

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION
FEDERAL STATE AUTONOMOUS EDUCATIONAL
INSTITUTION OF HIGHER EDUCATION
"NORTH CAUCASUS FEDERAL UNIVERSITY"

SUGGESTIONS FOR TEACHING
for the performance of practical work
in the discipline "**Statistical Methods in Machine Learning**"
for students of the field of study
11.03.02 Infocommunication Technologies and Communication Systems
Profile
"Intelligent Data Processing in Infocommunication Systems and Networks"

Table of Contents

Introduction

Practical Work No1 The Basics of Working with Jupyter Notebook, JupyterLab, and Google Colab

Practical Work No2 NumPy Package Basics

Practical Work No3 Basics of working with the matplotlib package

Practical Work No4 Introduction to pandas: Object Series

Practical Work No5 Introduction to pandas: DataFrame object

Practical Work No6 Fundamentals of Research Data Analysis

Practice No7 Linear Regression

Practical Work No8 Logistic Regression

Practice No9 Quality Metrics of Classification Models

Practical Paper No10 Nearest Neighbor Method and Support Vector Method

Practical Work No11 Decision Trees

Practical Work No12 Random Forest

Practical Work No13 Gradient Boosting

Practical Work No14 Unsupervised Learning: Data Clustering Tasks

Practical Work No15 Unsupervised Learning: Data Dimensionality Reduction Problems

Practical Work No16 Selection of Model Hyperparameters

Conclusion

Educational, methodological and information support of the discipline

INTRODUCTION

1. Purpose and objectives of mastering the discipline (module)

The purpose of the discipline is to form, according to the educational program, the competencies of the OPK-1 student.

To achieve this goal, the following tasks are solved in the course of studying the discipline: familiarization with the basic concepts and types of machine learning problems; study of data preparation methods, including cleansing, normalization and engineering transformation of features; mastery of basic supervised learning models, such as linear and logistic regression; mastering stochastic gradient descent as a method of model optimization; forming ideas about metrics for assessing the quality of classification and interpretation of model results; studying methods for combating overfitting, including regularization and cross-validation; mastering classification algorithms based on the nearest neighbor method and the support vector method; studying algorithms for building decision trees and ensemble models, such as random forest and gradient boosting; familiarity with unsupervised learning methods, including clustering algorithms; mastering methods for reducing dimensionality and data visualization, such as PCA and t-SNE; developing practical skills in tuning models using hyperparameter selection; formation of skills to compare and interpret the results of various models based on objective metrics.

2. Place of the discipline (module) in the structure of the educational program

Discipline refers to the part formed by the participants of educational relations.

3. List of planned learning outcomes in the discipline (module) correlated with the planned results of the development of the educational program

Code, formulation of competence	Indicator code, wording	Planned learning outcomes in the discipline (module), characterizing the stages of the formation of competencies, indicators
OPK-1. Able to use the provisions, laws and methods of natural sciences and mathematics to solve problems of engineering activities	ID-1 OPK-1 understands the fundamental laws of nature and the basic physical mathematical laws and methods of accumulating, transmitting and processing information.	Knows and understands optimization methods in the design of AI and ML systems, is able to justify the choice.
	ID-2 OPK-1 applies physical laws and mathematical methods to solve problems of a theoretical and applied nature.	Ability to solve optimization problems in the design of machine learning systems.
	ID-3 OPK-1 uses the knowledge of physics and mathematics to solve practical problems.	Collecting, processing, and analyzing the results of ready-made machine learning systems for compliance.

4. Scope of the academic discipline (module) and forms of control *

Scope of classes: total: 3 EC 108 academic hours.	OFO, In acad. hours
Contact work:	

Lectures/from them practical training	32.00
Laboratory work/from them practical training	-
Practical training/of which practical training	
Independent work	76.00
Forms of control	
Exam	-
Credit	~~Yes~~
Credit with a grade	-
Course work	no

5. The importance of practical work of students

The practical work of students is the central element of mastering the discipline **"Statistical Methods in Machine Learning"**. In the context of the rapid development of technologies and the high applied orientation of this field, theoretical knowledge without practical application loses relevance and depth.

The main goal of practical classes is not only to consolidate theoretical material, but also to form students' real skills in working with algorithms, libraries and data analysis tools. Practice allows students to:

- understand how the theory is implemented in the code and how algorithms work with specific examples;
- learn how to work with modern Python libraries such as Scikit-learn, Pandas, and Matplotlib;
- work out the full cycle of building a model: from loading and cleaning data to evaluating and interpreting the results;
- identify and eliminate errors that occur during the implementation of models;
- compare different machine learning methods and choose the most suitable for the task;
- develop analytical thinking, learn to interpret metrics and visualize the behavior of the model.

Practical work also forms important competencies that are in demand in the professional environment: the ability to independently find solutions, work with open data, adjust model parameters and evaluate their effectiveness in real conditions. Regular performance of practical tasks increases students' motivation, builds confidence in their own abilities and prepares them for the use of machine learning in real projects and scientific research.

PRACTICAL WORK №1

Topic: Basics of working with Jupyter Notebook, JupyterLab and Google Colab

Purpose of the work:

Learn the basics of interactive computing in Jupyter Notebook, JupyterLab, and Google Colab. Learn the basics of entering, formatting, and executing code and text cells, as well as loading and saving work files.

