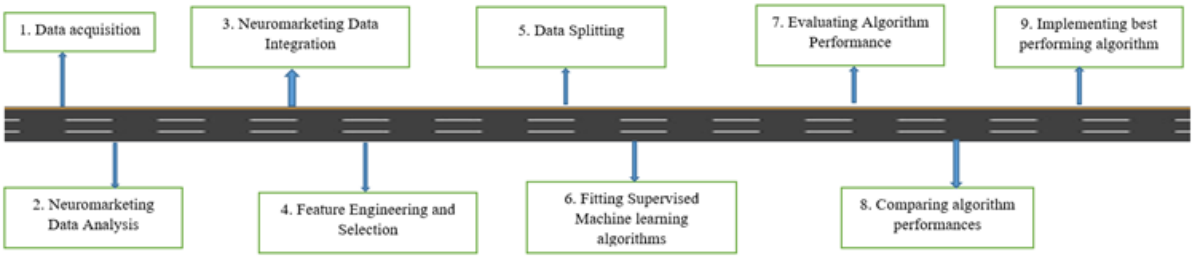
5. PROPOSED METHODOLOGY ROAD-MAP

Figure 1 depicts the proposed road map for the methodology.

Table 1. Metrics used in evaluating performance of churn prediction algorithms

Metric	Description	Formula	Importance
Accuracy	Measures the overall correctness of the model's predictions.	Accuracy = (TP + TN) / (TP + TN + FP + FN)	Important for evaluating the model's performance in predicting customer churn.
Precision score	Calculates the ratio of true positive predictions to the total number of positive predictions.	Precision = TP / (TP + FP)	Relevant when minimising false positives, which is important in predicting potential churners accurately.
Recall score	Measures the ratio of true positive predictions to the total number of actual positive instances.	Recall = TP / (TP + FN)	Useful for minimising false negatives and ensuring that actual churners are correctly identified.
F1 Score	Provides a balanced evaluation metric by combining precision and recall.	F1 Score = 2 * (Precision * Recall) / (Precision + Recall)	Offers a single score that considers both false positives and false negatives, important for evaluating model performance holistically.
Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AUC)	Evaluates the model's ability to discriminate between positive and negative instances.	TPR = True Positives / (True Positives + False Negatives) FPR = False Positives / (False Positives + True Negatives)	Useful for assessing the model's performance and selecting an appropriate classification threshold.
Confusion Matrix	Summarises the performance of a classification model, showing true/false positives and negatives.	Predicted Class Positive Negative Actual Positive TP FN Class Negative FP TN	Provides detailed analysis of the model's performance and error types, aiding in understanding and improving the model.
Time complexity	Measures the computational resources required to train and predict with a model.	- T(n) = O(f(n)) (Though it varies according to the nature of the algorithm)	Important to consider when dealing with large datasets or real-time prediction scenarios to ensure efficient model implementation.
Loss	Measures the discrepancy between predicted and actual values, quantifying how well the model fits the data.	Loss = -1/N * ∑[y * log(y_pred) + (1 - y) * log(1 - y_pred)]	Crucial for training the model and optimising its performance to accurately predict customer churn.
Space complexity	Refers to the memory or storage required by a model to store parameters and make predictions.	S(n) = O(g(n)) (Varies according to the nature of the algorithm)	Essential to consider, especially in resource-constrained environments or models with a large number of parameters.

Figure 1. Proposed methodology



The methodology begins with data acquisition, where relevant information is collected from various sources, including customer demographics, past purchase history and neuromarketing data obtained through techniques like fMRI, EEG, eye-tracking, facial expression analysis and physiological markers. The collected data is then analysed to extract insights into customer behavior, emotional responses and

engagement, allowing for a deeper understanding of customers' subconscious motivations and preferences. Feature engineering and selection are performed to transform the data into a suitable format for machine learning algorithms, selecting the most relevant features for accurate churn prediction. The dataset is divided into training and testing sets to train and evaluate the supervised machine-learning model's performance. Various algorithms, such as logistic regression, decision trees, random forests or neural networks are applied to associate features with churn labels and make predictions on new data. Model evaluation helps assess the model's performance and select the best one based on evaluation metrics. Finally, the selected model is deployed to predict customer churn, predict high-risk customers, gain insights into factors influencing churn and develop targeted retention strategies to reduce customer attrition.

6. UNPACKING THE METHODOLOGY

6.1 Data Acquisition

The data acquisition process involves gathering relevant data from various sources, including customer databases, transaction records, website analytics, CRM systems, surveys, social media platforms and neuromarketing research. Specific data attributes and variables are defined, such as customer demographics, purchase history, engagement metrics, sentiment/feedback and neuromarketing data like neuroimaging or facial expressions. The data is collected by extracting it from databases, conducting surveys, accessing APIs or analysing research findings. Data mining tools like Python with libraries such as scikit-learn or TensorFlow, SQL/NoSQL databases can be used. Deployment of sensors, such as EEG for brainwave activity or eye-tracking devices for gaze patterns is employed to capture neuro data in controlled environments. This neuro data is integrated with other customer data sources for comprehensive analysis.

