Machine Learning Systems for Analyzing and Predicting User Behavior

Yuri Kravchenko[1,†], Olga Leshchenko[1,*,†], Andriy Dudnik[1,2,†], Nataliia Dakhno[1,†], and Hennadii Dakhno[1,†]

[1] *Taras Shevchenko National University of Kyiv, 60, Volodymyrska Str., Kyiv, 01033, Ukraine*
[2] *Interregional Academy of Personnel Managemen, 2 Frometivska, Str, Kyiv, 03039, Ukraine*

Abstract

In the article are considered machine learning systems designed for analysis and forecasting behavior users, with an emphasis on methods that effectively modeled complex patterns interaction users. using various algorithms such as the k - nearest method neighbors, method of support vectors, logistic regression, decision trees and random forests, in research are evaluated precision, accuracy, completeness and F - measure different approaches to forecasting behavior users. It is compared productivity these methods, emphasizing the potential of each to provide accurate and reliable predictions in different contexts. In the article also are considered challenges related to the selection and adjustment of machine learning models, including the previous one processing data, choice signs and optimization hyperparameters. The results of the study demonstrate the importance of selecting the right models for specific behavioral analysis tasks, providing valuable recommendations on optimal approaches to predicting individual behavior. This allows for improved user engagement strategies, increased participation, and enhanced decision-making support across various industries.

Keywords

machine learning, behavior prediction, SVM, KNN, decision tree, random forest, logistic regression, deep neural networks, information system

1. Introduction

Modern information systems are becoming increasingly complex and integrated into the lives of users, which creates new opportunities for analyzing their behavior. Machine learning systems open up powerful tools for developers to analyze large volumes of data and predict user behavior in various areas. Thanks to such technologies, it is possible to detect hidden patterns, predict user needs and make informed decisions about improving interaction with them. Machine learning makes it possible to create adaptive systems capable of predicting user actions based on their previous activity, thereby increasing the efficiency of interaction and user experience.

However, the development of such an information system is associated with many challenges. In particular, it is important to choose the right machine learning methods that are best suited for analyzing customer behavior. Such methods include logistic regression, decision trees, support vector machines, neural networks, and others. Each of these methods has its advantages and disadvantages, which must be taken into account when developing the system. In addition, effective use of these methods requires careful data preprocessing, feature selection, and hyperparameter tuning.

Information Technology and Implementation (IT&I-2024), November 20-21, 2024, Kyiv, Ukraine
* Corresponding author.
† These authors contributed equally.
kr34@ukr.net (Y. Kravchenko); olga.leshchenko@knu.ua (O. Leshchenko); andrii.dudnik@knu.ua (A.Dudnik); gennadiy.dakhno@gmail.com (H.Dakhno); nataly.dakhno@ukr.net (N.Dakhno)
0000-0002-0281-4396 (Y. Kravchenko); 0000-0002-3997-2785 (O. Leshchenko); 0000-0003-1339-7820 (A.Dudnik); 0000-0003-3892-4543 (N.Dakhno)

The purpose of the work is an information system for predicting the behavior of bank customers using machine learning methods. To achieve this goal, it is necessary to review and compare different machine learning methods, determine their effectiveness in the context of predicting customer behavior, and develop a user interface for convenient interaction with the system. Such a system is designed to help organizations better understand the behavioral patterns of their users, which will allow them to make more accurate and effective decisions in the field of interaction with customers.

To achieve this goal, the following tasks were performed:

- Analyze machine learning models and evaluate model performance: compare the results of different models and select the best ones using appropriate metrics.
- Research and collect data on user behavior, including cleaning, normalization and selection of relevant features that will have the greatest impact on analysis results.
- Develop a model to predict future user behavior and evaluate its accuracy using relevant performance metrics.

2. Analysis of machine learning models of user behavior

In modern research, considerable attention is paid to machine learning methods and their application in various fields, in particular, in the management of unmanned aircraft systems and process optimization. For example, works [1, 2] consider modified gradient methods for controlling unmanned aerial vehicles, using integro-differential models. These approaches can be adapted for the tasks of predicting user behavior, where high accuracy and efficiency of algorithms are required. In addition, the works [3, 4] investigate optimization and management models in conditions of limited resources, which has practical value for the development of information systems focused on the analysis of user behavior. In the context of modeling complex systems, as shown in [5], the use of genetic algorithms to create artificial ecosystems demonstrates the possibilities of optimizing complex processes, which can also be useful in predicting behavioral models. Thanks to the different approaches to text classification described in [6], it is possible to improve the methods of analyzing text data for studying user behavior.

Research [7] uses web usage mining to evaluate the quality of websites, improving the user experience. In [8], an algorithm based on machine learning is proposed for predicting behavior in "smart" homes, which adapts systems to user actions. Research [9] demonstrates how behavioral analysis can reduce risks in border regions, and work [10] describes a model for e-commerce that takes into account global trends, improving the compliance of platforms with digital requirements. These works confirm the role of machine learning in modeling behavior for different contexts.

The choice of machine learning models for predicting the behavior of bank customers is an important stage in the development of an information system. The following models were chosen in this work: Support vector method (SVM), k-nearest neighbors method (KNN), Decision tree, Random Forest, Logistic Regression and Deep Neural Networks. This section is devoted to the analysis of the reasons for choosing these methods, their advantages and disadvantages, as well as the evaluation of their effectiveness in the context of this project.

