

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

GUIDELINES
for the completion of practical work
in the discipline "
for students in the field of study
02.04.01 Mathematics and Computer Science
Directionality (profile)
"Training Program for Electronic Information"

Content

Introduction

Practical Work #1: Basics of Working with Jupyter Notebook , JupyterLab and Google Colab

Practical Work #2: Basics of Working with the NumPy Package

Practical Work #3 Basics of Working with the Matplotlib Package

Practical Work #4 Introduction to pandas : Series Object

Practical Work #5 Introduction to pandas : The DataFrame Object

Practical Work #6: Fundamentals of Exploratory Data Analysis

Practical Work #7 Linear Regression

Practical Work #8 Logistic Regression

Practical Work #9: Quality Metrics for Classification Models

Practical Work #10: Nearest Neighbor Method and Support Vector Machine

Practical Work #11 Decision Trees

Practical Work #12: Random Forest

Practical Work #13 Gradient Boosting

Practical Work #14 Unsupervised Learning: Data Clustering Problems

Practical Work #15 Unsupervised Learning: Data Dimensionality Reduction Problems

Practical Work #16 Selection of Model Hyperparameters

Conclusion

Educational, methodological and informational support of the discipline

INTRODUCTION

1. The purpose and objectives of mastering the discipline (module)

The purpose of the discipline is to develop, in accordance with the educational program, the competencies of OPC-1, OPC-3 of the student.

To achieve this goal, the following objectives are addressed in the course: familiarization with the basic concepts and types of machine learning problems; study of data preparation methods, including cleaning, normalization, and feature engineering; mastering basic supervised learning models, such as linear and logistic regression; mastering stochastic gradient descent as a method for optimizing models; forming an understanding of the metrics for assessing the quality of classification and interpreting 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 machine; study of algorithms for constructing decision trees and ensemble models, such as random forest and gradient boosting; familiarity with unsupervised learning methods, including clustering algorithms; mastering dimensionality reduction and data visualization methods, such as PCA and t-SNE; developing practical skills in tuning models using hyperparameter selection; developing the ability to compare and interpret the results of different models based on objective metrics.

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

The discipline is a compulsory part.

3. List of planned learning outcomes for the discipline (module), correlated with the planned learning outcomes for the educational program

Code, formulation of competence	Code, indicator formulation	Planned learning outcomes for the discipline (module), characterizing the stages of formation of competencies, indicators
OPK-1 Capable of finding, formulating and solving current and significant problems in applied and computer mathematics	ID-1 _{OPK-1} Finds and formulates significant problems of applied and computer mathematics.	Identifying and formulating significant problems in applied and computer mathematics that arise during data analysis and the development of machine learning systems; analyzing applied problems for their mathematical formulation; defining mathematical models necessary for solving problems of classification, regression, clustering, and data dimensionality reduction; identifying the limitations and specific features of mathematical methods used in machine learning; substantiating the relevance and practical significance of

		the mathematical problems under consideration; formulating problems for subsequent software implementation in Python .
	ID-2 _{OPK-1} knows and uses methods for solving significant problems of applied and computer mathematics.	Using applied and computer mathematics methods to solve data analysis and machine learning problems; selecting and applying mathematical methods to solve problems of classification, regression, clustering, and data dimensionality reduction; applying methods of linear algebra, mathematical statistics, and probability theory to construct machine learning models; analyzing and interpreting calculation and experimental results; assessing the correctness and effectiveness of the mathematical methods used; implementing mathematical models and algorithms using Python and specialized machine learning libraries.
OPK-3. Capable of carrying out conceptual, functional, and logical design of systems of medium and large scale and complexity.	ID-1 _{OPK-3} Independently creates application software products.	Independent development of applied software products for solving data analysis and machine learning problems in Python ; designing the structure of a software solution taking into account the assigned task; implementing algorithms for data processing, training, and evaluating machine learning models; using specialized Python libraries for data analysis, visualization, and machine learning; testing and debugging software components; analyzing and interpreting the results of the developed software product; documenting the software solution and preparing it for practical use.