6.2 Data Analysis and Pre-Processing

Data pre-processing involves preparing the raw neuromarketing data for analysis and modelling. This step typically includes the following tasks:

i. Data Cleaning: Removing any noise, outliers or artefacts from the data that may distort the results. This can involve techniques such as filtering, smoothing or interpolation. The following code snippet is a demonstration of how this can be achieved.

```
function data_cleaning(data):

cleaned_data = apply_noise_removal(data)

cleaned_data = apply_outlier_removal(cleaned_data)

cleaned_data = apply_artifact_removal(cleaned_data)

return cleaned_data
```

ii. Testing the data for normal distribution

Several statistical tests and algorithms can be used to test the data for normality. These tests provide different approaches to assessing the normality of data, taking into account various aspects such as the shape of the distribution, skewness and kurtosis. It is advisable to apply multiple tests and consider the results collectively, along with other factors, to make a comprehensive determination about the normality of the data. The pseudocode for testing the normal distribution of data is as follows:

a. Shapiro-Wilk Test

The Shapiro-Wilk test is a popular statistical test for assessing normality. It evaluates the null hypothesis that a sample is drawn from a normally distributed population. The following pseudocode demonstrates implementing Shapiro-Wilk Test:

function shapiro_wilk_test(data):

statistic, p_value = shapiro_test(data)

return statistic, p_value

b. Anderson-Darling Test:

The Anderson-Darling test is another statistical test commonly used to assess normality. It computes a test statistic based on the discrepancy between the sample data and the expected values under the null hypothesis of normality (Xie, 2022). The pseudocode is as follows:

function anderson_darling_test(data):

statistic, critical_values, significance_level = anderson_test(data)

return statistic, critical_values, significance_level

c. QQ Plot

A QQ plot (quantile-quantile plot) is a graphical method for visually assessing the normality of data. It compares the quantiles of the sample data against the quantiles of a theoretical normal distribution (Cao, 2022). The pseudocode is as follows:

function qq_plot(data):

plot = create_qq_plot(data)

return plot

d. Kolmogorov-Smirnov Test

The Kolmogorov-Smirnov test is a non-parametric statistical test used to determine if a sample follows a specific distribution, such as the normal distribution. It compares the cumulative distribution function (CDF) of the sample data with the theoretical CDF of the expected distribution (Val-Calvo, 2020). If the test statistic derived from this comparison exceeds a critical value at a chosen significance level, it suggests that the sample does not come from the specified distribution. The pseudocode is as follows:

function kolmogorov_smirnov_test(data, expected_distribution):

statistic, p_value = kolmogorov_smirnov_test(data, expected_distribution)

return statistic, p_value

e. Lilliefors Test

The Lilliefors test is a modification of the Kolmogorov-Smirnov test specifically designed for testing normality. It addresses the issue of estimating the parameters of the normal distribution from the sample data by using the empirical distribution function (Aslam, 2021). The Lilliefors test computes the test statistic by comparing the empirical distribution function with the theoretical CDF of the normal distribution. If the test statistic exceeds a critical value at a chosen significance level, it indicates that the sample does not follow a normal distribution. The pseudocode is as follows:

function lilliefors_test(data):

statistic, p_value = lilliefors_test(data)

return statistic, p_value

f. Jarque-Bera Test

The Jarque-Bera test is a statistical test used to determine if a sample has skewness and kurtosis that match a normal distribution (Aslam, 2021). It assesses whether the skewness and kurtosis of the sample data significantly deviate from the expected values of a normal distribution, which are sero for both skewness and excess kurtosis. The test statistic is calculated based on the sample skewness and kurtosis. If the test statistic exceeds a critical value at a chosen significance level, it suggests that the sample does not exhibit the characteristics of a normal distribution. The pseudocode is as follows:

function jarque_bera_test(data):

```
statistic, p_value = jarque_bera_test(data)

return statistic, p_value
```

g. D'Agostino-Pearson Test

The D'Agostino-Pearson test is another statistical test that evaluates if a sample follows a normal distribution based on its skewness and kurtosis. It computes a test statistic using skewness and kurtosis to assess whether they significantly deviate from the values expected in a normal distribution. The test statistic is asymptotically chi-squared distributed. If the test statistic exceeds a critical value at a chosen significance level, it indicates that the sample does not conform to a normal distribution. The pseudo-code is as follows:

```
function dagostino_pearson_test(data):

statistic, p_value = dagostino_pearson_test(data)

return statistic, p_value
```

When conducting tests for normality, such as the Shapiro-Wilk Anderson-Darling, Kolmogorov-Smirnov, Lilliefors, Jarque-Bera and D'Agostino-Pearson tests, the null hypothesis (H0) assumes that the data follows a normal distribution. To assess whether the data is normally distributed, the p-value resulting from each test is compared to a predetermined significance level (often set at 0.05). If the p-value is greater than the significance level, we fail to reject H0, indicating that the data can be considered normally distributed. Conversely, if the p-value is less than or equal to the significance level, we reject H0, suggesting that the data does not follow a normal distribution. However, it is important to note that no single test can definitively establish normality. It is recommended to use multiple tests and take into account other factors such as sample size and the specific characteristics of the data to make an informed judgment about normality.