Theoretical substantiation:

Jupyter Notebook and JupyterLab are interactive data development and analysis environments widely used in machine learning and artificial intelligence tasks. These frameworks support the Python language and allow you to combine code, explanatory text, visualizations, and inference in a single document. JupyterLab is a more modern and modular environment that extends the capabilities of the classic Jupyter Notebook.

Google Colab is a cloud-based platform that provides users with free access to laptops that can run on remote Google servers. Colab is especially useful when there is no on-premises environment and allows the use of computing resources, including GPUs and TPUs.

Working in interactive environments allows you to efficiently build and debug models, document the progress of the experiment, share the results and save them in a single file. Students should be able to distinguish between cell types (code and text), format text using Markdown, insert formulas in LaTeX, and manage data loading and library connections.

Knowledge and confident command of Jupyter Notebook, JupyterLab and Google Colab is the basis for further successful mastering of the discipline and performing laboratory, design and scientific tasks.

Equipment and materials:

Computer or laptop with internet access

Google Chrome or Mozilla Firefox browser

Installed JupyterLab or Jupyter Notebook environment (via Anaconda or pip)

Google account to access Google Colab and Google Drive

Example CSV file (provided by the instructor or taken from `sklearn.datasets`)

Software: Python 3.8+, libraries `numpy`, `pandas`, `matplotlib`

Typical tasks:

1. Create a Jupyter Notebook with a brief description of the goal of the future project and insert code samples that demonstrate how to perform arithmetic operations.
2. Open Google Colab, create a new laptop, connect it to your Google Drive and upload a CSV file, then print the first rows of the spreadsheet.
3. In JupyterLab, create a file that contains a text cell with a title, a bulleted list, and a LaTeX formula, and a code cell that implements a simple function.

Control questions:

1. How is Jupyter Notebook different from JupyterLab?
2. What advantages does Google Colab provide over on-premises environments?
3. How do I save my laptop on my local computer and in the cloud?
4. What are the main types of cells in Jupyter?
5. How do I insert formulas and rich text into a Markdown cell?
6. How do I connect to Google Drive in Colab?
7. What commands are used to install and mount libraries?
8. What is "runtime" in Google Colab?
9. How do I download data to my laptop and view its contents?
10. What export formats are available for Jupyter Notebook?

PRACTICAL WORK №2

Topic: NumPy Package Basics

Purpose of the work:

Master the basic operations of the NumPy library, learn how to create, index, and modify

multidimensional arrays, perform vector and matrix calculations, and use built-in functions to analyze and process numerical data.

Theoretical substantiation:

The NumPy library is fundamental to numerical computing in Python and serves as the basis for most of the tools used in machine learning and data analysis. The main object of NumPy is a multidimensional array `ndarray`, which provides high performance by storing data in a compact form and the ability to perform operations on arrays without the use of loops.

Working with arrays allows you to efficiently represent data samples, perform linear algebra, statistical operations, normalization, and other transformations necessary at the stage of data preprocessing. One of the key features of NumPy is the ability to perform operations element by element and vectorized, which significantly speeds up the processing of large amounts of information.

NumPy also includes a set of tools for generating random numbers, transforming the shape of arrays, combining and splitting data, and performing boolean operations and filtering. A thorough understanding of the capabilities of this library is essential for successfully mastering tools such as pandas, scikit-learn, and other machine learning libraries.

Equipment and materials:

Computer or laptop with internet access

On-premise Jupyter Notebook or Google Colab

Installed Library `numpy` (included with Anaconda or installed via `pip`)

Examples of arrays and tasks prepared by the teacher or formed by the student independently

Additional materials: documentation <https://numpy.org/doc>

Typical tasks:

1. Create one-dimensional and two-dimensional arrays using `np.array()`, output their shape, size, and data type.
2. Generate an array of random numbers using `np.random`, find the mean, minimum, and maximum.
3. Perform matrix multiplication of two compatible arrays using both `@` and `np.dot()`.

Control questions:

1. What is an `ndarray` object in the NumPy library?
2. What is the difference between elemental multiplication and matrix multiplication?
3. How do you know the dimension of an array and the number of its elements?
4. How do I create arrays filled with zeros, ones, and random numbers?
5. How are NumPy arrays indexed and truncated?
6. What is vectorization and why is it important?
7. How do I change the shape of an array (reshape) and when do I need it?
8. What functions are used to combine and split arrays?
9. How do I perform logical filtering and masking of data in an array?
10. What are the advantages of using NumPy compared to standard Python lists?

PRACTICAL WORK №3

Topic: Matplotlib Basics

Purpose of the work:

Learn basic data visualization tools using the Matplotlib library, learn how to create line graphs, bar charts, scatterplots, and customize graph elements to visualize information.

Theoretical substantiation:

The Matplotlib library is a standard tool for data visualization in Python and is actively used in data analysis and machine learning tasks. The main module of the library is `pyplot` – provides an interface similar to the graphical capabilities of MATLAB and allows you to quickly build graphs of various types.