	ID-2 ОПК-3 Knows and can find modern information technologies, including domestically produced ones, necessary for the creation of applied software products.	Search, analysis, and selection of modern information technologies, software platforms, libraries, and network resources necessary for the development of applied software products in the field of machine learning; assessment of the capabilities and limitations of modern software tools, including domestic software solutions; selection of development tools, data analysis tools, and machine learning libraries in accordance with the requirements of the problem being solved; use of technical documentation, professional communities, and specialized information resources to find technological solutions; justification of the choice of software tools and technologies for the implementation of applied software products; application of the selected technologies in the development and maintenance of software solutions in the Python language .
--	--	---

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

Volume of classes: total: 4 credits , 144 academic hours .	OFO, In academic hours
Contact work:	32.00/0
Lectures/including practical training	0/0
Laboratory work /including practical training	
Practical classes /including practical training	32.00/0
Independent work	58.00
Control	54.00
Forms of control	
Exam	Yes

5. Importance practical work students

Practical work of students is a central element of mastering the discipline " *Machine Learning Programming Design Based on Python / Разработка программного обеспечения для машинного обучения на основе Python* ». With the rapid development of technology and the highly applied nature of this field, theoretical knowledge without practical application loses its relevance and depth.

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

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

Practical work also develops important competencies in demand in the professional environment: the ability to independently find solutions, work with open data, tune model parameters, and evaluate their effectiveness in real-world conditions. Regularly completing practical assignments increases students' motivation, builds self-confidence, and prepares them for the application of machine learning in real-world projects and scientific research.

PRACTICAL WORK #1

Topic: Jupyter Basics Notebook, JupyterLab and Google Colab

Objective :

To become familiar with the basics of organizing and conducting interactive calculations in Jupyter Notebook, JupyterLab and Google Colab. Master the basics of entering, formatting, and executing code and text cells, as well as loading and saving work files.

Theoretical background:

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

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

Working in interactive environments allows for efficient model building and debugging, experiment documentation, sharing results, and saving them in a single file. Students should be able to distinguish between cell types (code and text), format text using Markdown, insert LaTeX formulas, and manage data loading and library connections.

Knowledge and fluency in Jupyter Notebook, JupyterLab and Google Colab is the foundation for further successful mastery of the discipline and the implementation of laboratory, project, and scientific tasks.

Equipment and materials:

Computer or laptop with Internet access

Google Browser Chrome or Mozilla Firefox

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

Google account for accessing Google Colab and Google Drive

Sample CSV file (provided by the instructor or taken from sklearn datasets)

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

Typical tasks:

1. Create Jupyter Create a notebook with a brief description of the purpose of the future project and insert code examples demonstrating the execution of arithmetic operations.
2. Open Google Colab, create a new notebook, connect it to your Google Drive and upload the CSV file, then output the first rows of the table.
3. In JupyterLab, create a file containing a text cell with a title, a bulleted list, and a LaTeX formula, and a code cell that implements a simple function.

Test questions:

1. Jupyter Notebook different from JupyterLab?
2. What benefits does Google provide? Colab vs. On-Premise Environments?
3. How to save a notebook on a local computer and in the cloud?
4. What are the main cell types in Jupyter?
5. How to insert formulas and formatted text into a Markdown cell?
6. How do I connect to Google Drive in Colab?
7. What commands are used to install and link libraries?
8. What is "runtime" in Google Colab?
9. How to download data to a laptop and view its contents?
10. What export formats are available for Jupyter Notebook?

PRACTICAL WORK #2

Topic: Basics of working with the NumPy package

Objective :

To 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 for analyzing and processing numerical data.

Theoretical background:

Library NumPy is fundamental to numerical computing in Python and serves as the basis for most tools used in machine learning and data analysis. NumPy's core object is the multidimensional `ndarray`, which provides high performance by storing data compactly and allowing operations on arrays without the use of loops.

Working with arrays allows for efficient representation of data samples, linear algebra, statistical operations, normalization, and other transformations required during data preprocessing. One of NumPy's key features is its ability to perform operations element-by-element and vectorized, significantly accelerating the processing of large volumes of data.