iii. Data Normalisation: Scaling the data to a common range or distribution should be done to ensure that different features contribute equally to the prediction process. Common normalisation techniques include s-score normalisation or min-max scaling which can be done as follows:

```
function s_score_normalisation(data):

mean = calculate_mean(data)

std_dev = calculate_standard_deviation(data)

normalised_data = (data - mean) / std_dev
```

```
return normalised_data

function min_max_scaling(data, min_value, max_value):

min_val = calculate_minimum_value(data)

max_val = calculate_maximum_value(data)

scaled_data = ((data - min_val) / (max_val - min_val)) * (max_value - min_value) + min_value

return scaled_data
```

In s-score normalisation, also known as standardisation, the data is transformed by subtracting the mean and dividing by the standard deviation. This guarantees a mean of 0 and a standard deviation of 1 for the modified data. When using min-max scaling, the data is translated linearly into a predetermined range, usually between 0 and 1. Each data point's minimum value is subtracted and the resulting value is divided by the range (highest value minus minimum value) to complete the transformation. These methods of normalisation are frequently employed to bring the values of various variables or features to a comparable scale and avoid the dominance of some features because of their bigger magnitudes.

iv. Handling Missing Data: Dealing with any missing values in the dataset, which may involve imputation techniques such as mean imputation or regression imputation. The following pseudocode is a demonstration of how these can be done.

```
function mean_imputation(data):

mean = calculate_mean(data)

imputed_data = replace_missing_values_with_mean(data)

return imputed_data

function regression_imputation(data):

for feature in data.columns:

known_values = data[~data[feature].isnull()]

missing_values = data[data[feature].isnull()]

X_train = known_values.drop(feature, axis=1)

y_train = known_values[feature]
```

```
regression_model = train_regression_model(X_train, y_train)

imputed_values = regression_model.predict(missing_values.drop(feature, axis=1))

data.loc[data[feature].isnull(), feature] = imputed_values

return data
```

v. Data Transformation: Data transformations are important when the assumptions of the analysis technique, such as normality or homoscedasticity, necessitate modifying the data. Two common transformations are logarithmic and power transformations. Logarithmic transformation reduces the impact of large values and skewness in the data. It involves applying the natural logarithm to each element of the dataset using the np.log() function from NumPy. Power transformation adjusts the data distribution or stabilises the variance. For instance, a square root transformation (power = 0.5) can be applied using the exponentiation operator (**). The choice of transformation depends on the data characteristics and analysis requirements. These are as demonstrated in the following pseudocodes:

```
function logarithmic_transformation(data):

transformed_data = []

for element in data:

transformed_element = apply_logarithm(element)

transformed_data.append(transformed_element)

return transformed_data

function power_transformation(data, power):

transformed_data = []

for element in data:

transformed_element = apply_power_transformation(element, power)

transformed_data.append(transformed_element)

return transformed_data

function apply_logarithm(element):
```

```
transformed_element = logarithm_function(element)

return transformed_element

function apply_power_transformation(element, power):

transformed_element = power_function(element, power)

return transformed_element

function logarithm_function(element):

// Apply the natural logarithm to the element

transformed_element = log(element)

return transformed_element

function power_function(element, power):

// Apply the power transformation to the element

transformed_element = element ** power

return transformed_element
```

6.3 Integration of Neuromarketing Data

Integrating neuromarketing data involves combining different modalities or sources of data to enhance the predictive modelling process. This integration can provide a more comprehensive understanding of customer behavior. Table 2 explains typical data that can be integrated in this scenario.

This example uses three feature categories. Firstly, Multimodal Data Fusion is performed by combining data from neurophysiological sources such as EEG, eye-tracking and facial expression analysis into a dictionary named neuro_data. The combined data is stored in the dictionary combined_data and the results are printed. Next, the focus shifts to Integrating with Behavioural Data. In this case, behavioural data, including purchase history, website interactions and survey responses, is represented in the dictionary behavioural_data. This data is integrated into the dictionary integrated_data, facilitating a comprehensive understanding of customer behavior and preferences. The integrated results are printed. Lastly, Contextual Data Integration is explored. Contextual information, comprising environmental factors, marketing stimuli and demographic information, is captured in the dictionary contextual_data. By incorporating this data into the dictionary enhanced_data, predictive models are improved, leading to deeper insights into customer responses. The pseudocode is as follows:

```
function multimodal_data_fusion(neuro_data):
```