Visualization is a critical step in data analysis. Graphs allow you to identify trends, outliers, patterns, and errors in data. They also serve as a means of presenting the results of building models, comparing predictions, and interpreting relationships between features.

With Matplotlib, you can build line graphs, bar charts, bar charts, pie charts, scatter plots, customize headers, axis labels, legends, and line styles. In addition, the library allows you to save graphs in various formats and integrate them into reports and presentations.

Visualization skills are especially important for communicating results in scientific, educational and applied activities.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed Library `matplotlib` (as part of Anaconda or installed via pip)

Additional libraries: `numpy`, `pandas` for data preparation

Sample data: Manually created or from `sklearn.datasets`

Typical tasks:

1. Plot the $y = \sin(x)$ line graph between -2π and 2π . Adjust the color, line style, axis labels, and header.
2. Plot a histogram of the distribution of random numbers generated by the `np.random.randn()` function. Adjust the number of intervals.
3. Build a scatter plot for random points and add a color map depending on the third variable.

Control questions:

1. What is the role of the Matplotlib library in data analysis?
2. What is the difference between a line plot and a scatter plot?
3. What functions of the `pyplot` module are used for plotting?
4. How do I change the color and style of a line on a chart?
5. How do I set the title and labels for the axes of the chart?
6. How are multiple charts displayed in a single image?
7. How is the histogram constructed and what does the `bins` parameter mean?
8. What are `figure` and `subplot` in the context of Matplotlib?
9. How do I save a chart to a file?
10. What types of graphs are most suitable for visualizing relationships between features?

PRACTICAL WORK №4

Topic: Introduction to pandas: Object Series

Purpose of the work:

Learn the structure and properties of the pandas Series object, master the basic operations of creating, indexing, filtering, and applying aggregate functions to data.

Theoretical substantiation:

The pandas library is one of the key Python libraries for processing tabular and structured data. It provides two main types of objects, Series and DataFrame. A Series object is a one-dimensional array with labels (indexes), which makes it convenient for storing and analyzing ordered sequences of data.

Series can contain values of various types, including numeric, string, and boolean. It is tightly integrated with the NumPy library and supports vector operations, making it an effective tool for performing common data analysis tasks.

Working with a Series involves creating manually and from other data structures (lists, dictionaries, NumPy arrays), accessing items by index or mask, applying arithmetic operations, using aggregation functions, sorting, and checking for missing values. Understanding the structure and behavior of a Series object is a necessary step before learning the structure of a DataFrame, a more complex and powerful tool for working with tabular data.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed Library pandas (в составе Anaconda или через pip)

Optional: Libraries numpy and matplotlib

Source data — manually defined or user-generated lists, dictionaries, arrays

Typical tasks:

1. Create a Series object from a list, a dictionary, and a NumPy array. Display the values, indexes, and data type.
2. Apply the `head()`, `tail()`, `describe()` and arithmetic methods to the Series.
3. Perform logical filtering of the Series: select the items that meet the specified condition.

Control questions:

1. What is a Series object and how is it different from a regular Python list?
2. What ways do you know how to create a Series?
3. How do I access a Series item by index and by value?
4. How does the index change when creating a Series from a dictionary?
5. What happens when I perform arithmetic operations on Series?
6. What methods are used to obtain statistics and describe data?
7. How are values filtered in Series?
8. What is masking and logical indexing?
9. How do I check for missing values in Series?
10. What are the advantages of working with Series compared to Python lists?

PRACTICAL WORK No5

Topic: Introduction to pandas: DataFrame object

Purpose of the work:

Master the basic techniques of creating and using a pandas DataFrame object, learn how to perform operations for viewing, selecting, filtering, sorting, and aggregating data.

Theoretical substantiation:

DataFrame Object The main structural element of the pandas library is a two-dimensional table with named rows (indexes) and columns. It is designed to represent and analyze tabular data, similar to tables in relational databases or spreadsheets.

You can create a DataFrame from a variety of sources: dictionaries, lists, NumPy arrays, CSV files, and other formats. Each column in the DataFrame has a corresponding Series object, which allows you to apply familiar methods and operations to all columns and rows.

DataFrame provides extensive capabilities for filtering data by conditions, accessing rows and columns by labels or positions, changing the structure of a table, handling missing values, calculating aggregates, and performing groupings. The ability to work effectively with DataFrame is a prerequisite for further study of data analysis, visualization, and machine learning.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `pandas`, `numpy`, `matplotlib`

Example: CSV file (provided by the instructor or created by the teacher)

Optional: built-in datasets `pandas` or `sklearn.datasets`

Typical tasks:

1. Create a DataFrame from a list dictionary. Display its dimension, column names, and the first 5 rows.
2. Upload a CSV file using `pd.read_csv()` and perform a basic analysis: data types, omissions, statistics (`describe()`).
3. Select all rows where the value in the specified numeric column exceeds the threshold. Sort the table by this column in descending order.

Control questions:

1. What is a DataFrame and how is it different from a Series?
2. How can I create a DataFrame manually?
3. How do I load data into a DataFrame from an external file?
4. How do I access a single column and row of a DataFrame?
5. How do I find out the dimensions, column names, and data types in a table?
6. What methods are used to filter rows by condition?
7. How is data sorted in a DataFrame?
8. What functions are used to analyze the statistical characteristics of the data?
9. How are missing values checked and handled?
10. What is row indexing and why can it be changed?