NumPy also includes a set of tools for generating random numbers, transforming arrays, combining and splitting data, and performing logical operations and filtering. A deep understanding of this library's capabilities 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 Local

Jupyter environment Notebook or Google Colab Installed

numpy library (included in Anaconda or installed via pip)

Examples of arrays and assignments prepared by the teacher or created by the student Additional materials: [documentation https://numpy.org/doc](https://numpy.org/doc)

Typical tasks:

1. Create one-dimensional and two-dimensional arrays using `np. array ( )`, print 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 on two compatible arrays using both `@` and `np. dot ( )`.

Test questions:

1. What is the ndarray object in the NumPy library?
2. What is the difference between element-wise multiplication and matrix multiplication?
3. How to find out the dimensions of an array and the number of its elements?
4. How to create arrays filled with zeros, ones, and random numbers?
5. NumPy arrays indexed and sliced?
6. What is vectorization and why is it important?
7. reshape an array and when is it needed?
8. What functions are used to merge and split arrays?
9. How to perform logical filtering and masking of data in an array?
10. What are the advantages of using NumPy over standard Python lists?

PRACTICAL WORK #3

Topic: Basics of working with the Matplotlib package

Objective :

To master the basic tools of data visualization using the Matplotlib library, learn to create line graphs, histograms, scatter plots and customize graph elements for a visual presentation of information.

Theoretical background:

Library Matplotlib is the standard data visualization tool for Python and is widely used in data analysis and machine learning. The library's core module, `pyplot`, provides an interface similar to MATLAB's graphics capabilities and allows for the rapid creation of various types of graphs.

Visualization is a crucial step in data analysis. Graphs help identify trends, outliers, patterns, and errors in data. They also serve as a means of presenting model results, comparing predictions, and interpreting relationships between features.

With Matplotlib, you can create line graphs, histograms, bar charts, pie charts, and scatter plots, as well as customize titles, axis labels, legends, and line styles. The library also 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 or Google Colab Installed

matplotlib library (included in Anaconda or installed via pip)

Additional libraries: numpy, pandas for data preparation

Sample data: manually created or from sklearn dataset

Typical tasks:

1. Plot a linear graph of $y = \sin(x)$ over the interval from -2π to 2π . Customize the color, line style, axis labels, and title.
2. Plot a histogram of the distribution of random numbers generated by the `np.random.randn()` function. Adjust the number of bins.
3. Construct a scatter plot (`scatter plot`) for random points and add a color scale (`color map`) depending on the third variable.

Test questions:

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

PRACTICAL WORK #4

Topic: Introduction to pandas : The Series Object

Objective :

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

Theoretical background:

Library pandas is one of the key Python libraries for processing tabular and structured data. It provides two main object types: Series and DataFrame. The Series object is a one-dimensional array with labels (indices), making it convenient for storing and analyzing ordered data sequences.

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

Equipment and materials

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed pandas library (included in Anaconda or via pip)

Additionally: numpy and matplotlib libraries

Source data - manually specified or user-generated lists, dictionaries, arrays

Typical tasks

1. Create a Series object from a list, dictionary, and NumPy array. Display the values, indices, and data type.
2. Apply head(), tail(), describe() methods and arithmetic operations to Series.
3. Perform logical filtering on a Series: select elements that satisfy a given condition.

Test questions

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

PRACTICAL WORK #5

Topic: Introduction to pandas: The DataFrame Object

Objective:

To master the basic techniques of creating and using the DataFrame object of the pandas library, and to learn how to perform operations on viewing, selecting, filtering, sorting, and aggregating data.

Theoretical justification:

Object A DataFrame, the fundamental building block of the pandas library, is a two-dimensional table with named rows (indices) and columns. It is designed for presenting and analyzing tabular data, similar to tables in relational databases or spreadsheets.

A DataFrame can be created from a variety of sources: dictionaries, lists, NumPy arrays, CSV files, and other formats. Each column in a DataFrame corresponds to a Series object, allowing you to apply familiar methods and operations to all columns and rows.