Table 2. Typical data for neuromarketing

Feature Category	Description
i. Multimodal Data Fusion	Combining data from multiple neurophysiological sources, such as EEG, eye-tracking or facial expression analysis, to capture different aspects of cognitive or emotional responses.
ii. Integrating with Behavioural Data	Integrating neuromarketing data with behavioural data, such as purchase history, website interactions or survey responses, to create a more holistic view of customer behavior and preferences.
iii. Contextual Data Integration	Incorporating contextual information, such as environmental factors, marketing stimuli or demographic information, to enhance the predictive models and provide a more nuanced understanding of customer responses.

```
combined_data = {}

# Perform multimodal data fusion by combining neurophysiological data

combined_data['neuro_data'] = neuro_data

# Print the combined data

print("Combined Data:")

print(combined_data)

return combined_data

function integrate_with_behavioural_data(combined_data, behavioural_data):

integrated_data = {}

# Integrate behavioural data with the previously combined data

integrated_data.update(combined_data)

integrated_data['behavioural_data'] = behavioural_data

# Print the integrated data

print("Integrated Data:")

print(integrated_data)

return integrated_data

function contextual_data_integration(integrated_data, contextual_data):

enhanced_data = {}
```

```
# Integrate contextual data with the previously integrated data

enhanced_data.update(integrated_data)

enhanced_data[‘contextual_data’] = contextual_data

# Print the enhanced data

print(“Enhanced Data:”)

print(enhanced_data)

return enhanced_data

# Example usage:

neuro_data = { ‘EEG’: ..., ‘eye_tracking’: ..., ‘facial_expression’: ... }

behavioural_data = { ‘purchase_history’: ..., ‘website_interactions’: ..., ‘survey_responses’: ... }

contextual_data = { ‘environmental_factors’: ..., ‘marketing_stimuli’: ..., ‘demographic_information’: ... }

# Perform multimodal data fusion

combined_data = multimodal_data_fusion(neuro_data)

# Integrate with behavioural data

integrated_data = integrate_with_behavioural_data(combined_data, behavioural_data)

# Perform contextual data integration

enhanced_data = contextual_data_integration(integrated_data, contextual_data)
```

6.4 Feature Extraction

Feature extraction involves predicting and extracting relevant features from the pre-processed neuromarketing data. This step aims to reduce the dimensionality of the data and capture the most informative aspects of the signals. Table 3 shows the typical feature extraction techniques in neuromarketing.

These following pseudocodes provides a basic structure for extracting features from different domains in neuromarketing. The actual implementation details, such as the specific functions to calculate statistical measures or apply transformations, would depend on the specific programming language and libraries in use.

Table 3. Feature extraction techniques in neuromarketing

Feature category	Description
i. Time-domain Features	Extracting statistical measures such as mean, variance or skewness from the raw signals over specific time intervals.
ii. Frequency-domain Features	Utilising Fourier or wavelet transforms to extract frequency-related information from the signals.
iii. Spectral Features	Capturing characteristics related to the power spectrum or spectral entropy of the signals.
iv. Event-related Potentials	Predicting and analysing specific components of the signals that correspond to cognitive or perceptual processes.
v. Connectivity Features	Assessing the functional or effective connectivity between different brain regions using techniques like coherence or Granger causality.

##Pseudocode for Time-domain Features

function calculateTimeDomainFeatures(signals, timeInterval):

features = []

for signal in signals:

interval = extractTimeInterval(signal, timeInterval)

mean = calculateMean(interval)

variance = calculateVariance(interval)

skewness = calculateSkewness(interval)

features.append([mean, variance, skewness])

return features

#Pseudocode for Frequency-domain Features

function calculateFrequencyDomainFeatures(signals):

features = []

for signal in signals:

fourierTransform = applyFourierTransform(signal)

waveletTransform = applyWaveletTransform(signal)

frequencyInfo = extractFrequencyInfo(fourierTransform, waveletTransform)

```
features.append(frequencyInfo)

return features

##Pseudocode for Spectral Features

function calculateSpectralFeatures(signals):

features = []

for signal in signals:

powerSpectrum = calculatePowerSpectrum(signal)

spectralEntropy = calculateSpectralEntropy(signal)

features.append([powerSpectrum, spectralEntropy])

return features

## Pseudocode for Event-related Potentials

function calculateEventRelatedPotentials(signals):

features = []

for signal in signals:

components = extractEventRelatedComponents(signal)

features.append(components)

return features

## Pseudocode for Connectivity Features

function calculateConnectivityFeatures(signals):

features = []

for signal in signals:

connectivity = calculateConnectivity(signal)

features.append(connectivity)
```

return features

A problem that can be experienced at this stage is the imbalanced state of the data. Feature imbalance occurs when the distribution of the target variable (churn vs. non-churn) is significantly skewed, with one class being dominant over the other. In this case, the majority class (e.g., non-churn) may overpower the minority class (e.g., churn) during model training, leading to biased predictions. It is therefore essential to address the imbalance so as to eliminate the potential bias that arise from such. Table 4 shows the techniques which can be used to address feature imbalances:

The following pseudocodes provides a general structure for implementing the techniques. The actual implementation details, such as the specific resampling or boosting algorithms used, would depend on the programming language and libraries of choice.

