The Netherlands Press

Journal of Emerging Strategies in New Economics

Article

PREDICTIONS OF CUSTOMER BEHAVIOUR OVER ECOMMERCE WEBSITES AND ANTICIPATING THEIR INTENTION

Novokhatska Anastasiia

PhD, Donetsk Institute for Physics and Engineering named after O.O. Galkin of National Academy of Sciences of Ukraine
Kyiv, Ukraine

Orchid Id: https://orcid.org/0000-0002-2558-2866

E-mail: a.novokhatska@gmail.com

Abstract.

This paper presents a real-time behavioural analytics solution for online consumers that consists of two modules that estimate visitor purchase intent and website desertion probability at the same time. The first module predicts a visitor's purchase intent using aggregated page view statistics acquired during a visit as well as certain session and user information. The collected features are fed into classifiers such as random forest (RF), support vector machine (SVM), and multilayer perceptron (MLP). To increase classifier performance and scalability, use oversampling and feature selection pre-processing processes. The findings reveal that MLP calculated with a robust backpropagation method with weight backtracking outperforms RF and SVM in terms of accuracy and F1 score. Another discovery is that, although clickstream data gathered from online navigation patterns transmit vital information about visitor buy intent, session information bases include unique information about purchasing inclinations. In the second module, everyone estimate the likelihood of a visitor's desire to exit the website without completing a purchase using just sequential clickstream data. Everyone trains a long short-term memory-based recurrent neural network to provide a sigmoidal output indicating the predicted horizon. When used together, the modules detect visitors who are ready to make a purchase but are likely to depart the site within the forecast time and take the necessary steps to enhance website abandonment and buy conversion rates to do. Our findings indicate the viability of employing clickstream and session information data to forecast purchase intent in virtual retail environments in an accurate and scalable manner.

Keywords: Multi-layer Perceptron, support vector machines, XG Boost, Logistic regression.

Journal of Emerging Strategies in New Economics Volume 1 Issue 1

Received Date: 07 July 2022
Accepted Date: 14 August 2022
Published Date: 07 September 2022

1. Introduction

Most enterprises today are funded or done online. One of the most effective techniques he may use to promote a developed virtual world is marketing offers. Previously, these offers were shown to all visitors to a specific e-commerce site at random. Recognizing the need to improve marketing targeting, an online shop started to study visitor data in near real time. The idea is to contact the most relevant users (by phone, email, and so on) and provide offers that will entice them to return to the site and complete a successful purchase[2].

Recently, a new trend in the virtual buying environment has evolved in which potential visitors are detected while they surf your website. The advantage over the near-real-time approach is that you avoid the significant risk of losing users when your online store's connection is disrupted. In reality, such a strategy is modelled after an experienced salesman who is battling to keep potential visitors by delivering a series of targeted marketing promos that are likely to lead them to purchase. To do. The most recent study suggesting such a method for e-commerce websites is detailed in where the authors offer a two-module system that predicts visitor purchase intentions using various session and user information. It is advised that he visit the consolidated page to examine his data. During your stay, you will be tracked. The first module of this system determines whether or not to offer material to the user, and the second module is activated only when the user is likely to leave the site. The suggested system in [1] is appealing in terms of efficiency and scalability, but the related termination risk is not insignificant.

The rising usage of e-commerce in recent years has created market potential, but the fact that conversion rates have not grown at the same rate highlights the need for solutions that provide targeted promotions to online customers. connects Sellers at brick-and-mortar establishments may provide consumers with a selection of personalised options based on years of expertise. This experience has a considerable influence on productivity, buy conversion rates, and sales .Many e-commerce and information technology firms have invested in early detection and behaviour prediction algorithms that replicate salespeople's conduct in virtual buying environments[8] .Parallel to these attempts, other scientific studies have been suggested that employ machine learning approaches to approach the topic from various angles. Some of these studies classify visits based on user navigation patterns, while others anticipate user behaviour and cart abandonment in real time. Some want to improve by taking suitable measures[4].

This article suggests a method for real-time online behavioural analysis of purchasers. The suggested system consists of just two modules, which, to the best of our knowledge, anticipate a visitor's purchase intent and likelihood of departing a website at the same time for the first time[5]. The first module, which assigns a real-time score to the visitor's purchase intent throughout the session, is only activated if the second module, which forecasts the possibility of site desertion, provides a number greater than a predetermined threshold[7]. Sir assess the performance of several machine learning algorithms under different scenarios using data from online shops[6].