2.1. Method of support vectors

The method of support vectors (SVM) is a powerful machine learning tool widely used for classification and regression. In this subsection, we will consider the theoretical basis of SVM, model parameters and their meaning, as well as examples of their use.

SVM is based on finding the optimal hyperplane that separates two classes in the multidimensional feature space with the maximum margin. The hyperplane is chosen so that the distance between it and the nearest points of both classes (support vectors) is maximal.

In the case when the data are not linearly separable, SVM uses kernel functions (kernel functions) to transform the feature space into a higher-dimensional space where the data can become linearly separable.

When applying the support vector method, the settings of various parameters that affect the performance and accuracy of the model are important. The main SVM parameters include: C (regularization parameter), Kernel (kernel function), Gamma (kernel function parameter), Degree (used for the polynomial kernel and determines the degree of the polynomial), Coef0 (kernel constant).

The SVM kernel is symmetric, positive semi-definite matrix K consisting of scalar products of pairs:$x_i\ x_j$: $K(x_i, x_j) =< f(x_i), f(x_j) >$,

where f - an arbitrary transforming function for forming the kernel.

For example:

1. linear core:$K(x_i, x_j) = x_i^T x_j$
2. sigmoid nucleus: $K(x_i, x_j) = \tan(\gamma x_i^T x_j + \beta_0)$
3. Gaussian kernel with the function: $K(x_i, x_j) = \exp(\gamma||x_i - x_j||^2)$
4. polynomial kernel with degree p:: $K(x_i, x_j) = (1 + x_i^T x_j)^p$.

The support vector method (SVM) is a powerful tool for solving classification and regression problems. Using different kernel functions and adjusting model parameters allows SVM to be adapted to different types of data and tasks, ensuring high accuracy and reliability of predictions.

2.2. The method of k-nearest neighbors (KNN)

The k-nearest neighbor method is a simple and popular machine learning algorithm used for classification and regression. It is based on the assumption that similar data have similar labels or values [11] (Figure 1).

Figure 1: Work diagrams of the k-nearest neighbors algorithm

The basic principle of KNN is to determine a label or value for a new sample by comparing it with its nearest neighbors from the training data set. The number of nearest neighbors (k) is user-defined and affects the classification or regression process.

The KNN algorithm has several key features. Choosing the number of neighbors (k) is an important step that depends on the specific data and problem. A distance metric, such as the Euclidean distance metric, is used to measure proximity between samples, and this choice is also context dependent. The label of the new model is determined by the principle of majority voting among the nearest neighbors. Since KNN can be sensitive to class scaling and imbalance, data pre-processing such as normalization or class balancing is important. In addition, as the dimensionality of the data increases, the computational complexity of the algorithm increases, which can become a problem for large data sets. Choosing the right hyperparameters, such as the number of neighbors and the distance metric, is critical to achieving the best results.

There are many metrics for calculating the distance between objects [12], among which the most popular are:

- Euclidean distance is the simplest and generally accepted metric, which is defined as the length of a segment between two objects and in space with features and is calculated by the formula:

$$d(a,b) = \sqrt{\sum_{i=1}^{n}(a_i - b_i} \quad (1)$$

- The Manhattan distance is a metric that is defined as the sum of the moduli of the differences in the coordinates of two points in space between two objects and with features and is calculated by the formula:

$$d(a,b) = \sum_{i=1}^{n}|a_i - b_i| \quad (2)$$

- The cosine distance is a metric that is defined as the angle between two vectors and in feature space and is calculated by the formula:

$$d(a,b) = 1 - \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} {a_i}^2}\sqrt{\sum_{i=1}^{n} {b_i}^2}} \quad (3)$$

Considering all these aspects, the k-nearest neighbors method can be a useful and efficient algorithm, especially for simple classification and regression problems. However, it has its limitations and requires appropriate data processing and hyperparameter tuning to achieve the best results.

2.3. The decision tree method

Decision trees are known in the machine learning world for a particular distinguishing characteristic: their visualization is easier to understand compared to other machine learning models. Figure 2 shows a graphical representation of a typical decision tree for classification. This is for the hypothetical situation where a person wants to see an R-rated movie in a theater.

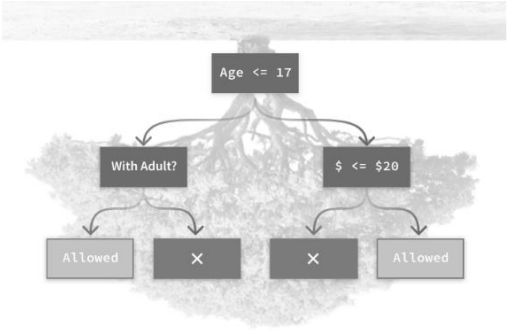

Figure 2: Tree of the Decision method Tree

Target the function, which determines the informativeness of features, is defined as follows [13]:

$IG(D_p, f) = I(D_p) - \sum_{j=1}^{n} \frac{N_j}{N_p} I(D_j)$, where

F - the sign by which splitting occurs;

D_p, D_j - is the data set of the parent and j th child node;

N_p, N_j - the total number of copies in the parent i jth daughter nodes;

I - peace inhomogeneities.

The method is used both for classification problems and for regression problems. A decision tree is a hierarchical structure in which each node is responsible for checking a certain condition, and each branch represents the result of this check. The leaf nodes of the tree contain the final solutions or predicted values.