DataFrames offer extensive capabilities for filtering data by conditions, accessing rows and columns by labels or positions, modifying table structure, handling missing values, calculating aggregates, and performing groupings. Knowing how to effectively work with

DataFrames is essential for further exploration of data analysis, visualization, and machine learning.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: pandas , numpy , matplotlib

Sample CSV file (provided by the teacher or created independently) Additional: built-in datasets pandas or sklearn. datasets

Typical tasks:

1. Create a DataFrame from a dictionary of lists. Display its dimensions, column names, and the first 5 rows.
2. CSV file with `pd.read_csv()` and perform basic analysis: data types, missingness, `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.

Test questions:

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

PRACTICAL WORK #6

Topic: Fundamentals of Exploratory Data Analysis

Objective of the work:

To learn how to apply basic methods of exploratory data analysis (Exploratory Data Analysis (EDA) using the pandas library, visualize the structure of the data and identify key patterns, gaps, and anomalies in the dataset.

Theoretical rationale:

Exploratory data analysis (EDA) is the first and most important step in working with data, aimed at understanding the structure, properties, and characteristics of a dataset. The goal of EDA is to identify relationships between features, test hypotheses, determine the level of data quality, and provide insights into further analysis or model building.

EDA utilizes a variety of tools, including data type and gap analysis, descriptive statistics calculation, and the construction of histograms, scatterplots, boxplots, correlation heatmaps, and more. Particular attention is paid to visualization, as it allows for the visualization of hidden dependencies, outliers, and errors in the data.

Exploratory analysis tools include functions from pandas, matplotlib, seaborn, and other libraries. Conducting EDA improves the validity of the choice of features, methods, and models used in subsequent stages of machine learning.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: pandas , matplotlib , seaborn , numpy

CSV file with real or synthetic data (e.g. titanic.csv , iris.csv)

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

Typical tasks:

1. Load a dataset (e.g. titanic.csv), determine its dimensions, feature types, presence of gaps, and statistical characteristics.
2. Create visualizations: histograms for quantitative variables, boxplots for categorical variables, and scatterplots for feature pairs.
3. Construct a heat map of feature correlation and conclude whether there are linear relationships between them.

Test questions:

1. What is EDA and what is its role in data analysis?
2. What methods can be used to determine the presence of missing values?
3. What statistical measures describe the distribution of features?
4. How to visualize the distribution of a quantitative variable?
5. How to interpret a boxplot chart ?
6. What are outliers and how can they be detected visually?
7. How to investigate 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 detect errors in data structure?

PRACTICAL WORK #7

Topic: Linear Regression

Objective :

To master the basic principles of constructing a linear regression model using the sklearn library , to learn how to interpret model coefficients and to evaluate its quality using appropriate metrics.

Theoretical background:

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

The linear regression model has the form:

$$y = w_0 + w_1 X_1 + w_2 X_2 + \dots + w_n X_n$$

where w are the model coefficients determined during the training process.

The main stages of model construction are: dividing the sample into training and test portions, training the model on the training data, predicting the target variable values on the test data , and assessing the prediction accuracy. The main metrics for assessing model quality are the mean square error (MSE), 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 model's coefficients, evaluate its generalization ability, and identify potential problems such as multicollinearity or overfitting.

Equipment and materials

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: pandas , numpy , sci ki t - l ear n , mat pl ot l i b , seabor n

Dataset with numerical features and a continuous target variable (e.g., Bost on , Cal i f or ni a housi ng , i nsur ance. csv)

Typical tasks

1. Load the dataset , extract features and the target variable, and split the data into training and test sets.
2. Build a linear regression model using Li near Regr essi on , print the coefficients and interpret them.
3. Evaluate model quality using MSE, MAE, and R^2 metrics, and visualize predictions on test data.