2. Dataset preparation:

This data collection contains 12,330 client sessions (rows), ten numerical variables, and eight category variables (columns). As the target variable, use the variable "Revenue" (True or False). My predictors, consumer qualities, or traits are the other 17 factors. Before modelling, create more characteristics. This information was compiled over the course of a year, without regard for vacations, special days, or unique promotions. The dataset size is adequate for the data science activities I intend to perform. Administrative Duration, Informational Duration, ProductRelated,

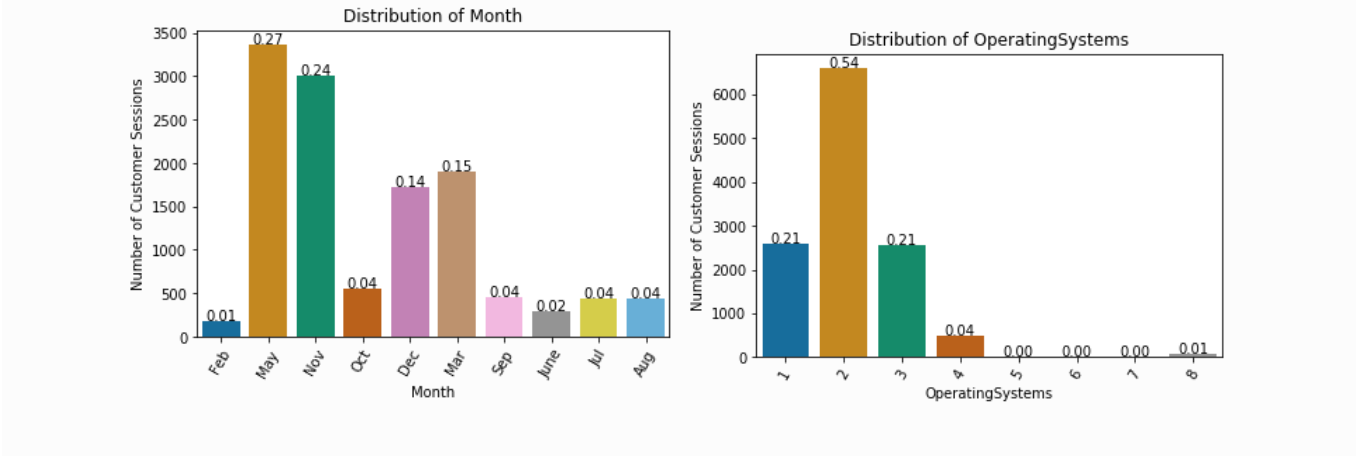
ProductRelated Duration, BounceRates, ExitRates, PageValues, SpecialDay, Month, OperatingSystems, Browser, Region, TrafficType, VisitorType, Weekend, Revenue are the columns.

	Administrative	Administrative_Duration	Informational	Informational_Duration	ProductRelated	ProductRelated_Duration	BounceRates	ExitRates	PageValues	SpecialDay
0	1	0.0	0	0.0	0	0.0	0.2	0.2	0.0	0.0
1	2	2.0	0	0.0	0	0.0	0.1	0.1	0.0	0.0
2	0	0.0	1	0.0	0	0.0	0.2	0.2	0.0	1.0
3	1	0.0	0	0.0	0	0.0	0.2	0.2	0.0	0.0
4	1	0.0	1	0.0	0	0.0	0.2	0.2	0.0	0.0

Month	OperatingSystems	Browser	Region	TrafficType	VisitorType	Weekend	Revenue
Feb	1	2	1	5	Returning_Visitor	True	False
May	1	1	3	2	Returning_Visitor	True	False
May	1	1	3	3	Returning_Visitor	True	False
May	1	1	1	2	Returning_Visitor	False	False
May	1	1	6	2	Returning_Visitor	False	False

```
Administrative             12330 non-null int64
Administrative_Duration    12330 non-null float64
Informational              12330 non-null int64
Informational_Duration     12330 non-null float64
ProductRelated             12330 non-null int64
ProductRelated_Duration    12330 non-null float64
BounceRates                12330 non-null float64
ExitRates                  12330 non-null float64
PageValues                 12330 non-null float64
SpecialDay                 12330 non-null float64
Month                      12330 non-null object
OperatingSystems           12330 non-null int64
Browser                    12330 non-null int64
Region                     12330 non-null int64
TrafficType                12330 non-null int64
VisitorType                12330 non-null object
Weekend                    12330 non-null bool
Revenue                    12330 non-null bool
dtypes: bool(2), float64(7), int64(7), object(2)
```

Currently, exploratory data analysis use X as the data columns and Y as counts. The graphic below is an exploratory data analysis of the columns in the data. When looking at the monthly distribution, one month may have more customer sessions, and the operating system 2 distribution may have more customer sessions. The browser 2 distribution, on the other hand, has a greater number of customer sessions, and the regional distribution has a higher number of region 1 sessions.