PRACTICAL WORK No6

Topic: Fundamentals of exploratory data analysis

Purpose of the work:

Learn how to apply basic methods of exploratory data analysis (EDA) using the pandas library, visualize the data structure, and identify the main patterns, gaps, and anomalies in the dataset.

Theoretical substantiation:

Exploratory data analysis (EDA) is the first and most important step in working with data, aimed at understanding the structure, properties, and features of a data set. The goal of EDA is to

identify relationships between features, test hypotheses, determine the level of data quality, and form an idea of the next stages of analysis or model building.

EDA uses a variety of tools, such as data type and gap analysis, descriptive statistics, histograms, scatterplots, boxplots, correlation heatmaps, and more.

Exploratory analysis tools are the functions of the `pandas`, `matplotlib`, `seaborn`, and other libraries. Conducting an EDA increases the validity of the choice of features, methods, and models used in the subsequent stages of machine learning.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `pandas`, `matplotlib`, `seaborn`, `numpy`

A CSV file with real or synthetic data (for example, `titanic.csv`, `iris.csv`)

Additionally: built-in datasets from `seaborn` (e.g., `tips`, `penguins`)

Typical tasks:

1. Upload a dataset (for example, `titanic.csv`), determine its dimensions, types of features, the presence of gaps, and statistical characteristics.
2. Build visualizations: histograms for quantitative variables, boxplot for categorical variables, and scatterplot for feature pairs.
3. Build a heat map of the correlation of traits and conclude that there are linear relationships between them.

Control questions:

1. What is EDA and what is its role in data analysis?
2. What methods can be used to determine if there are missing values?
3. What statistical measures describe the distribution of features?
4. How to visualize the distribution of a quantitative variable?
5. How do you interpret a boxplot diagram?
6. What are outliers and how can they be detected visually?
7. How to study the relationship between two quantitative characteristics?
8. What types of correlation exist and how to interpret them?
9. Why Use a Heat Map and How Is It Built?
10. How can EDA identify errors in the data structure?

PRACTICAL WORK No7

Topic: Linear Regression

Purpose of the work:

Master the basic principles of building a linear regression model using the library `scikit-learn`, learn how to interpret the coefficients of the model and evaluate its quality using appropriate metrics.

Theoretical substantiation:

Linear regression is one of the simplest and most common machine learning techniques. It is used to model the relationship between one or more explanatory variables (features) and a dependent variable (target variable), assuming that the relationship is linear.

The linear regression model looks like this:

$$y = w_0 + w_1x_1 + w_2x_2 + \dots + w_nx_n,$$

where w – model coefficients determined in the process of training.

The main stages of building a model are: dividing the sample into training and test parts, training the model on training data, predicting the values of the target variable on test data, and estimating the accuracy of the prediction. The main metrics for assessing the quality of the model are the standard error (MSE), the mean absolute error (MAE), and the coefficient of determination (R^2).

Linear regression is widely used in forecasting, trend analysis, and interpreting relationships between variables. It is important to be able to interpret the coefficients of a model, assess its generalizability, and identify possible problems such as multicollinearity or overfitting.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: pandas, numpy, scikit-learn, matplotlib, seaborn

A dataset with numeric features and a continuous target variable (for example, Boston, California housing, insurance.csv)

Typical tasks:

1. Upload the dataset, highlight the features and the target variable, and divide the data into training and test samples.
2. Build a linear regression model using `LinearRegression`, derive the coefficients, and interpret them.
3. Evaluate model quality with MSE, MAE, and R^2 metrics, and visualize predictions on test data.

Control questions:

1. What does linear regression model and what is its main assumption?
2. How are linear regression coefficients interpreted?
3. What are the steps involved in building a linear regression model?
4. How is the data divided into training and test samples?
5. What are MSE, MAE and R^2 and how are they interpreted?
6. What does the coefficient of determination R^2 show?
7. How to visually assess the quality of a regression model?
8. What signs might indicate that the model is being overfitted?
9. What are the advantages and limitations of linear regression?
10. How can the model be improved if there is a weak linear dependence?

PRACTICAL WORK No8

Topic: Logistic regression

Purpose of the work:

To study the principles of building a logistic regression model for binary classification problems,

to learn how to train a model using the library `scikit-learn`, interpret the coefficients and evaluate its quality using the accuracy metric.

Theoretical substantiation:

Logistic regression is a basic classification algorithm used to predict the probability of an object belonging to one of two classes. The model uses a logistic (sigmoid) function that converts a linear combination of input features into a value from 0 to 1, which is interpreted as the probability of assigning an object to a positive class.

The decisive rule of the model is based on a threshold value: if the probability exceeds a given threshold (the default is 0.5), the object is classified as positive, otherwise as negative. The coefficients of the model reflect the influence of the corresponding features on the logarithm of the odds ratio (log-odds).

In the process of training, the model selects coefficients, maximizing the logarithmic probability function. After training, the model can be used to predict classes of new observations. The main quality criterion in this work is accuracy, which reflects the proportion of correctly classified objects.