Test 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 constructing a linear regression model?
4. How is data divided into training and test samples?
5. What are MSE, MAE and R^2 and how to interpret them?
6. What does the coefficient of determination R^2 show?
7. How to visually assess the quality of a regression model?
8. What are some signs that a model may be overfitting?
9. What are the advantages and limitations of linear regression?
10. How can a model be improved when there is a weak linear relationship?

PRACTICAL WORK #8

Topic: Logistic Regression

Objective :

To study the principles of constructing a logistic regression model for binary classification problems, learn how to train a model using the sci ki t - l ear n l i br ar y , interpret the coefficients and evaluate its quality using the accuracy metric .

Theoretical background:

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, which transforms a linear combination of input features into a value between 0 and 1, which is interpreted as the probability of assigning the object to the positive class.

The model's decision rule is based on a threshold: if the probability exceeds a specified threshold (0.5 by default), the object is classified as positive; otherwise, it is classified as negative. The model's coefficients reflect the influence of the corresponding features on the logarithm of the odds ratio (log-odds).

During training, the model selects coefficients by maximizing the log-likelihood 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 various applications, including medical diagnostics, credit scoring, and marketing, due to its interpretability and performance on small and medium-sized samples.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: pandas, numpy, sci-kit-learn, matplotlib, seaborn

Binary classification dataset

(e.g., breast_cancer from sklearn.datasets, titanic.csv, diabetes.csv)

Typical tasks:

1. Load the dataset and extract features and the target variable. Split the data into training and test sets.
2. Build a logistic regression model using LogisticRegression, train it, and print the coefficients.
3. Obtain predictions on the test set and calculate the model accuracy using the accuracy_score metric.

Test questions:

1. What is the basic 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 a logistic regression model interpreted?
5. What does the classification threshold of 0.5 mean and how can it be changed?
6. How is the logistic model prediction formed?
7. What is accuracy and how is it calculated?
8. Why is it important to split data into training and test sets?
9. How does feature scaling affect the performance of logistic regression?
10. What are the limitations of logistic regression when working with real data?

PRACTICAL WORK #9

Topic: Quality metrics for classification models

Objective :

To become familiar with the main metrics for assessing the quality of classification models, to

learn how to calculate and interpret accuracy, recall, prediction accuracy, F1-measure and to construct confusion matrix using the `sci ki t - l e a r n l i b r a r y` .

Theoretical rationale:

Assessing the quality of classification models is essential for understanding how well the model performs in predicting classes. Using only one metric, such as accuracy , may be insufficient, especially in problems with imbalanced classes, where most objects belong to a single class.

The following metrics are used for a comprehensive assessment of classification quality:

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

Understanding and correctly interpreting these metrics allows for objective comparison of models and informed choice of the appropriate algorithm.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: `pandas` , `numpy` , `sci ki t - l e a r n` , `mat pl ot l i b` , `seabor n`

Binary classification dataset (e.g. `br east _ cancer` , `t i t a n i c . csv` , `di abet es . csv`)

Typical tasks:

1. Train a logistic regression or decision tree model on a binary dataset and obtain predictions.
2. Build confusion matrix and visualize it using `seabor n . heat map ( )` .
3. Calculate and analyze accuracy , precision , recall , and F1 score values using `cl assi f i cat i on _ r e p o r t ( )` .

Test questions:

1. What is 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 score measure and when is it particularly useful?
5. Why might accuracy be a poor metric for imbalanced classes?
6. How to interpret high precision and low recall values ?
7. What shows `cl assi f i cat i on _ r e p o r t ( )` from `skl e a r n . met r i c s` ?
8. How to visualize confusion matrix ?
9. Which metric is more important— precision or recall —and what does it depend on?
10. How to compare two models across multiple metrics at once?

PRACTICAL WORK #10

Topic: Nearest Neighbor and Support Vector Machines

Objective :

To study the operating principles of the k-nearest neighbors (`kNN`) algorithm and the support

vector machine (SVM), learn how to apply them to classification problems using the `sci-kit-learn` library, configure model parameters and evaluate their accuracy.