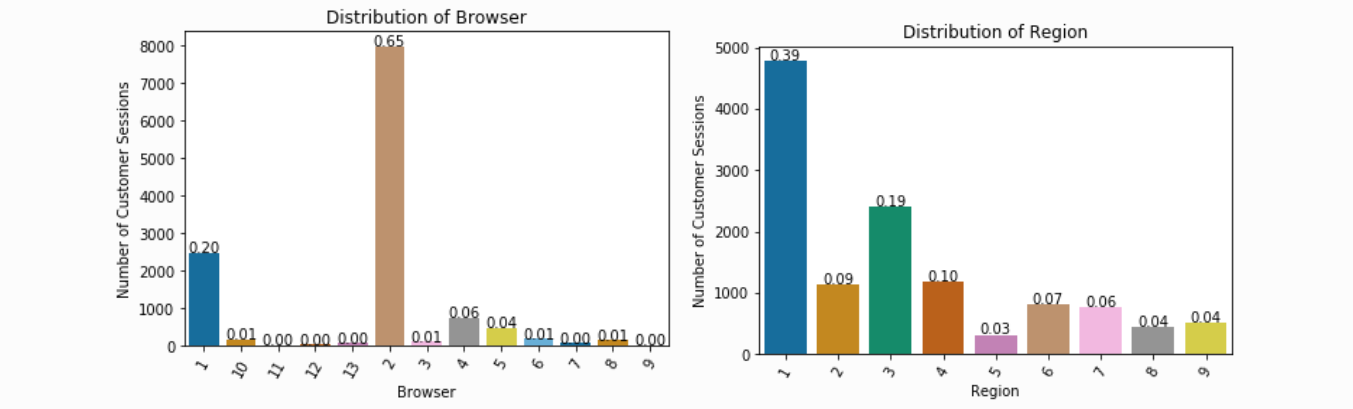


Figure 1: Distributions of various columns

Unless they purchase the item, this violin chart displays consumers who purchase things off-site. This demonstrates that customers who buy products usually know exactly what they want, so they don't need to spend as much time shopping on your site. Customers that are unsure of where they are and browse to learn more about themselves spend more time on your site but do not purchase. Sir may think of techniques to assist customers in making purchase decisions.

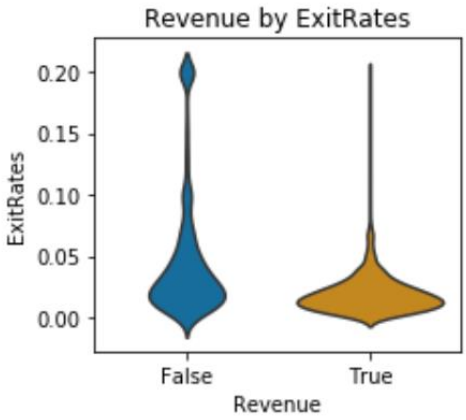


Figure 2: Revenue by Exit rates

2.1. Outliers detection:

There are other outlier identification methods, however Tukey's approach is employed to discover outliers in this case. According to Tukey's criterion, an outlier is any result that is more than 1.5 times the interquartile range apart from the interquartile. They'll come in handy when sir write a function that detects outliers using Tukey's rule. Tukey is the name of the function.

```
# Calculate number of outliers and its percentage with Tukey's method.
for i, col in enumerate(['Administrative', 'Administrative_Duration', 'Informational', 'Informational_Duration',
       'ProductRelated_Duration', 'BounceRates', 'PageValues', 'SpecialDay'], start=1):
    q75, q25 = np.percentile(new_num_df[col], [75, 25])
    iqr = q75 - q25

    min_val = q25 - (iqr*1.5)
    max_val = q75 + (iqr*1.5)
    print("Number of outliers and percentage of it in {} : {} and {}".format(col,
                                                                             len((np.where((new_num_df[col] > max_val) |
                                                                                           (new_num_df[col] < min_val))[0])),
                                                                             len((np.where((new_num_df[col] > max_val) |
                                                                                           (new_num_df[col] < min_val))[0]))*100/12330))
```