##Pseudocode for Resampling

function resampleDataset(dataset, technique):

if technique == “oversampling”:

return oversampleDataset(dataset)

else if technique == “undersampling”:

return undersampleDataset(dataset)

Table 4. Techniques for addressing feature imbalances

Technique	Description
Resampling	Involves creating a more balanced dataset by either oversampling the minority class or undersampling the majority class. ADASYN (Adaptive Synthetic Sampling), SMOTE (Synthetic Minority Over-sampling Technique) and random oversampling are examples of oversampling techniques. Tomek connections and random undersampling are two examples of undersampling techniques.
SMOTE (Synthetic Minority Over-sampling Technique)	A popular oversampling technique that generates synthetic samples for the minority class. It does this by interpolating between neighbouring instances. SMOTE randomly selects a minority class instance and finds its k nearest neighbours. Synthetic samples are created by randomly selecting one or more of the nearest neighbours and adding a fraction of the difference between the chosen neighbour and the original instance to the original instance. SMOTE helps balance the dataset and increases the representation of the minority class.
Class weights	Adjusting class weights during model training assigns higher weights to the minority class and lower weights to the majority class. This gives more importance to the minority class. Many machine learning algorithms and libraries provide options to specify class weights.
Ensemble methods	Ensemble techniques, such as boosting algorithms (e.g., AdaBoost, XGBoost), can effectively handle class imbalance. These algorithms iteratively train weak classifiers, giving more weight to misclassified instances. This improves the prediction effectiveness on the minority class.
Anomaly detection	Churn instances may be considered as anomalies. Anomaly detection algorithms can be employed to predict and address these anomalies separately, as they may exhibit different patterns compared to the majority class.
Evaluation metrics	Traditional evaluation metrics like accuracy may not be suitable for imbalanced datasets. Instead, metrics like precision, recall, F1 score and area under the ROC curve (AUC-ROC) should be used to assess model performance.

```
else:
return dataset
##Pseudocode for Synthetic Minority Over-sampling Technique (SMOTE)
function applySMOTE(minorityClass, majorityClass, k, fraction):
syntheticSamples = []
for instance in minorityClass:
neighbors = findKNearestNeighbors(instance, majorityClass, k)
for neighbor in neighbors:
syntheticSample = generateSyntheticSample(instance, neighbor, fraction)
syntheticSamples.append(syntheticSample)
return syntheticSamples
##Pseudocode for Class Weights
function adjustClassWeights(dataset, minorityClassWeight, majorityClassWeight):
for instance in dataset:
if instance.class == minorityClass:
instance.weight = minorityClassWeight
else:
instance.weight = majorityClassWeight
return dataset
##Pseudocode for Ensemble Methods
function applyBoostingAlgorithm(dataset, boostingTechnique):
return boostingTechnique.train(dataset)
##Pseudocode for Anomaly Detection
```

function applyAnomalyDetection(dataset):

anomalies = []

for instance in dataset:

if isAnomaly(instance):

anomalies.append(instance)

return anomalies

##Pseudocode for Evaluation Metrics

function calculateEvaluationMetrics(predictions, trueLabels):

accuracy = calculateAccuracy(predictions, trueLabels)

precision = calculatePrecision(predictions, trueLabels)

recall = calculateRecall(predictions, trueLabels)

f1Score = calculateF1Score(predictions, trueLabels)

aucROC = calculateAUCROC(predictions, trueLabels)

return accuracy, precision, recall, f1Score, aucROC

6.5 Splitting the Dataset Into Training and Testing Sets

The train_test_split function from the sklearn.model_selection module is used in the following code snippet to divide the data into training and testing sets. The input features and associated goal values are represented, respectively, by the variables x and y. The code indicates that 20% of the dataset should be assigned to the testing set by specifying test_sise=0.2. The model will be trained using the remaining 80%. random_state = 5 is configured to guarantee repeatability. The split will be consistent and consistent outcomes will be ensured by utilising the same random state value each time the code is executed. This is very helpful when looking for predictable results. This can be done using the following pseudocode:

Splitting the data into input features (X) and goal values (y)

X = processed_dataset[<input_features_columns>]

y = processed_dataset[<goal_values_column>]

```
# Splitting the data into training and testing sets

X_train, X_test, y_train, y_test = <train_test_split_function>(X, y, test_sise=<test_sise>, random_
state=<random_state>)

# Train the model using the training set

model = <initialise_model>()

model.fit(X_train, y_train)

# Evaluate the model using the testing set

predictions = model.predict(X_test)

accuracy = <calculate_accuracy>(y_test, predictions)
```

The train_test_split function returns four variables: x_train, x_test, y_train and y_test. The variables represent the following:

By assigning the returned values of train_test_split to these variables, you can access and use them in subsequent parts of your code. For example, you can train a machine-learning model using x_train and y_train and then evaluate the model's performance using x_test and y_test.

6.6 Fitting Supervised Machine Learning Algorithms

The procedure for fitting the classification algorithms is done in a staged sequence as follows:

- Create an instance of the classifier.
- Fit the classifier to the training data.
- Predict the churn labels for the test data using the trained classifier.
- Calculate the accuracy of the predictions by comparing them to the actual churn labels.
- Print the accuracy classification report.