Theoretical background:

k-nearest neighbors (k-Nearest) method Neighbors is a lazy learning algorithm: the model is not built in advance, and predictions are made based on the distance to the nearest objects in the training set. The algorithm is easy to implement and requires no training, but is sensitive to feature scaling and the choice of the number of neighbors.

Support Vector Machine (Support) Vector SVM (Surface-Model Machine) is a more complex algorithm. It searches for a hyperplane that maximally separates classes, taking into account the maximum possible margin. Using kernel functions, the method can solve both linear and nonlinear classification problems. SVM is sensitive to the choice of parameters, especially C and the kernel type, and requires pre-scaling of the data.

Both methods are used in classification and recognition problems, work well on problems with a moderate number of features and observations, but require careful tuning and evaluation.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: `sci-kit-learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`

Datasets: `iris`, `digits`, `wine` from `sklearn.datasets` or another dataset for classification

Typical tasks:

1. Load the selected dataset, perform feature scaling using `StandardScaler`.
2. Build a `KNeighborsClassifier` model, train it, find the value of k (number of neighbors), calculate the accuracy.
3. Build an `SVC` model with different kernels (`linear`, `rbf`), tune the parameter C and compare the accuracy of the models.

Test questions:

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

PRACTICAL WORK #11

Topic: Decision Trees

Objective :

To master the basic principles of constructing and analyzing classification and regression models based on decision trees, learn to visualize the tree, interpret the structure of the model and evaluate its accuracy.

Theoretical rationale: Decision

trees are interpretable and intuitive models used for both classification and regression problems. The basic idea of the method is to sequentially partition the feature space into regions in which predictions are made based on the dominant class or mean.

During the tree construction process, it selects features and threshold values that provide the best split of the sample based on the target feature. The main criteria for split quality are the Gini index, entropy, and variance. The tree depth, the minimum number of objects per node, and other parameters regulate the model's complexity and prevent 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 a decision. However, trees are prone to overfitting, especially on small samples, and are sensitive to noise.

Equipment and materials

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: `sci-kit-`

`learn`, `pandas`, `numpy`, `matplotlib`, `seaborn`, `graphviz` (optional)

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

Typical tasks

1. Build a classification model using `DecisionTreeClassifier`, train it, and print the accuracy on the test set.
2. Plot the tree structure using `plot_tree()` or `export_graphviz()`, analyze the features used and the tree levels.
3. Investigate the impact of tree depth limitation (`max_depth`) on model quality.

Test questions

1. How does the decision tree algorithm work?
2. What criteria are used to evaluate the partitioning of data in tree nodes?
3. What is overfitting and how does it manifest itself in decision trees?
4. How to interpret tree structure?
5. What does tree depth mean and how does it affect the model?
6. What parameters allow you to control the complexity of the tree?
7. Why are decision trees sensitive to small changes in the data?
8. How is different `DecisionTreeClassifier` from `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

Subject: Random Forest

Objective :

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

Theoretical background:

Random forest (Random Forest is an ensemble algorithm based on constructing multiple

decision trees, each of which is trained on a random subsample of the original data and using a random subset of features. The final prediction is formed by voting (for classification) or averaging (for regression) across all trees.

The algorithm combines the simplicity and interpretability of decision trees with high robustness and generalization ability, reducing the risk of overfitting. It performs well on problems 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 random forests is the ability to evaluate the importance of features based on their participation in splits within the trees. This allows for feature selection and model simplification without significant loss of accuracy.

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: `sci-kit-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, calculate the accuracy on the test set.
2. feature importance plot using `feature_importances_` and visualize it.
3. Investigate the impact of the number of trees (`n_estimators`) and depth (`max_depth`) on the quality of the model.

Test 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 are individual trees trained in a random forest?
4. What does the `n_estimators` parameter mean?
5. What features are selected when training each tree?
6. How is the final prediction calculated in a random forest?
7. How is feature importance determined in a model?
8. What random forest parameters affect overfitting?
9. What advantages does random forest have over other classification models?
10. For what problems is random forest particularly effective?