A key feature of decision tree construction algorithms is the method of selecting the next attribute. There are the following algorithms [14]:

- the ID3 algorithm, where the selection of the attribute is based on the increase or on the basis of the Gini index;
- algorithm C4.5 (improved version of ID3), where the selection of the attribute is based on the normalized increase of information;
- the CART algorithm and its modifications – IndCART, DB-CART [15].

The decision tree method uses three measures inhomogeneities (or splitting criteria):

- Entropy $I_H(t) = \sum_{i=1}^{c} p(i|t) log_2 p(i|t)$ for all nonempty classes$p(i|t)$
  p_i - part of the elements from the i -th class for node t.
  The maximum is reached with a uniform distribution classes
- Mira heterogeneity Genie $I_G(t) = 1 - \sum_{i=1}^{c} p(i|t)^2$.
  This value shows the frequency that an element of the training sample is randomly selected is recognized incorrectly, provided that the value of the target known functions distribution That is, the distance between distribution target values and distribution of model predictions. The maximum, as in entropy, is reached at uniform distribution classes
- Classification error $I_E(t) = 1 - \max\{p(i|t)\}$.

Advantages of the decision tree method: simplicity and intuitive comprehensibility; the decision tree is easy to visualize and interpret; can process both numerical and categorical data; does not require data scaling. Disadvantages of the method: tendency to overfitting, especially on small data sets; can be unstable because small changes in the data can lead to large changes in the tree structure.

2.4. Random forest method

A random forest is an ensemble machine learning method that consists of a large number of decision trees. The main idea of the method is to combine the results of several decision trees to obtain a more accurate and stable forecast. A random forest uses the bagging method to build a set of decision trees, each of which is trained on a random subset of data [16-17].

To build an ensemble of algorithms based on bagging using bootstrap, samples are generated, on each of which a classifier is trained $a_i(x)$. The resulting classifier will average the responses of all algorithms (in the case of classification, this corresponds to voting):

$$a(x) = \frac{1}{M} \sum_{i=1}^{M} a_i(x). \quad (4)$$

Bagging allows to reduce the variance of the classifier and prevents overtraining. The effectiveness of bagging is achieved due to the fact that the basic algorithms trained on different subsamples are quite different, and their errors are mutually compensated during voting. In addition, outliers may not be included in some training subsamples.

All trees of the ensemble are built independently of each other according to the following procedure:

- generate a random subsample of size n from the training sample;
- build a decision tree, and during the creation of the next tree node, not all features are considered, but only m randomly selected features, on the basis of which the division will be carried out;
- the tree is built until the objects of the subsample are completely exhausted and is not subject to the branch cutting procedure.

Advantages of the random forest method: high accuracy and stability; a random forest is less prone to overtraining than individual decision trees; can efficiently process large data sets and a large number of attributes; is relatively insensitive to missing data and can handle a dataset with missing values. Disadvantages of the random forest method: high computational complexity and

memory consumption due to the large number of trees; difficult to interpret because the result is an average of many trees.

2.5. Logistic regression method

The logistic regression method (logistic regression) is a statistical method in machine learning that is used to predict the probability of belonging to two or more categories or classes. It is one of the most common algorithms for binary classification problems [18-19].

The main idea of the method is to model the logarithm of the odds (the logarithm of the likelihood ratio) as a linear combination of the input attributes. A logistic function (also known as a sigmoid) is used to convert a linear combination into a probability of belonging to a particular class.

The main steps of the logical regression method:

1. Data preparation: Loading and preprocessing of data, including scaling, normalization or removal of missing values.
2. Model definition: establishing a logistic regression model that includes input attributes and model parameters.
3. Parameter estimation: using the maximum likelihood method or other optimization methods, model parameters that best fit the training data are estimated.
4. Classification: with the help of a trained model, the probability of belonging to a certain class for new data samples is predicted. A classification decision is made based on the probability threshold.
5. Evaluation of the results: evaluate the accuracy and efficiency of the model using metrics such as accuracy, sensitivity, specificity or ROC curve.

Logistic regression has several advantages, such as ease of implementation, interpretability of results, ability to work with different types of attributes (categorical and numerical) and copes well with large amounts of data. However, it may also have limitations, such as linearity assumptions and vulnerability to outliers or unbalanced data. Regularization techniques, selection of optimal attributes or use of ensemble methods can be used to improve the results.

2.6. Deep neural networks

Deep neural networks (Deep Neural Networks, DNN) are powerful machine learning models that consist of many layers of neurons. They are capable of automatically learning complex nonlinear relationships in large volumes of data and are widely used for a variety of classification problems. In this subsection, we will consider the theoretical foundations of deep neural networks, their architecture, parameters and examples of use [20-21].

A DNN is a multilayer neural network that consists of an input layer, one or more hidden layers, and an output layer. Each layer consists of neurons that are interconnected by weighted connections. Neurons calculate the weighted sum of their input signals, apply an activation function to this sum, and pass the result to the next layer.

Main components of DNN:

1. Neurons (Nodes): basic computing elements that receive input signals, calculate a weighted sum and apply an activation function.
2. Layers: collections of neurons, where each layer processes the output signals of the previous layer.

- the input layer receives initial data;
- hidden layers process data, revealing complex patterns and dependencies;
- the output layer generates the final predictions or classifications.

3. Activation functions (Activation Functions): non-linear functions applied to the original sum of neurons. Popular functions include ReLU, Sigmoid, Tanh, and Softmax.