2.2. Data pre-processing

For starters, the X&Y columns are specified as, the features as X, and the labels as Y. By deleting the revenue column from feature columns, all remaining columns become features and the revenue column becomes a label. The data is then divided into training and testing groups. Because it provides a method named train test split, the scikit'learn package is used in this operation. And the test size is calculated as 20% of the data, with the remaining 80% serving as training data. Creating a pipe ln now transforms all columns to numbers, employs the hot encoder, and converts all columns to standard scalars.

```
# Define X and y.
y = online_df['Revenue']
X = online_df.drop(['Revenue'], axis=1)
```

```
# Splitting the data into train (80%) and test (20%) data.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=2, stratify=y)
```

```
# Loading libraries for preprocessing data and creating pipelines.
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
```

```
# Create pipelines to transform numeric and categorical variables.
numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])
```

```
# Split data into numeric and categorical variables.
numeric_features = X_train.select_dtypes(include=['int64', 'float64']).columns
categorical_features = X_train.select_dtypes(include=['object']).columns
```

```
# Apply transformations to the correct columns.
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])
```

3. MODEL SELECTION:

Logistic regression is one of the well-known classifiers in machine learning challenges. Logistic regression calculates the likelihood of an event, such as whether they are elected, based on a collection of independent factors. Because the outcome is a probability, the dependent variable is limited to values between 0 and 1. SVC is a C support vector taxonomy that is implemented using libsvm. Scikit-learn makes use of the sklearn module. svm. SVC. DecisionTreeClassifier is another common classifier. A decision tree is a tree structure that looks like a flowchart, with inner nodes representing characteristics (or attributes), branches representing decision rules, and each leaf node representing a conclusion. The root node is the top node in a decision tree. Learn how to divide data depending on attribute values. XGBoost, unlike many other algorithms, is an ensemble learning algorithm. To put it another way, it integrates the outcomes of several models, known as base learners, to produce predictions. XGBoost, like random forests, use decision trees as base learners. The author created the images. Rain prediction using a decision tree. Light GBM (short for Light Gradient Boosting Machine) is a Microsoft-developed free and open source distributed gradient boosting machine learning system. It uses decision tree algorithms to do ranking, classification, and other machine learning tasks. MLPClassifier is an abbreviation for Multi-Layer Perceptron Classifier, which is related to neural networks.

Unlike other classification methods such as Support Vector and Naive Bayes Classifier, MLPClassifier does classification using an underlying neural network.

```
# Create a list classifiers to use pipeline in model selection.
classifiers = [
    LogisticRegression(),
    SVC(kernel='rbf', C=0.025, probability=True),
    DecisionTreeClassifier(),
    RandomForestClassifier(),
    GradientBoostingClassifier(),
    XGBClassifier(),
    LGBMClassifier(),
    MLPClassifier()
    ]
```

```
# Create a pipeline for the various classifiers to use for model selection.
# https://www.codementor.io/@bruce3557/beautiful-machine-learning-pipeline-with-scikit-learn-uiqapbxuj
for classifier in classifiers:
    pipe = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', classifier)
                           ])
    pipe.fit(X_train, y_train)
    print(classifier)
    print('Model score: %.3f' % pipe.score(X_test, y_test))

# Display classification report for all the models for model selection.
    y_pred = pipe.predict(X_test)
    results = confusion_matrix(y_test, y_pred)
    print('Confusion Matrix: ')
    print(results)
    print('Classification Report: ')
    print(classification_report(y_test, y_pred))
```

Classification is accomplished by passing or matching X and Y to the classifier and predicting 0 or 1. This process is served in FlaskApp, which uses the Flask web framework to send the results to the front end. This may be accomplished by preserving the model together with its weights and biases.

```
# Create a simple flask app.
from flask import Flask, request
from flask_restful import Resource, Api
from sklearn.externals import joblib

app = Flask(__name__)
api = Api(app)

class Test_index(Resource):
    def post(self):
        loaded_model = joblib.load('./model/gbc_model')
        test_data = request.get_json()
        input_df = pd.DataFrame([test_data])
        input_df.rename(columns=
                        {'input_TrafficType':'TrafficType', 'input_VistorType':'VistorType', 'input_Weekend':'Weekend'}, inplace=True)
        print(input_df)
        y_train_predict = loaded_model.predict(input_df)
        test_output = pd.DataFrame(y_train_predict, columns={'output'})
        output = test_output.to_dict(orient='list')
        return output

api.add_resource(Test_index, '/test')
if __name__ == '__main__':
    app.run(debug=True)
```

4. Results & Discussions:

Outliers are found using boxplots for outlier detection. The graph below is a boxplot of all columns in the data. The data contains outliers that impact accuracy, as shown by the box slot. As a result, by eliminating data rows, these outliers may be deleted.