PRACTICAL WORK #13

Topic: Gradient Boosting

Objective :

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

Theoretical background:

Gradient Boosting is an ensemble method that builds a sequence of weak models (usually decision trees), each learning from the errors of the previous one. Each subsequent tree adjusts the model's predictions, seeking to minimize a given loss function.

A distinctive feature of gradient boosting is its high accuracy and ability to detect complex nonlinear relationships. The algorithm is sensitive to the choice of parameters, such as

the number of trees (`n_estimators`), learning rate (`learning_rate`), and tree depth (`max_depth`). When properly configured, it can outperform simpler models, but requires regular validation to prevent overfitting.

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

Equipment and materials

Computer or laptop with internet access

Jupyter Notebook or 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 a `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 a feature importance graph and draw conclusions about the significant variables.

Test 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 the `n_estimators` parameter 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 the boosting model?
9. How to determine the optimal model parameters?
10. In what problems might gradient boosting be preferable to other methods?

PRACTICAL WORK #14

Topic: Unsupervised Learning: Data Clustering Problems

Objective :

To become familiar with data clustering methods, master the use of the `KMeans` and `AgglomerativeClustering` algorithms from the `scikit-learn` library, learn to visualize clustering results and interpret the structure of the resulting clusters.

Theoretical rationale:

Unsupervised learning is used in situations where pre-labeled classes are unavailable. Clustering, one of the key approaches in this type of learning, involves dividing objects into groups (clusters) based on similar features without prior knowledge of the true class labels.

k-means method (**KMeans**) partitions the data into a given number of clusters, minimizing Intra-cluster distance. This method is simple, interpretable, and scalable, but requires the number of clusters to be determined in advance.

method (**Agglomerative Clustering**) constructs a tree-like structure of cluster associations, allowing for data exploration at different hierarchical levels. Clusters are formed by sequentially combining nearby objects or groups of objects.

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

Equipment and materials:

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: sci ki t - l e a r n , pandas , numpy , mat pl o t l i b , seaborn

Datasets: i r i s , wi ne , synthetic samples

from skl e a r n. dat a s e t s. make _bl o b s or make _moons

Typical tasks:

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

Test questions:

1. How is unsupervised learning different from supervised learning?
2. What is the basic idea of the k-means method?
3. How to 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 initialization of cluster centers affect the final result?
10. What real-world problems can be solved using clustering methods?

PRACTICAL WORK #15

Topic: Unsupervised Learning: Problems of Data Dimensionality Reduction

Objective :

To study methods for reducing data dimensionality, to learn how to apply PCA (principal component analysis) and t - SNE al gor i t hms to simplify the data structure and visualize multidimensional feature spaces.

Theoretical rationale: Dimensionality reduction

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

PCA method (Principal Component PCA (Performance Analysis)) is a linear algorithm based on transforming features into a new orthogonal base ordered by decreasing variance. PCA is used for data compression, multicollinearity elimination , and 2D/3D visualization.

Method **t-SNE (t- distributed Stochastic Neighbor Embedding)** is a nonlinear algorithm focused on preserving the local structure of data. It is often used to visualize clusters and the

structure of complex multidimensional spaces, especially when working with high-dimensional and difficult-to-interpret data.

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

Equipment and materials

Computer or laptop with internet access

Jupyter Notebook or 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 plot with the PCA result.
3. Plot the explained variance for the first PCA components and comment on the result.

Test questions

1. What is the purpose of dimensionality reduction?
2. How does the PCA algorithm work?
3. What are principal components and how are they interpreted?
4. How to 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 visualization result?
7. How can visualization after PCA help in understanding the data structure?
8. In what problems is it appropriate to use dimensionality reduction?
9. What are the limitations of the PCA method?
10. Why apply dimensionality reduction methods before clustering or classification?

PRACTICAL WORK #16

Topic: Selecting hyperparameters for models

Objective of the work:

To learn how to configure hyperparameters of machine learning models using grid search (`GridSearchCV`) and random search (`RandomizedSearchCV`) methods, and to master the use of cross-validation to increase the reliability of model quality assessment.