Logistic regression is widely used in a variety of applications, including medical diagnostics, credit scoring, and marketing, due to its interpretability and effectiveness in small and medium-sized samples.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `pandas`, `numpy`, `scikit-learn`, `matplotlib`, `seaborn`

A dataset for binary classification (e.g., `breast_cancer` of `sklearn.datasets`, `titanic.csv`, `diabetes.csv`)

Typical tasks:

1. Load the dataset and highlight the features and the target variable. Divide the data into training and test samples.
2. Build a logistic regression model with `LogisticRegression`, train it, and derive coefficients.
3. Get predictions on a test sample and calculate the accuracy of the model using the `accuracy_score` metric.

Control questions:

1. What is the main idea of logistic regression?
2. How does the sigmoid function work and why is it used?
3. What data is suitable for logistic regression?
4. How are the coefficients of the logistic regression model interpreted?
5. What does the 0.5 classification threshold mean and how can it be changed?
6. How is the prediction of a logistics model formed?
7. What is accuracy and how is it calculated?
8. Why is it important to divide data into training and test samples?
9. How does feature scaling affect logistic regression?

10. What are the limitations of logistic regression when working with real data?

PRACTICAL WORK №9

Topic: Quality Metrics of Classification Models

Purpose of the work:

Get acquainted with the basic metrics for assessing the quality of classification models, learn how to calculate and interpret accuracy, completeness, prediction accuracy, F1 measure and build a confusion matrix using the library `scikit-learn`.

Theoretical substantiation:

Assessing the quality of classification models is essential to understanding how well the model performs on the task of predicting classes. Using only one metric, such as accuracy, may not be enough, especially in problems with unbalanced classes, where most features belong to a single class.

The following metrics are used to comprehensively assess the quality of classification:

- **Accuracy** is the proportion of correct predictions out of the total number of observations.
- **Recall** is the proportion of objects of a positive class that are correctly predicted by the model.
- **Prediction accuracy (precision)** is the proportion of correct positive predictions among all predictions of a positive class.
- **The F1 measure** is the harmonic mean between precision and recall, used when you need to balance both metrics.
- **A confusion matrix** is a table showing the ratio of true and false positive and negative classifications.

Understanding and correctly interpreting these metrics allows you to objectively compare models and make informed choices about the appropriate algorithm.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `pandas`, `numpy`, `scikit-learn`, `matplotlib`, `seaborn`

A dataset for binary classification (e.g., `breast_cancer`, `titanic.csv`, `diabetes.csv`)

Typical tasks:

1. Train a logistic regression or decision tree model on a binary dataset and get predictions.
2. Build a confusion matrix and visualize it with `seaborn.heatmap()`.
3. Calculate and analyze the accuracy, precision, recall, and F1 measures using `classification_report()`.

Control questions:

1. What is a confusion matrix and what values does it show?
2. How is accuracy calculated?
3. What is the difference between precision and recall?
4. What does the F1 measure measure and when is it particularly useful?
5. Why Can Accuracy Be Insufficient Metric With Unbalanced Classes?
6. How do you interpret a high precision value and a low recall value?
7. What does the `sklearn.metrics.classification_report()` show?
8. How to visualize a confusion matrix?
9. Which metric is more important — precision or recall — and what does it depend on?

10. How do I compare two models across multiple metrics at the same time?

PRACTICAL WORK No10

Topic: Nearest Neighbor Method and Support Vector Method

Purpose of the work:

To study the principles of operation of the k-nearest neighbors algorithm (kNN) and the support vector method (SVM), to learn how to apply them to classification problems using the library `scikit-learn`, adjust model parameters, and evaluate their accuracy.

Theoretical substantiation:

The k-Nearest Neighbors method is one of the lazy training algorithms: the model is not built in advance, and the prediction is made based on the distance to the nearest objects of the training sample. The algorithm is easy to implement, does not require training, but is sensitive to scaling features and choosing the number of neighbors.

The Support Vector Machine (SVM) is one of the more complex algorithms. It looks for the hyperplane that separates the classes as much as possible, taking into account the maximum possible margin. Using kernel functions, the method can solve both linear and nonlinear classification problems. SVM is sensitive to parameter selection, especially `C` and `kernel` type, and requires data scaling first.

Both methods are used in classification and recognition tasks, work well on tasks with an average number of features and observations, but require careful configuration and evaluation.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `scikit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`

Datasets: `iris`, `digits`, `wine` of `sklearn.datasets` or other set for classification

Typical tasks:

1. Load the selected dataset and scale features using `StandardScaler`.
2. Build a `KNeighborsClassifier` model, train it, select the value of `k` (the number of neighbors), calculate the accuracy.
3. Build an `SVC` model with different cores (`linear`, `rbf`), adjust the `C` parameter, and compare the accuracy of the models.

Control questions:

1. How does the k-nearest neighbor method work?
2. What does the `k` parameter mean and how does it affect the behavior of the model?
3. Why is data scaling important for kNN and SVM?
4. How does the support vector method work and what is a separating hyperplane?
5. What is a kernel in SVM and what types of kernels do you know?
6. How does the `C` parameter affect the generalizability of an SVM model?
7. When is it preferable to use kNN and when is it preferable to use SVM?
8. How to evaluate the quality of models on a test sample?
9. What features does kNN have in terms of performance?
10. How to interpret class boundaries derived from the SVM model?