The architecture of deep neural networks defines the number of layers, the number of neurons in each layer, and the types of activation functions. Some popular architectures include:

1. Direct neural networks (Feedforward Neural Networks, FNN) [22]: the simplest form of DNN, where signals pass from the input layer to the output layer through one or more hidden layers without feedback.
2. Convolutional neural networks (Convolutional Neural Networks, CNN) [23].
3. Recurrent neural networks (Recurrent Neural Networks, RNN) [24].

The main parameters that are adjusted when building and training deep neural networks include:

- Number of layers (Number of Layers): a larger number of layers allows the model to detect more complex patterns, but also increases computational costs and the risk of overtraining.
- The number of neurons in the layer (Number of Neurons per Layer): defines the number of computing units in each layer. Increasing the number of neurons can improve the learning ability of the model, but also increases the risk of overtraining.
- Activation functions (Activation Functions): ReLU (Rectified Linear Unit), Sigmoid, Softmax.
- Speed of learning (Learning Rate):
- Batch size (Batch Size):
- Number of epochs (Number of epochs.

Deep neural networks (DNNs) are a powerful tool for solving classification problems due to their ability to learn complex nonlinear relationships in large volumes of data. The choice of network architecture and parameter settings are critical to achieving high model accuracy and performance. In the context of predicting customer behavior, DNNs can provide high classification accuracy, especially when dealing with large and complex datasets.

3. Collection and preparation of data on user behavior

The subscription_prediction.csv file was used as a Dataset to study the behavior of the bank's clients. It contains data on clients of the banking institution and consists of 21 columns: age, job, marital status, education, default, housing, loan (availability of a personal loan), contact (type of communication), month (month of last contact), day_of_week (day of last contact), duration (duration of last contact in seconds), campaign (number of contacts made during this campaign and for of this client), pdays (the number of days that have passed since the last contact with the client from the previous campaign), previous (the number of contacts made before this campaign and for this client), poutcome (the result of the previous marketing campaign), emp.var. rate (rate of employment change - quarterly indicator), cons.price.idx (consumer price index - monthly indicator), cons.conf.idx (consumer sentiment index - monthly indicator), euribor3m (3-month euribor rate - daily indicator), nr.employed (the number of employees is a quarterly indicator), y (has the client signed a term deposit).

To read data from a CSV file and create a Dataframe, the function " pandas.read_csv ()" from the pandas library is used (Figure 3):

```
import pandas as pd
banking_df = pd.read_csv('subscription_prediction.csv')
```

Figure 3: Reading data and creating a Dataframe

Data preparation plays an important role in the implementation of the algorithm, so it is necessary to carefully examine the data set. ". head ()" method used to display the first few rows of the DataFrame.

After examining the results of the methods, it was found that there are no missing values in the data set. Also, the target column 'y' takes the value " yes " or " no ". For convenience, it is necessary to encode these lines as numbers "0" for " no " and "1" for " yes ".

For error-free operation of the fit () method, which will be used later, it is necessary to convert all categorical columns into binary variables (Figure 4):

```
banking_df = pd.get_dummies(data = banking_df, drop_first = True)
```

Figure 4: Conversion of columns to binary variables

The variable "top_5_features" contains the indices of the five variables that have the highest correlation with the target variable. These variables are the most informative for predicting the target variable and can be used for further analysis or model building (Figure 5).

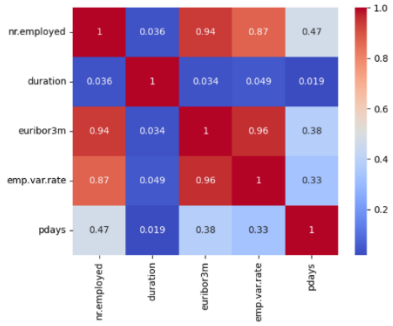

Figure 5: Correlation matrix of variables from “top_5_features”

4. Development of a model for predicting user behavior

The next step will be to create and train the classifier using the scilit-learn library. The goal when training a model is to evaluate its performance on a test data set or on new data. To ensure accuracy, it is necessary to create an intermediate data set between training and test data.

Thus, the initial data set will be divided into three parts: training set (60%), validation set (20%), test set (20%). To implement this task, there is a method " train_test_split ()" (Figure 6):

```
X = banking_df.drop('y',axis = 1) #Змінна X створюється шляхом видалення стовпця 'y' з banking_df.
y = banking_df['y']               #Змінна y містить цільовий стовпець 'y' з banking_df.
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.20, random_state = 417)

X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.20*X.shape[0]/X_train.shape[0],
                                                    random_state = 417)
```

Figure 6: Splitting the data into three parts

In the next step, you need to normalize the values of the columns by scaling their values to the range [0, 1].

NeighborsClassifier " is used to create a classifier using the k-nearest neighbors algorithm. In this case, the " knn " model is used as the base classifier, and the " grid_params " is passed as the " param_grid " parameter in the " GridSearchCV " constructor. The " scoring =' accuracy '" parameter indicates the use of the accuracy metric to evaluate the model.

After initializing the " knn_grid " object of the " GridSearchCV " class, the "fit()" method is called, which starts the search for the best hyperparameters on the training data set that has been pre-scaled by " X_train_scaled " and the corresponding target values " y_train ". The algorithm goes through all possible combinations of hyperparameters, calculates the results according to a certain metric (in this case, accuracy), and stores the best combination of hyperparameters in the " knn_grid " object.