The following pseudocode demonstrates how to implement the prediction algorithms:

Table 5. Variable and description

Variable	Description
x_train	Input features of the training set
x_test	Input features of the testing set
y_train	Target values of the training set
y_test	Target values of the testing set

```
# Initialise and train the classification model
model = <initialise_classification_model>()
model.fit(X_train, y_train)
# Make predictions on the testing set
y_pred = model.predict(X_test)
# Evaluate the model performance
accuracy = <calculate_accuracy>(y_test, y_pred)
precision = <calculate_precision>(y_test, y_pred)
recall = <calculate_recall>(y_test, y_pred)
f1_score = <calculate_f1_score>(y_test, y_pred)
auc_roc = <calculate_auc_roc>(y_test, y_pred)
# Print or display the evaluation metrics
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1_score)
print("AUC-ROC:", auc_roc)
```

6.7 Evaluating the Performance of the Algorithms

To evaluate the model performance, the following code snippets are used to assess the model performance using the sampled metrics:

a. a. Evaluating using classification report

```
function calculate_class_metrics(y_true, y_pred, class):
```

```
# Calculate metrics for a specific class

true_positives = count_true_positives(y_true, y_pred, class)

false_positives = count_false_positives(y_true, y_pred, class)

false_negatives = count_false_negatives(y_true, y_pred, class)

precision = calculate_precision(true_positives, false_positives)

recall = calculate_recall(true_positives, false_negatives)

f1_score = calculate_f1_score(precision, recall)

class_metrics = {

‘class’: class,

‘precision’: precision,

‘recall’: recall,

‘f1_score’: f1_score

}

return class_metrics
```

This produces a classification report with four metrics. Figure 2 is an example of a classification report produced in Python.

The evaluation of the model’s performance in predicting customer churn are that the precision score of 0.78 implies that out of all the instances predicted as churners, 78% are true churners, while 22% are false positives. This suggests that the model has a relatively good ability to accurately predict churners. On the other hand, the recall score of 0.67 indicates that the model correctly identifies 67% of all actual

Figure 2. Sample classification report

	precision	recall	f1-score	support
0	0.83	0.90	0.87	100
1	0.78	0.67	0.72	54
accuracy			0.82	154
macro avg	0.81	0.78	0.79	154
weighted avg	0.82	0.82	0.81	154

churners, while 33% of churners remain undetected as false negatives. This means that there is room for improvement in capturing all the churners in the dataset. The F1 score of 0.72 provides an overall assessment of the model's performance by considering both false positives and false negatives, with a higher score indicating better performance. The accuracy score of 0.82 signifies that the model accurately predicts 82% of the instances in the dataset, considering both churners and non-churners. The dataset consists of 100 instances of the non-churn class and 54 instances of the churn class, indicating a class imbalance. The macro average precision, recall and F1 score, which are all around 0.81, suggest a similar performance for both churn and non-churn classes. Finally, the weighted average precision, recall and F1 score, all around 0.82, represent the overall performance of the model, taking into account the class imbalance.

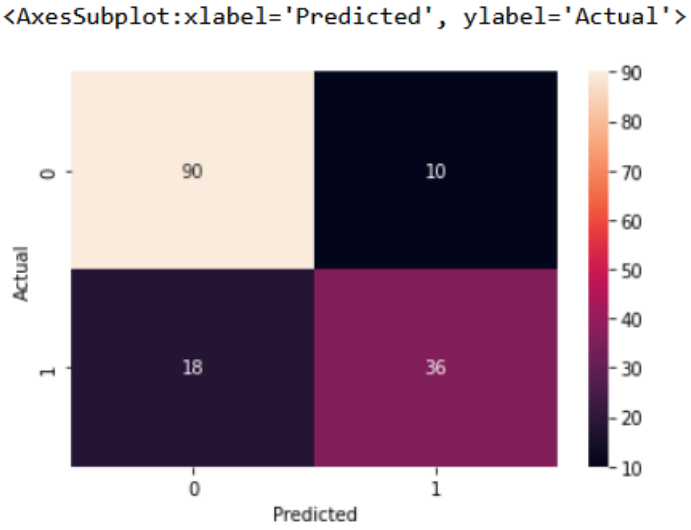
b. Evaluating using the confusion matrix

A confusion matrix is a tabular representation of the counts of true positive, true negative, false positive and false negative predictions that summarises the performance of a classification model. It gives us an evaluation of how well the model predicts both churners and non-churners, revealing both its advantages and disadvantages. A high false positive (FP) rate indicates that the model may be misclassifying non-churners as churners, which could result in costly or pointless actions. On the other hand, a high proportion of false negatives (FN) suggests that the model is not able to accurately identify churners, which means that retention efforts are being neglected. Performance metrics including accuracy, precision, recall and F1 score may be computed by analysing the values in the confusion matrix. These metrics provide a more thorough assessment of the model's predictive skills in customer churn prediction. The following pseudocode can be used to compute the confusion matrix:

```
function create_matrix(num_classes):

# Create a num_classes x num_classes matrix initialised with seros

matrix = []

for i in range(num_classes):

row = []

for j in range(num_classes):

row.append(0)

matrix.append(row)

return matrix
```

In Python, this produces a visualised matrix as shown in Figure 3.
The following explanation can be derived from this matrix:

Figure 3. Confusion matrix



- True Negative (TN): In this case, the number of true negatives is 90. This means that the model correctly predicted 90 instances as non-churners (class 0) when they were actually non-churners.
- False Positive (FP): The number of false positives is 10. This means that the model incorrectly predicted 10 instances as churners (class 1) when they were actually non-churners.
- False Negative (FN): The number of false negatives is 18. This means that the model incorrectly predicted 18 instances as non-churners when they were actually churners.
- True Positive (TP): The number of true positives is 36, indicating that the model correctly predicted 36 instances as churners.