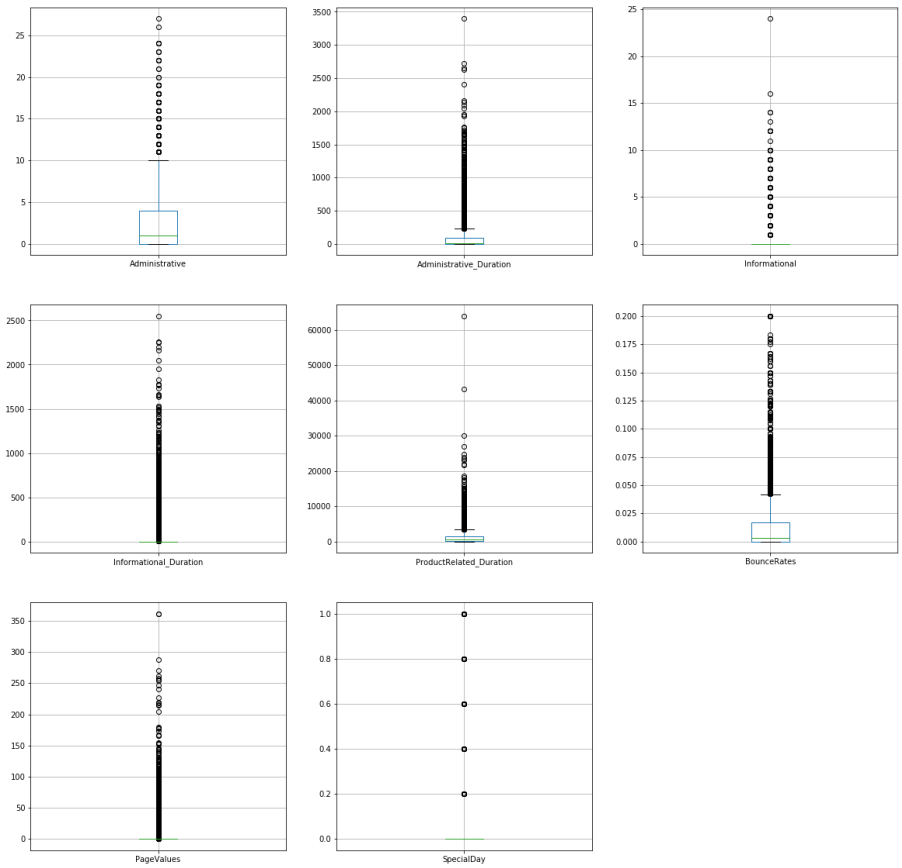


Figure 3: Boxplot for Outliers

The classification report may be used to assess the performance of the logistic regression model, and the accuracy attained by fitting the data to the model is 88.8%. Accuracy, recall, and f1 values are also critical in assessing model performance. The categorization report includes values.

Classification Report:

	precision	recall	f1-score	support
False	0.90	0.98	0.94	2084
True	0.77	0.40	0.53	382
accuracy			0.89	2466
macro avg	0.84	0.69	0.73	2466
weighted avg	0.88	0.89	0.87	2466

The classification report may be used to assess the performance of the SVC model, and the accuracy attained by fitting the data to the model is 87.9%. Accuracy, recall, and f1 values are also critical in assessing model performance. The categorization report includes values.

Classification Report:

	precision	recall	f1-score	support
False	0.89	0.98	0.93	2084
True	0.78	0.30	0.44	382
accuracy			0.88	2466
macro avg	0.83	0.64	0.69	2466
weighted avg	0.87	0.88	0.86	2466

The classification report may be used to assess the performance of the Decisiontreeclassifier model, and the accuracy attained by fitting the data to the model is 85.6%. Accuracy, recall, and f1 values are also critical in assessing model performance. The categorization report includes values.

Classification Report:

	precision	recall	f1-score	support
False	0.92	0.91	0.91	2084
True	0.54	0.56	0.55	382
accuracy			0.86	2466
macro avg	0.73	0.74	0.73	2466
weighted avg	0.86	0.86	0.86	2466

The classification report may be used to assess the performance of the Randomforestclassifier model, and the accuracy attained by fitting the data to the model is 90.0%. Accuracy, recall, and f1 values are also critical in assessing model performance. The categorization report includes values.