After calling " fit()", the " knn_grid " model will contain the best hyperparameter values that can be used for further prediction and evaluation.

The last stage of the model implementation is the derivation of the efficiency of its work (Figure 7):

```
X_test_scaled = scaler.transform(X_test)
accuracy = knn_grid.best_estimator_.score(X_test_scaled, y_test)
print(f"Model Accuracy on test set: {accuracy*100:.2f}")

y_pred = knn_grid.predict(X_test_scaled)
report = classification_report(y_test, y_pred, target_names=["no", "yes"])
print("\nClassification Report:")
print(report)
```

Figure 7: Output of the accuracy of the model

In this code, the accuracy of the model is evaluated on the test data set after finding the best hyperparameters using " Grid Search ».« X_test_scaled » - test data set " X_test " that was scaled using " scaler.transform ()". This is done to ensure the same scale between the training and test datasets. " accuracy " is a variable that stores the accuracy of the model on the test data set. The score() method is called on the best estimator " best_estimator " of the " knn_grid " object (the NeighborsClassifier model with the best hyperparameters), and the accuracy of the model is calculated on the test data [25].

The main difference from the k-nearest neighbors implementation is the use of LogisticRegression () instead of NeighborsClassifier. LogisticRegression () is a class in the scikit-learn library that is used to build a logistic regression model.

Implementation by the decision tree method is slightly different from, for example, the KNN method:

The main difference is as follows:

- for KNN, KneighborsClassifier from the sklearn.neighbors library is used. DecisionTreeClassifier from the sklearn.tree library is used for the decision tree;
- in the case of KNN, the hyperparameter is configured n_neighbors (number of neighbors) and metric (Euclidean, Manhatta, etc.). In the case of a decision tree, hyperparameters are set, such as the criterion (gini or entropy) and the maximum depth of the tree;

In general, both approaches have a similar structural skeleton for loading data, training, training a model, and evaluating its accuracy, but they use different algorithms for classification.

The RandomForestClassifier from the sklearn.ensemble library is used for the random forest. A random forest combines decision trees to reduce overtraining and improve accuracy. SelectKBest with f_classif is also used to select the top 5 features using analysis of variance [26].

Feature selection can improve a model's speed and efficiency, but it can affect its workflow and decisions. To reduce the code execution time, in this case, RandomizedSearchCV was used, it allows you to explore a random subset of the hyperparameter space [27-29].

The implementation of the model differs by importing additional libraries for working with data, building and training a neural network: "Sequential", "Dense", "Dropout", "Adam", "to_categorical", "tensorflow.keras". Next, a neural network is built. The network consists of three " Dense " layers (fully connected layers), each of which has a different number of neurons and uses the " ReLU " activation function, except for the output layer, which uses the "softmax " activation function for the classification task. Two " Dropout " layers are placed after each hidden layer to help prevent overtraining by turning off random neurons during training.

This architecture provides flexibility in training complex models capable of recognizing patterns in data, while " Dropout " helps avoid overtraining problems.

5. Evaluation of the quality of models

The accuracy of the k-nearest neighbors model was 86.62%. With the help of the " classification_report ()" function, a detailed report on the classification quality indicators for the machine learning model was obtained (Figure 8).

```
Model Accuracy on test set: 86.62

Classification Report:
              precision    recall  f1-score   support

          no       0.91      0.83      0.87      1103
         yes       0.82      0.90      0.86       922

    accuracy                           0.87      2025
   macro avg       0.87      0.87      0.87      2025
weighted avg       0.87      0.87      0.87      2025
```

Figure 8: Display of model accuracy

Classification report (Classification Report) provides detailed information about the classification results:

- precision: measures how many positive predictions were correct. For " no " it is 0.91, for " yes " - 0.82. This means that the model was accurate in predicting the " no " class 91% of the time and the " yes " class 82% of the time;
- recall (sensitivity): this measures what fraction of true positive cases were found by the model. For " no " it is 0.83, for " yes " – 0.90. This means that the model is able to reproduce 83% of positive cases for " no " and 90% of positive cases for " yes ";
- f1-score (F1-index): indicates the balance between accuracy and sensitivity. For " no " it is equal to 0.87, for " yes " - 0.86;
- support: this is the number of instances in each class in the test data set.

“macro avg”, " weighted avg” represent the average value for each indicator across all classes. “Macro avg" calculates the average regardless of class size, while " weighted avg » takes class size into account.

The accuracy of the model by the method of support vectors is 86.62% (Figure 9), To assess the quality of the model, the validation curve graph is also used (Figure 10):

```
Model Accuracy on test set: 86.62%

Classification Report:
              precision    recall  f1-score   support

          no       0.92      0.83      0.87      1103
         yes       0.82      0.91      0.86       922

    accuracy                           0.87      2025
   macro avg       0.87      0.87      0.87      2025
weighted avg       0.87      0.87      0.87      2025
```

Figure 9: Display of model accuracy by the method of support vectors

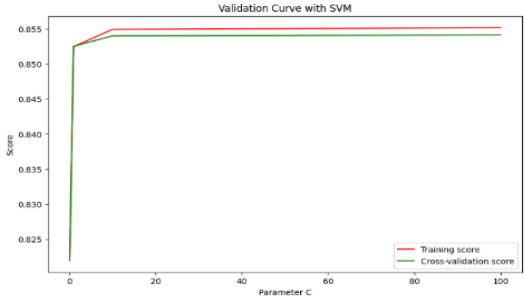

Figure 10: Graph of the validation curve for parameter

Training score (red line) and Cross-validation score (green line) show the estimation of the accuracy of the model at different values of the C parameter. Both graphs have a similar shape and reach a plateau around C = 1. This indicates that the model performs well already at low values of C, and increasing this parameter does not brings significant performance improvements. The

accuracy of the model stabilizes at the level of about 0.855. This indicates a good overall quality of the model, since the cross- validation accuracy is very close to the training accuracy, which indicates the absence of strong overfitting or underfitting.