c. Area under the curve

The Receiver Operating Characteristic (ROC) curve is a widely used metric for evaluating the performance of a model in predicting customer churn. The trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) at different categorisation thresholds is shown graphically. Plotting the true positive rate (TPR) on the y-axis versus the false positive rate (FPR) on the x-axis yields the ROC curve. The ratio of true positives to all actual positive instances (churners) is known as the TPR and the ratio of false positives to all genuine negative cases (non-churners) is known as the FPR.

function plot_roc_curve(true_positive_rates, false_positive_rates, auc):

Plot the ROC curve

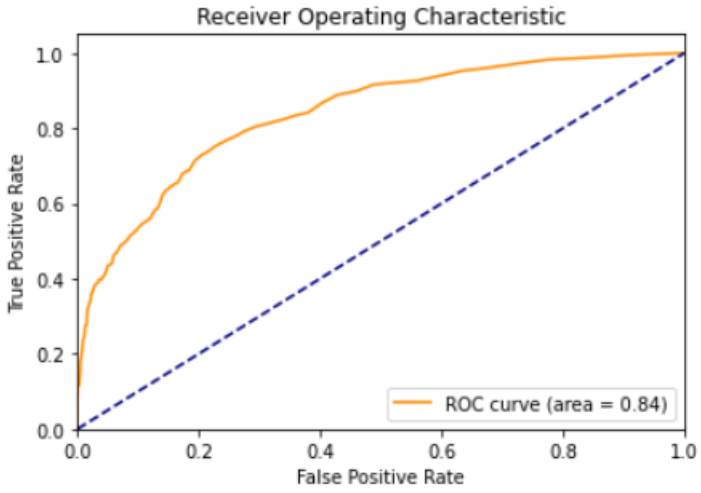
plot_curve(false_positive_rates, true_positive_rates)

Plot the diagonal line representing random classifier

```
plot_diagonal_line()

# Customise the plot

customise_plot(auc)

# Display the plot

display_plot()

function plot_curve(x_values, y_values):

# Plot the curve using x and y values

plot(x_values, y_values)

function plot_diagonal_line():

# Plot the diagonal line representing random classifier

plot([0, 1], [0, 1], 'k--')

function customise_plot(auc):

# Customise the plot with axis labels, title and legend

set_xlim([0.0, 1.0])

set_ylim([0.0, 1.05])

set_xlabel('False Positive Rate (1 - Specificity)')

set_ylabel('True Positive Rate (Sensitivity)')

set_title('Receiver Operating Characteristic (ROC) Curve')

set_legend('ROC curve (AUC = ' + str(auc) + ')')

function display_plot():

# Display the plot

show()
```

In Python, this produces a curve as depicted in Figure 4.

Figure 4. ROC curve visualisation



A model that exhibits high TPR and low FPR will have a curve that is in the top-left corner of the plot, signifying its exceptional prediction ability. A model with poor predictive performance, on the other hand, will have a curve that approximates the diagonal line, which would indicate a wild guess. The area under the ROC curve or AUC-ROC, is frequently employed as a summary indicator to evaluate the model's overall effectiveness. A random classifier is represented by a value of 0.5 on the AUC-ROC scale, while a perfect classifier is represented by a value of 1. A larger AUC-ROC often indicates a better model performance in differentiating between non-churners and churners.

6.8 Algorithm Performance Comparison

Comparing the performance of the algorithms can be done by visualising the respective metrics and deducing the best performing algorithm form the differences. Additionally, statistical tests like paired t-tests or ANOVA can be employed to determine if there are statistically significant differences in performance. Considering factors beyond evaluation metrics, such as model interpretability, required computational resources and scalability, is important for selecting the best-performing algorithm. To ensure the robustness of the findings, the evaluation process should be repeated multiple times using different random data splits. Additionally, validating the performance of the selected algorithm(s) on new, unseen data helps assess its generalisability.

6.9 Deployment

The selected algorithm can be deployed by integrating it into workflows or systems that already exist like CRM systems or APIs. To guarantee the deployed model's continued efficiency, it should be regularly inspected and maintained. This entails routinely retraining and updating the model in order to account for shifts in consumer behavior and market dynamics as new data become available. In order to use the churn prediction model's predictions to maximise customer retention efforts, it is imperative to establish a feedback loop between the model and company operations. The company can continuously enhance

its client retention efforts by assessing how churn projections affect business outcomes and adjusting measures accordingly.

7. IMPLICATIONS TO THEORY AND PRACTICE

Integrating neuromarketing techniques with supervised machine learning algorithms allows for a deeper understanding of customer behavior and the development of precise churn prediction models. This empowers businesses to proactively implement personalised retention strategies, leading to improved customer satisfaction, loyalty and profitability. From a theoretical standpoint, this approach bridges the gap between customer churn research and neuromarketing, enhances predictive accuracy and expands the understanding of emotional and cognitive factors in customer decision-making. It contributes to the advancement of knowledge in both neuromarketing and customer churn research.