PRACTICAL WORK No11

Topic: Decision trees

Purpose of the work:

Master the basic principles of building and analyzing classification and regression models based on a decision tree, learn how to visualize the tree, interpret the structure of the model and evaluate its accuracy.

Theoretical substantiation:

Decision trees are interpretable and illustrative models used for both classification and regression problems. The main idea of the method is to sequentially divide the feature space into areas in which a prediction is made based on the prevailing class or mean.

During the construction process, the tree selects features and threshold values that provide the best division of the sample by the target feature. The main criteria for the quality of splits are the Gini index, entropy, and variance. The depth of the tree, the minimum number of objects in the node, and other parameters regulate the complexity of the model and avoid overfitting.

The tree model is easy to visualize and interpret: each path from root to leaf can be interpreted as a set of logical conditions that determine the solution. However, trees are prone to overfitting, especially in small samples, and are sensitive to noise.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `scikit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`, `graphviz` (optional)

Datasets: `iris`, `titanic`, `wine` or any other set with categorical and numerical features

Typical tasks:

1. Build a classification model using the `DecisionTreeClassifier`, train it, and derive the accuracy on the test sample.
2. Visualize the structure of the tree using `plot_tree()` or `export_graphviz()`, analyze the features used and the levels of the tree.
3. Examine the impact of the tree depth (`max_depth`) constraint on model quality.

Control questions:

1. How does the algorithm for building a decision tree work?
2. What criteria are used to evaluate data partitioning in tree nodes?
3. What is retraining and how does it manifest itself in decision trees?
4. How to interpret the structure of the tree?
5. What does tree depth mean and how does it affect the model?
6. What settings can control the complexity of the tree?
7. Why are decision trees sensitive to small changes in data?
8. What is the difference between `DecisionTreeClassifier` and `DecisionTreeRegressor`?
9. How to visualize a tree and why is it necessary?
10. What are the advantages and limitations of decision trees compared to other models?

PRACTICAL WORK №12

Theme: Random Forest

Purpose of the work:

To study the principle of operation of the ensemble method "random forest", to learn how to build classification and regression models using `RandomForestClassifier` and `RandomForestRegressor`, interpret the importance of features, and evaluate the accuracy of the model.

Theoretical substantiation:

Random Forest is an ensemble algorithm based on the construction of a set of decision trees, each of which is trained on a random subset of the initial data and using a random subset of features. The final prediction is formed based on voting (for classification) or averaging (for regression) across all trees.

The algorithm combines the simplicity and interpretability of decision trees with high stability and generalizability, reducing the likelihood of overfitting. It works well on tasks with a large number of features and observations, does not require data scaling, and can be used for both classification and regression.

One of the useful properties of a random forest is the ability to evaluate the significance of features based on their participation in splits within trees. This allows for feature selection and simplification of models without significant loss of accuracy.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `scikit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`

Datasets: `breast_cancer`, `wine`, `titanic`, `diabetes`, or any CSV file with class labels

Typical tasks:

1. Train the `RandomForestClassifier` model on the selected dataset and calculate the accuracy on the test sample.
2. Graph the importance of traits using `feature_importances_` and visualize it.
3. Examine the impact of tree count (`n_estimators`) and depth (`max_depth`) on model quality.

Control questions:

1. What is the idea behind the random forest method?
2. How is a random forest different from a single decision tree?
3. How does the training of individual trees in a random forest work?
4. What does parameter `n_estimators` mean?
5. What features are selected during the training of each tree?
6. How is the final prediction calculated in a random forest?
7. How is the importance of features in a model determined?
8. What parameters of a random forest affect retraining?
9. What advantages does a random forest have over other classification models?
10. What tasks is a random forest particularly effective at?

PRACTICAL WORK №13**Topic:** Gradient boosting**Purpose of the work:**

To learn the principles of the gradient boosting method, to learn how to train a model using `GradientBoostingClassifier` from the library `scikit-learn`, evaluate accuracy, and analyze the impact of model parameters on model quality.

Theoretical substantiation:

Gradient boosting is an ensemble technique that builds a sequence of weak models (usually decision trees), each of which learns from the errors of the previous one. Each successive tree adjusts the model's predictions in an effort to minimize a given loss function.

A distinctive feature of gradient boosting is its high accuracy and ability to identify complex nonlinear dependencies. The algorithm is sensitive to parameter selection, such as tree count (`n_estimators`), learning rate (`learning_rate`), and tree depth (`max_depth`). When properly tuned, it can outperform simpler models, but requires regular validation to prevent overfitting.

Gradient boosting is used in a wide range of tasks, including credit scoring, customer churn analysis, medical diagnostics, and machine learning competitions.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `scikit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`

Dataset: `breast_cancer`, `titanic`, `wine`, or your own CSV file for classification

Typical tasks:

1. Build the `GradientBoostingClassifier` model on the selected dataset, train it, and calculate the accuracy on the test set.
2. Change the values of the `n_estimators`, `learning_rate`, and `max_depth` parameters, and analyze how they affect the accuracy of the model.
3. Plot the importance of traits and infer significant variables.