The accuracy of the logistic regression model was 84.20%, which is quite a good result (Figure 11).

The ROC curve (Receiver Operating Characteristic curve) (Figure 12). The ROC curve displays the relationship between sensitivity (True Positive Rate, TPR) and specificity (True Negative Rate, TNR) of the classifier at different threshold values. True Positive Rate defines how often the classifier correctly identifies positive examples among all true positive examples. False Positive Rate defines how often the classifier incorrectly identifies negative examples among all true negative examples. An ROC curve is a graph where the X- axis shows FPR and the Y- axis shows TPR. Each point in this graph corresponds to a different threshold value at which the classifier identifies positive and negative examples.

The ROC curve diagonally passes through the point (0,0) and (1,1) and corresponds to a classifier that randomly determines a class. The optimal classifier will lie above this diagonal.

The area under the ROC curve (AUC-ROC) determines the overall quality of the classifier. The larger the AUC-ROC, the better the classifier. Typically, AUC-ROC ranges from 0 to 1. A classifier with an AUC-ROC of 0.5 corresponds to random class selection, while a classifier with an AUC-ROC close to 1 is considered very effective.

```
Model Accuracy on test set: 84.20

Classification Report:
              precision    recall  f1-score   support

          no       0.85      0.86      0.86      1103
         yes       0.83      0.82      0.82       922

    accuracy                           0.84      2025
   macro avg       0.84      0.84      0.84      2025
weighted avg       0.84      0.84      0.84      2025
```

Figure 11: Representation of the accuracy of the logistic regression model

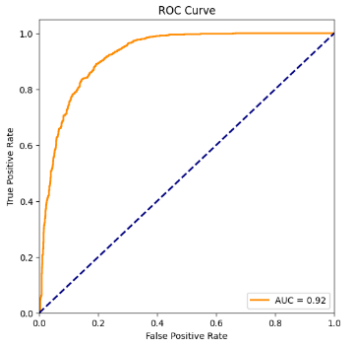

Figure 12: ROC curve graph

In the ROC curve plot, the optimal classifier will have an ROC curve that approaches the upper left corner of the plot and is 0.92, which corresponds to a high TPR and a low FPR at any threshold value.

The accuracy of the decision tree model was 86.81%, which is shown in Figure 13:

```
Model Accuracy on test set: 86.81

Classification Report:
              precision    recall  f1-score   support

          no       0.90      0.85      0.88      1103
         yes       0.84      0.89      0.86       922

    accuracy                           0.87      2025
   macro avg       0.87      0.87      0.87      2025
weighted avg       0.87      0.87      0.87      2025
```

Figure 13: Accuracy of the decision tree model

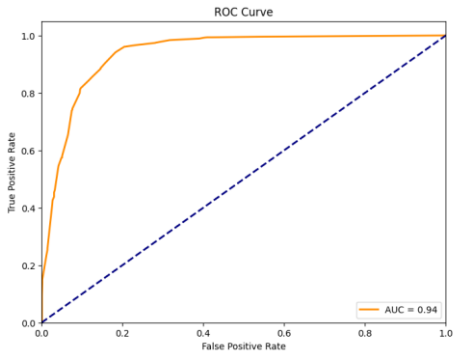

Figure 14: Graph of the ROC curve of the decision tree model

The model has high accuracy and is able to effectively distinguish between positive and negative classes, which is confirmed by a high AUC value (0.94). The ROC curve indicates a good balance between sensitivity and specificity, which makes the model reliable for classification tasks.

The model built on the basis of **the random forest method** has an accuracy of 88.05%, which is shown in Figure 15. The model has high accuracy and is able to effectively distinguish between positive and negative classes, which is confirmed by a high AUC value (0.94). The ROC curve indicates a good one balancing between sensitivity and specificity that makes the model reliable for classification problems (Figure 16):

```
Model Accuracy on test set: 88.05

Classification Report:
              precision    recall  f1-score   support

          no       0.92      0.85      0.89      1103
         yes       0.84      0.91      0.87       922

    accuracy                           0.88      2025
   macro avg       0.88      0.88      0.88      2025
weighted avg       0.88      0.88      0.88      2025
```

Figure 15: Accuracy of the random forest model

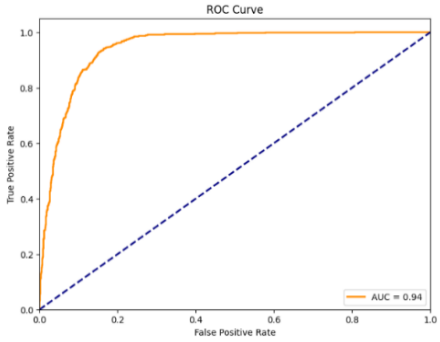

Figure 16: Graph of the ROC curve

Value AUC = 0.94 is very tall indicator that indicates a high productivity models. The curve shows high True Positive Rate even at low False Positive Rate that means that the model detects well positive cases at a minimum quantity false positives activations. Proximity curve to the top left corner of the graph indicates a high sensitivity and specificity models. The model is significant deviates from random lines, confirming high precision models.