Classification Report:

	precision	recall	f1-score	support
False	0.92	0.96	0.94	2084
True	0.74	0.55	0.63	382
accuracy			0.90	2466
macro avg	0.83	0.76	0.79	2466
weighted avg	0.89	0.90	0.89	2466

The classification report may be used to assess the performance of the Gradient boosing classifier model, and the accuracy attained by fitting the data to the model is 90.6%. Accuracy, recall, and f1 values are also critical in assessing model performance. The categorization report includes values.

Classification Report:

	precision	recall	f1-score	support
False	0.93	0.96	0.94	2084
True	0.73	0.63	0.67	382
accuracy			0.91	2466
macro avg	0.83	0.79	0.81	2466
weighted avg	0.90	0.91	0.90	2466

The performance of the XGBclassifier model can be assessed by utilising the classification report, and the accuracy produced by fitting the data into the model is 89.7 percent. The precision, recall, and f1-score also play important roles in defining the model performance. The categorization report includes the values.

Classification Report:

	precision	recall	f1-score	support
False	0.93	0.95	0.94	2084
True	0.70	0.59	0.64	382
accuracy			0.90	2466
macro avg	0.81	0.77	0.79	2466
weighted avg	0.89	0.90	0.89	2466

The classification report may be used to assess the performance of the LGBMC classifier model, and the accuracy attained by fitting the data to the model is 90.3%. Accuracy, recall, and f1 values are also critical in assessing model performance. The categorization report include values.

Classification Report:

	precision	recall	f1-score	support
False	0.93	0.95	0.94	2084
True	0.72	0.62	0.66	382
accuracy			0.90	2466
macro avg	0.82	0.79	0.80	2466
weighted avg	0.90	0.90	0.90	2466

The performance of the MLPClassifier model can be determined using the classification report, and the accuracy attained by fitting the data to the model is 88.5%. Accuracy, recall, and f1 values are also critical in assessing model performance. The categorization report includes values.

Classification Report:

	precision	recall	f1-score	support
False	0.92	0.95	0.93	2084
True	0.65	0.55	0.60	382
accuracy			0.89	2466
macro avg	0.79	0.75	0.77	2466
weighted avg	0.88	0.89	0.88	2466

5. Conclusion:

The appeal of the proposed work is related to the need to set up a model that forecasts the visitor's shopping intent as soon as the e-commerce website is visited. The advantage behind that is to avoid the risk of abandon implied by each visit on the website. In this scope, the challenging issue consisted in finding out the most appropriate machine learning which could reach this purpose. Some classification techniques have been investigated to resolve the addressed problem, logistic regression, and random forest etc. Moreover, oversampling has been carried out to improve the performance

and the scalability of each classifier. Based on experimentation and comparison results and have proven the efficiency of the LGBM Classifier as a balanced classifier which is able to fit the requirements of our problem.

References

[1] Sakar, C.O., Polat, S.O., Katircioglu, M., Kastro, Y.: Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. Neural Comput. Appl. 31(10), 6893–6908 (2019).

[2] Mobasher, B., Dai, H., Luo, T., Nakagawa, M.: Discovery and evaluation of aggregate usage profiles for web personalization. Data Min. Knowl. Discov. 6(1), 61–82 (2002).

[3] Moe, W.W.: Buying, searching, or browsing: differentiating between online shoppers using in-store navigational clickstream. J. Consum. Psychol. 13(1–2), 29–39 (2003)

[4] Poggi, N., Moreno, T., Berral, J.L., Gavaldà, R., Torres, J.: Web customer modeling for automated session prioritization on high traffic sites. In: Conati, C., McCoy, K., Paliouras, G. (eds.) UM 2007. LNCS (LNAI), vol. 4511, pp. 450–454. Springer, Heidelberg (2007).

[5] Suchacka, G., Skolimowska-Kulig, M., Potempa, A.: Classification of e-customer sessions based on support vector machine. ECMS 15, 594–600 (2015)

[6] Suchacka, G., Skolimowska-Kulig, M., Potempa, A.: A k-nearest neighbors method for classifying user sessions in e-commerce scenario. J. Telecommun. Inf. Technol. 3(64), 64–69 (2015)

[7] Suchacka, G., Chodak, G.: Using association rules to assess purchase probability in online stores. Inf. Syst. e-Bus. Manag. 15(3), 751–780 (2016).

[8] Budnikas, G.: Computerised recommendations on e-transaction finalisation by means of machine learning. Stat. Transit. 16(2), 309–322 (2015)