8. ETHICAL CONSIDERATIONS

The application of supervised machine learning algorithms for predicting customer churn using neuromarketing techniques brings forth several ethical considerations. The collection and use of consumer data for predictive modelling should adhere to strict privacy regulations and guidelines. It is crucial to obtain informed consent from individuals whose data is being utilised, ensuring transparency regarding the purpose and scope of data collection. Additionally, the algorithms employed should be subjected to rigorous testing and validation to minimise biases and ensure fairness in decision-making. Attention should be given to potential discriminatory outcomes and the inadvertent reinforcement of existing biases. Interpretability of the chosen algorithms is vital to provide explanations for the predictions, enabling individuals to understand and challenge the decisions made based on their personal data. It is also important to consider the long-term implications of these predictive models on customer relationships and the potential impact on individuals' autonomy and freedom of choice. Ensuring responsible and ethical use of these algorithms is essential to maintain trust, protect consumer rights and foster a positive and equitable customer experience.

9. FUTURE STUDIES

Future studies in this domain could explore longitudinal analysis to assess the long-term effectiveness of neuromarketing techniques and machine learning algorithms in predicting customer churn. The integration of multiple data sources, such as social media analytics and customer surveys, alongside neuromarketing data, could be investigated to enhance the accuracy of churn prediction models. Cross-industry studies would provide insights into industry-specific patterns and challenges. Ethical considerations regarding privacy and the responsible use of biometric and neuroimaging data should be addressed. Exploring advanced machine learning techniques, real-time churn prediction systems and conducting comparative studies to evaluate different approaches are avenues for further research. Additionally, investigating methods to enhance model interpretability and explainability would facilitate their adoption and application in real-world business settings. These future studies would contribute to a deeper understanding of the

integration of neuromarketing and supervised machine learning for churn prediction, leading to more robust and actionable insights for businesses.

10. CHAPTER SUMMARY

This chapter explores the combination of neuromarketing strategies and supervised machine learning algorithms for accurate customer churn prediction. It emphasises the importance of churn prediction and how neuromarketing techniques can enhance accuracy by understanding emotional reactions, cognitive processes and attention patterns. Various neuromarketing techniques such as neuroimaging, psychophysiological measurements, eye tracking, facial expression analysis and biometric measurements are discussed. The integration of supervised machine learning algorithms with these insights is examined, including examples of algorithms like logistic regression, decision trees, random forests, support vector machines, gradient boosting and neural networks. The chapter emphasises the significance of evaluating model performance using metrics and highlights the potential of understanding consumer behavior through neuromarketing and machine learning for precise churn prediction. Further research and evaluation in this field are encouraged.

REFERENCES

Aslam, M., Sherwani, R. A. K., & Saleem, M. (2021). Vague data analysis using neutrosophic Jarque–Bera test. *PLoS One*, *16*(12), e0260689. doi:10.1371/journal.pone.0260689 PMID:34855840

Bhattacharyya, J., & Dash, M. K. (2021). Investigation of customer churn insights and intelligence from social media: A netnographic research. *Online Information Review*, *45*(1), 174–206. doi:10.1108/OIR-02-2020-0048

Cao, J., Garro, E. M., & Zhao, Y. (2022). EEG/FNIRS based workload classification using functional brain connectivity and machine learning. *Sensors (Basel)*, *22*(19), 7623. doi:10.3390/s22197623 PMID:36236725

Casado-Aranda, L., Sánchez-Fernández, J., & Viedma-del-Jesús, M. I. (2022). Neural responses to hedonic and utilitarian banner ads: An fMRI study. *Journal of Interactive Marketing*, *57*(2), 296–322. doi:10.1177/10949968221087259

Chaush, A. (2022). *Aspects of eCRM and firmographics that influence customer acquisition probability*. [Master's thesis, University of Twente].

Dsedsickis, A. (2020). Human emotion recognition: Review of sensors and methods. *Sensors (Basel)*, *20*(3), 592. doi:10.3390/s20030592 PMID:31973140

Duque-Hurtado, P. (2020). Neuromarketing: Its current status and research perspectives. *Estudios gerenciales, 36(157)*, 525-539.

Mashrur, F., Rahman, K. M., Miya, M. T. I., Vaidyanathan, R., Anwar, S. F., Sarker, F., & Mamun, K. A. (2022). An intelligent neuromarketing system for predicting consumers' future choice from electroencephalography signals. *Physiology & Behavior*, *253*, 253. doi:10.1016/j.physbeh.2022.113847 PMID:35594931

Shadaydeh, M., Muller, L., Schneider, D., Thummel, M., Kessler, T., & Denzler, J. (2021). Analysing the direction of emotional influence in nonverbal dyadic communication: A facial-expression study. *IEEE Access : Practical Innovations, Open Solutions*, *9*, 73780–73790. doi:10.1109/ACCESS.2021.3078195

Val-Calvo, M. (2020). Real-time multi-modal estimation of dynamically evoked emotions using EEG, heart rate and galvanic skin response. *nternational Journal of Neural Systems, 30(04), 2050013*.

Xie, H. (2022). An overview of facial micro-expression analysis: Data, methodology and challenge. *IEEE Transactions on Affective Computing*.

Zaha, S. (2022). Chirplet transform-based machine-learning approach towards classification of cognitive state change using galvanic skin response and photoplethysmography signals. *Expert Systems: International Journal of Knowledge Engineering and Neural Networks*, *39*(6).