The evaluation was carried out using the following metrics and is presented in Table 1:

- Accuracy;
- Precision;
- Recall;
- F- measure.

The table shows the comparison of different machine learning algorithms applied to predict user behavior in terms of metrics such as accuracy, accuracy, completeness, and F-measure. High accuracy rates are observed for KNN, SVM, decision tree, and random forest methods with values ranging from 0.87 to 0.88, indicating their ability to make correct predictions. The random forest method performed best with an accuracy of 0.88, indicating its potential to predict general user behavior patterns, while logistic regression performed slightly lower (0.84).

Table 1
Evaluation methods

	The method of k-nearest neighbors	By the method of support vectors	Method of logistic regression	Decision tree method	Random forest method
Metrics	Result	Result	Result	Result	Result
Precision	0.87	0.87	0.84	0.87	0.88
Precision	0.82	0.82	0.83	0.84	0.84
Completeness	0.9	0.91	0.82	0.89	0.91
F- measure	0.86	0.86	0.82	0.86	0.87

In the context of user behavior analysis, accuracy and completeness help assess a model's ability to correctly identify positive behavioral patterns. All algorithms have similar accuracy rates (about 0.82-0.84), which indicates their effectiveness in correctly predicting certain user behavior. However, the SVM and random forest methods showed the highest completeness values (0.91), indicating their ability to detect all possible behavior scenarios without missing any important cases. The F-measure as a harmonic mean of accuracy and completeness confirms the balance and reliability of the models, particularly in the decision tree, SVM (0.86) and random forest (0.87) methods, making these approaches the most effective for predicting complex user behavior.

6. Conclusions

This paper analyzed k-nearest neighbors (KNN), support vector (SVM), logistic regression, decision tree, random forest, and deep neural network methods for predicting user behavior. The study found that all six methods are effective and capable of providing accurate predictions of customer behavior.

The behavior of the bank's customers was studied based on such parameters as demographic data, the history of interaction with the bank, the availability of loans, the method of communication, as well as macroeconomic indicators. The analysis showed that certain factors, such as age, type of work, availability of a home loan and previous campaigns of the bank, have a significant impact on the probability of a client signing up for a term deposit.

According to the obtained results, the accuracy of the k-nearest neighbors method was 86.62%, which indicates its good prognostic ability. The support vector method (SVM) also showed an accuracy of 86.62%. Logistic regression showed an accuracy of 84.20%, which is also a satisfactory result. The decision tree model had an accuracy of 86.81%, while the accuracy of the random forest model was 88.20%, which is the highest among all methods studied. The deep neural network (DNN) showed an accuracy of 86.00%.

Based on these results, it can be concluded that the random forest is the most effective among the studied methods for predicting the behavior of bank customers. However, the choice of method should depend on the specifics of the data and the specifics of the task. In some cases, a combination of methods, such as using k-nearest neighbors and logistic regression, can lead to even better results and improve the quality of the prediction.

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] N. Dakhno, O. Barabash, H. Shevchenko, O. Leshchenko and A. Musienko, " Modified Gradient Method for K- positive Operator Models for Unmanned Aerial Vehicle Control," 2020 IEEE 6th International Conference he Methods and Systems of Navigation and Motion Control (MSNMC), KYIV, Ukraine, 2020, pp. 81-84, doi: 10.1109/MSNMC50359.2020.9255516.

[2] Dakhno N., Barabash O., Shevchenko H., Leshchenko O., Dudnik A. Integro-differential Models with a K- symmetric Operator for Controlling Unmanned Aerial Vehicles Using a Improved Gradient Method. 2021 IEEE 6th International Conference " Actual Problems of Unmanned Aerial Vehicles Development (APUAVD). Proceedings. October 19 – 21, 2021, Kyiv, Ukraine. P. 61 – 65. DOI: 10.1109/APUAVD53804.2021.9615431.

[3] Dakhno, N., Leshchenko, O., Kravchenko, Y., Dudnik, A., Trush, O., Khankishiev, V. Dynamic model of the spread of viruses in a computer network using differential equations (2021) 2021 IEEE 3rd International Conference he Advanced Trends in Information Theory, ATIT 2021 - Proceedings, pp. 111-115.