Control questions:

1. How does gradient boosting work?
2. How is gradient boosting different from random forest?
3. What does the `learning_rate` parameter mean and how does it affect the model?
4. How does `n_estimators` affect overfitting and underfitting?
5. What is the role of trees in gradient boosting?
6. What loss functions can be used in gradient boosting?
7. What is the role of regularization in boosting?
8. What does `feature_importances_` show in a boosting model?
9. How to determine the optimal parameters of the model?
10. In which tasks can gradient boosting be preferable to other methods?

PRACTICAL WORK №14

Topic: Unsupervised Learning: Data Clustering Tasks

Purpose of the work:

Get acquainted with the methods of data clustering, master the work with algorithms `KMeans` and `AgglomerativeClustering` from the library `scikit-learn`, learn to visualize the results of clustering and interpret the structure of the resulting clusters.

Theoretical substantiation:

Unsupervised learning is used in situations where there are no pre-marked classes. Clustering, one of the key approaches in this type of learning, consists in dividing objects into groups (clusters) according to the similarity of features without prior knowledge of the true class labels.

The **k-means method (KMeans)** splits the data into a specified number of clusters, minimizing the intra-cluster distance. This method is simple, interpretable, and scales well, but requires you to determine the number of clusters in advance.

The AgglomerativeClustering **method** builds a tree-like structure for combining clusters, which allows you to explore data at different levels of the hierarchy. Clusters are formed by sequentially combining nearby objects or groups of objects.

Clustering is used in user segmentation, anomaly search, bioinformatics, text analysis, and other tasks where it is important to identify the internal structure of the data.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `scikit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`

Datasets: `iris`, `wine`, synthetic samples from `sklearn.datasets.make_blobs` or `make_moons`

Typical tasks:

1. Perform clustering with `KMeans` on the selected dataset, visualize the clusters, and compare them to the true labels (if known).
2. Build a hierarchical clustering using `AgglomerativeClustering`, display a dendrogram, and interpret the number of clusters.
3. Perform dimensionality reduction using PCA to visualize the clustering results in 2D.

Control questions:

1. How is unsupervised learning different from supervised learning?
2. What is the main idea of the k-means method?
3. How do I choose the optimal number of clusters in KMeans?
4. How does hierarchical clustering work?
5. What is a dendrogram and how to interpret it?
6. What measures are used to assess the quality of clustering?
7. Why are clustering results sensitive to feature scale?
8. What is intra-cluster distance and how is it used in KMeans?
9. How does the initial initialization of cluster hubs affect the final result?
10. What real problems can be solved by clustering methods?

PRACTICAL WORK No15

Topic: Unsupervised learning: data dimensionality reduction problems

Purpose of the work:

Study methods for reducing the dimensionality of data, learn how to apply algorithms PCA (principal component analysis) and t-SNE to simplify the data structure and visualize multidimensional feature spaces.

Theoretical substantiation:

Dimensionality reduction tasks aim to represent data with a large number of features in a lower-dimensional space while preserving as much of the information as possible. This simplifies post-analysis, visualizes data, reduces noise and computational complexity, and prepares features for machine learning.

PCA (**Principal Component Analysis**) is a linear algorithm based on the transformation of features into a new orthogonal base, ordered in descending order of variance. PCA is used for data compression, multicollinearity elimination, and 2D/3D visualization.

The t-distributed Stochastic Neighbor Embedding (t-SNE) **method** is a nonlinear algorithm focused on preserving the local data structure. It is often used to visualize clusters and the structure of complex multidimensional spaces, especially when working with high-dimensional and poorly interpreted data.

Dimensionality reduction techniques play an important role in the preliminary analysis of data, helping to identify hidden dependencies and facilitate the interpretation of models.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `scikit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`

Datasets: `iris`, `digits`, `wine`, `make_blobs` or custom CSV files

Typical tasks:

1. Perform PCA on the selected dataset, determine the number of components that retain $\geq 90\%$ of the variance, and visualize the data in 2D space.
2. Run t-SNE on the same dataset, compare the distribution of points on the chart with the PCA result.
3. Plot the explained variance for the first components of PCA and comment on the result.

Control questions:

1. What is the purpose of dimensionality reduction?
2. How does the PCA algorithm work?
3. What are the principal components and how are they interpreted?
4. How do I determine how many components to keep when using PCA?
5. What is the difference between linear PCA and nonlinear t-SNE?
6. What t-SNE parameters affect the result of the visualization?
7. How Can Post-PCA Visualization Help Understand Data Structure?
8. In which problems is dimensionality reduction appropriate?
9. What are the limitations of the PCA method?
10. Why use dimensionality reduction techniques before clustering or classification?

PRACTICAL WORK №16

Topic: Selection of hyperparameters of models

Purpose of the work:

Learn how to hyperparameter tune machine learning models using mesh search techniques (`GridSearchCV`) and random search (`RandomizedSearchCV`), to master the use of cross-validation to improve the reliability of model quality assessment.

Theoretical substantiation:

Hyperparameters are model parameters whose value is not determined during training, but is set manually before training. These include, for example, the number of neighbors in `KNeighborsClassifier`, the depth of the decision tree in `DecisionTreeClassifier`, the regularization parameter `C` in logistic regression, etc.