Theoretical rationale:

Hyperparameters are model parameters whose values are not determined during training but are manually specified before training begins. 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, and others.

The correct choice of hyperparameters significantly impacts the generalization ability of a model. One approach to their selection is **cross-validation**, in which the training set is divided into several parts, and the model is evaluated on each of them.

For automated selection of parameters the following are used:

- `GridSearchCV` — complete enumeration of all possible combinations of given parameter values;
- `RandomizedSearchCV` - random search of a given number of random combinations.

Both procedures allow one to find the best model configuration based on a given quality metric, usually accuracy . This approach improves model robustness and helps prevent overfitting.

Equipment and materials

Computer or laptop with internet access

Jupyter Notebook or Google Colab

Installed libraries: `sci ki t - l e a r n` , `pandas` , `numpy` , `mat pl ot l i b` , `seabor n`

Datasets: `i r i s` , `w i n e` , `t i t a n i c` , or any CSV file for classification

Typical tasks

1. Tune the parameters of t h e `KNei ghbor sCl assi f i e r` model using `Gr i dSear chCV` , determine the optimal value of `n_nei ghbor s` .
2. Apply `Randomi zedSear chCV` to the decision tree model by adjusting `max_dept h` and `mi n_sampl es_spl i t` and analyze the result.
3. Compare the accuracy of the model with optimal parameters and the model with default parameters.

Test questions

1. What are hyperparameters and how are they different from model parameters?
2. Why perform hyperparameter selection ?
3. What is the difference between `Gr i dSear chCV` and `Randomi zedSear chCV` ?
4. What is cross- validation and how does it work?
5. How to select ranges of values to iterate over?
6. What metrics can be used when selecting hyperparameters ?
7. How to use the resulting model after selecting parameters?
8. What are the risks associated with over-fitting hyperparameters ?
9. What is overfitting in parameter selection and how to avoid it?
10. hyperparameter selection ?

CONCLUSION

The practical work presented in these guidelines covers the key areas of the discipline " *Machine Learning Programming Design Based on Python / Разработка программного обеспечения для машинного обучения на основе Python* " and are aimed at developing students' sustainable 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 according to increasing complexity: from mastering the basic tools of the software environment (Jupyter Notebook (NumPy , pandas , matplotlib) covers the practical application of supervised and unsupervised learning algorithms, including methods for classification, regression, ensemble learning, clustering, and dimensionality reduction. The final sessions focus on improving model performance, including choosing quality metrics and selecting hyperparameters .

Each practical work contains an objective, theoretical justification, a list of required software and data, typical tasks to complete, as well as test questions for self-assessment and ongoing monitoring.

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

The practical portion of the course focuses on the use of open and accessible software tools, ensuring learning flexibility and the applicability of acquired knowledge across a wide range of fields.

EDUCATIONAL, METHODOLOGICAL AND INFORMATIONAL SUPPORT OF THE DISCIPLINE

1. Géron , A. Hands-On Machine Learning with Scikit -Learn, Keras , and TensorFlow : Concepts, Tools, and Techniques to Build Intelligent Systems. — 3rd ed. - Sebastopol, CA: O'Reilly Media, 2022. - 850 p. — ISBN 978-1098125974.
2. McKinney, W. Python for Data Analysis: Data Wrangling with pandas, NumPy , and Jupyter . — 3rd ed. - Sebastopol, CA: O'Reilly Media, 2022. - ISBN 978-1098104030.
3. Huyen , C. Designing Machine Learning Systems: An Iterative Process for Production-Ready Applications. - Sebastopol, CA: O'Reilly Media, 2022. - ISBN 978-1098107963.
4. Raschka , S.; Liu, Y.; Mirjalili , V. Machine Learning with PyTorch and Scikit -Learn: Develop machine learning and deep learning models with Python. - Birmingham: Packt Publishing, 2022. - 770 p. — ISBN 978-1801819312.
5. Murphy, K. P. Probabilistic Machine Learning: An Introduction. - Cambridge, MA: The MIT Press, 2022. - ISBN 978-0262046824.