[4] Shevchenko, H., Dakhno, N., Leshchenko, O., Barabash, O., Kravchenko, Y., & Dudnik, A. (2022, December). Using Mathematical Optimization Methods that Maximize Audience Reach with Budget Constraints. In 2022 IEEE 4th International Conference he Advanced Trends in Information Theory (ATIT) (pp. 249-254). IEEE.
[5] Y. Kravchenko, O. Leshchenko, N. Dakhno, O. Pliushch, O. Trush and Y. Yermakov," Development of Model of Artificial Ecosystem he the Basis of Genetic Algorithm," 2022 IEEE 4ht International Conference he Advanced Trends in Information Theory (ATIT), 2022, pp. 199–203
[6] Kovaluk T., Dukhnovska K., Kovtun O., Nikolaienko, A., Yurchuk, I. Text classification using term co-occurrence matrix. XX International Scientific Conference " Dynamic System Modeling and Stability Investigation " (DSMSI-2023). December 19-21, 2023.
[7] Sawant, P., & Kulkarni, R. (2013). A Knowledge Based Methodology To Understand The User Browsing Behavior For Quality Measurement Of The Websites Using Web Usage Mining. International Journal Of Engineering And Computer Science, 2, 1522-1538.
[8] T. Liang, B. Zeng, J. Liu, L. Ye and C. Zou, "An Unsupervised User Behavior Prediction Algorithm Based on Machine Learning and Neural Network For Smart Home," in IEEE Access, vol. 6, pp. 49237-49247, 2018, doi: 10.1109/ACCESS.2018.2868984.
[9] Hubanova, T., Shchokin, R., Hubanov, O., Antonov, V., Slobodianiuk, P., Podolyaka, S. Information technologies in improving crime prevention mechanisms in the border regions of southern Ukraine (2021) Journal of Information Technology Management, 13, pp. 75-90. Cited 50 times. DOI: 10.22059/JITM.2021.80738
[10] Alazzam, F.A.F., Shakhatreh, H.J.M., Gharaibeh, Z.I.Y., Didiuk, I., Sylkin, O. Developing an Information Model for E-Commerce Platforms: A Study on Modern SocioEconomic Systems in the Context of Global Digitalization and Legal Compliance (2023) Ingenierie des Systemes d'Information, 28 (4), pp. 969-974. Cited 32 times. DOI: 10.18280/isi.280417
[11] Aggarwal, CC, Zhai, C. A survey of text classification algorithms. In: Aggarwal, CC, Zhai, C (ed.) Mining text data. Berlin: Springer, 2012
[12] MACHINE LEARNING AZ™: DOWNLOAD PRACTICE DATASETS [Electronic resource]– Resource access mode: https://www.superdatascience.com/machine-learning/ 4. Russell, SJ, Norvig, P. Artificial intelligence: a modern approach. 3rd ed. Upper Saddle River, NJ: Prentice hall /
[13] Li, S., & Amenta, N. (2015). Brute-force k- nearest neighbors search he the GPU. In Similarity Search and Applications: 8th International Conference, SISAP 2015, Glasgow, UK, October 12-14, 2015, Proceedings 8 (pp. 259-270). Springer International Publishing.
[14] Hopcroft, JE, Ullman, JD, & Aho, AV (1983). Data structures and algorithms (Vol. 175). Boston, MA, USA: Addison-wesley.
[15] Avinash Navlani « Support Vector Machines with Scikit-learn ", 2019. [Electronic resource] - Resource access mode: https://www.datacamp. com / community / tutorials / svm-classification-scikit-learn-python
[16] Patel, F. Decision Tree the CART Algorithm. [Electronic resource]. – https://medium.com/analytics-vidhya/decision-tree-the-cart-algorithm-28c481d28813
[17] Breiman, L. Classification and Regression Trees / L. Breiman, JH Friedman, RA Olshen, CT Stone. – Chapman and Hall/CRC, 1984. – 368
[18] Velu, A. (2021). Application of logistics regression models in risk management. International Journal of Innovations in Engineering Research and Technology, 8(04), 251–260. Retrieved from https://repo.ijiert.org/index.php/ijiert/article/view/2594.
[19] 27. Hosmer, DW, & Lemeshow, S. (2000). Applied Logistics Regression. John Wiley & Sons, Inc. DOI: https://doi.org/10.1002/0471722146.
[20] Akanbi, LA, Oyedele, AO, Oyedele, LO, & Salami, RO (2020). Deep learning model for Demolition Waste Prediction in a circular economy. Journal of Cleaner Production, 274, 122843.

[21] Schulz, Hannes; Behnke, Sven (1 November 2012). "Deep Learning". KI - Künstliche Intelligence. 26 (4): 357–363. doi:10.1007/s13218-012-0198-z. ISSN 1610-1987. S2CID 220523562.
[22] Jump up to:a b LeCun, Yann; Bengio, Yoshua; Hinton, Geoffrey (2015). " Deep Learning " (PDF). Nature. 521 (7553): 436–444. Bibcode:2015 Natur.521..436L. doi:10.1038/nature14539. PMID 26017442. S2CID 3074096.
[23] Le, QV (2013, May). Building high-level features using large scale unsupervised learning. In 2013 IEEE international conference on acoustics, speech and signal processing (pp. 8595-8598). IEEE.
[24] LeCun, Y.; et al. (1998). " Gradient-based learning applied that document recognition ". Proceedings of the IEEE. 86 (11): 2278–2324. doi:10.1109/5.726791
[25] Abdel-Hamid, O.; et al. (2014). "Convolutional Neural Networks for Speech Recognition". IEEE/ACM Transactions on Audio, Speech, and Language Processing. 22 (10): 1533–1545. doi: 10.1109/taslp.2014.2339736.
[26] Metrics and scoring: quantifying the quality of predictions. URL: https:// scikit - learn. org / stable / modules / model _ evaluation. html (date of application 04/03/2024). (electronic resource)
[27] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. the Journal of machine learning research, 12, 2825-2830.
[28] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. Journal of machine learning research, 13(2).
[29] Brownlee, J. (2019). A gentle introduction that the rectified linear unit (ReLU). Machine learning mastery, 6.