The correct choice of hyperparameters significantly affects the generalizing ability of the model. One of the approaches to their selection is the use of **cross-validation**, in which the training sample is divided into several parts, and the model is evaluated on each of them.

For automated parameter selection, the following are used:

- `GridSearchCV` — a complete search of all possible combinations of specified parameter values;

- `RandomizedSearchCV` is a random search for a specified number of random combinations.

Both procedures allow you to find the best model configuration based on a given quality metric, usually accuracy. This approach increases the stability of the model and helps avoid overfitting.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook или Google Colab

Installed libraries: `scikit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`

Datasets: `iris`, `wine`, `titanic`, or any CSV file for classification

Typical tasks:

1. Configure the parameters of the `KNeighborsClassifier` model using `GridSearchCV`, determine the optimal value of `n_neighbors`.
2. Apply `RandomizedSearchCV` to the decision tree model, `max_depth` and `min_samples_split`, and analyze the result.
3. Compare the accuracy of the model with the optimal parameters and the model with the default parameters.

Control questions:

1. What are hyperparameters and how do they differ from model parameters?
2. Why do hyperparameter bringing?
3. What is the difference between `GridSearchCV` and `RandomizedSearchCV`?
4. What is cross-validation and how does it work?
5. How do I choose the ranges of values to iterate with?
6. What metrics can be used when selecting hyperparameters?
7. How to use the resulting model after selecting the parameters?
8. What are the risks associated with over-selecting hyperparameters?
9. What is parameterization overfitting and how can I avoid it?
10. What are the ways to speed up the selection of hyperparameters?

CONCLUSION

The practical works presented in these guidelines cover the key areas of the discipline **"Statistical Methods in Machine Learning"** and are aimed at developing students' stable skills in building, configuring and analyzing machine learning models using modern tools and libraries of the Python language.

The structure of the work is built on increasing complexity: from mastering the basic tools of the software environment (Jupyter Notebook, NumPy, pandas, matplotlib) to the practical application of supervised and unsupervised learning algorithms, including methods of classification, regression, ensemble learning, clustering, and dimensionality reduction. The final classes are devoted to improving the efficiency of models, such as the selection of quality metrics and the selection of hyperparameters.

Each practical work contains a goal, theoretical justification, a list of necessary software and data, typical tasks to perform, as well as control questions for self-examination and current control.

Systematic performance of practical tasks contributes to: – consolidation of theoretical knowledge; – formation of applied competencies in data analysis and interpretation; – development of analytical thinking and the ability to work with real problems; – preparation for independent professional and research activities in the field of data analysis and artificial intelligence.

The practical part of the course is focused on the use of open and accessible software tools, which allows you to ensure the flexibility of learning and the applicability of the knowledge gained in a variety of fields.

EDUCATIONAL, METHODOLOGICAL AND INFORMATION SUPPORT OF THE DISCIPLINE

Reference citations:

Bolotsky, A. V.; Investigation of operations and optimization methods Electronic resource / Bolotsky A. V., Kochetkova O. A. - St. Petersburg : Lan, 2020. - 116 p. - ISBN 978-5-8114-4568-4

Saraev, P. V.; Methods of Machine Learning Electronic resource: Guidelines and tasks for laboratory work on the course / P. V. Saraev. - Lipetsk : Lipetsk State Technical University, EBS ASV, 2017. - 48 p. - The book is in the premium version of EBS IPR BOOKS. - ISBN 2227-8397

Sopov, E. A.; Multi-Criteria Neuroevolutionary Systems in Machine Learning and Human-Machine Interaction Electronic resource / Sopov E. A., Ivanov I. A. : monograph. - Krasnoyarsk : SibFU, 2019. - 160 p. - ISBN 978-5-7638-3969-2

Further reading:

Methodical recommendations for practical classes in the discipline "Theory and Practice of Machine Translation": Direction of training 45.04.02 Linguistics. Master's program "Translation and Translation Studies". Qualification of the graduate - master. Form of study - full-time. Curriculum of 2015. Studied in the 3rd semester. - Stavropol : NCFU, 2015. - 9 p.

Nedelko, V. M.; Fundamentals of Statistical Methods of Machine Learning Electronic resource: Textbook / V. M. Nedelko. - Novosibirsk: Novosibirsk State Technical University, 2010. - 72 p. - The book is in the premium version of EBS IPR BOOKS. - ISBN 978-5-7782-1385-2

Rapakov, G. G.; Methods and algorithms of machine learning in making management decisions in the regional system of medical prevention (experience of the Vologda region) Elektronnyi resurs / Rapakov G. G., Kasimov R. A. : monograph. - Vologda : VoSU, 2014. - 143 p. - ISBN 978-5-87851-548-1

Educational and methodical literature:

Saraev, P. V.; Methods of Machine Learning Electronic resource: Guidelines and tasks for laboratory work on the course / P. V. Saraev. - Lipetsk : Lipetsk State Technical University, EBS ASV, 2017. - 48 p. - The book is in the premium version of EBS IPR BOOKS. - ISBN 2227-8397