Data Science with Python

Seminar, BSc Computer Science

Institute of Computer Science, University of Rostock

Course organisers: Olaf Wolkenhauer and Saptarshi Bej, www.sbi.uni-rostock.de

Motivation for this seminar

Digitalisation and the widespread use of information technologies in all areas of our life, are generating data not only in unprecedented quantities but also domains that were unthinkable only a few years ago. With the fairly recent development of algorithms for deep convoluted neural networks, deep learning and artificial intelligence are penetrating all aspects of our life. Autonomous cars are no longer science fiction but a reality. Whether we like it, or not, machine learning techniques will become relevant to most areas in science and industry.

With this seminar, you can learn the terminology, methodologies and tools used for machine learning or data science in general. You should learn how to define a problem, how to prepare data, how to evaluate algorithms, how to improve data analysis workflows and how to present and visualise results. We don't want you to just prepare a text and presentation by searching the Internet for material. Instead, we want you to experiment and code, preparing the report as a documentation of your data analysis.

You find below a selection of 'case studies', from which each student selects one. The goal of the seminar is to prepare a Jupyter notebook using Python to analyse the data and describe the data and their analysis in the style of a scientific report.

We do not expect any prior experience with Python. Instead, the seminar is an opportunity to learn Python and Jupyiter notebooks. This document provides all information on the course content, it's realisation, marking and links to material and further information.

Access to the seminar

The course is only available to students registered with the Institute of Computer Science, University of Rostock. Please note that the lecture and all material will be in English. We will not provide any material in German. By taking this seminar, you accept that the seminar, all material, communication and the reports and/or presentations will be in English.

See StudIP for information on the course. The meetings may take place online. A link to join the video conference will be posted on StudIP.

With your participation you accept the rules and regulations associated with online lectures and exams, as set out by the university and faculty, including the use of Zoom or BigBlueButton Software.

Mit der Teilnahme an dem Kurs erklären Sie dass Sie den „Leitfaden zur Durchführung von Online-Kolloquien“ der Universität Rostock gelesen haben und mit den genannten Bedingungen einverstanden sind. Mit der Nutzung der Plattform Zoom sind Sie mit der Teilnahme für die Prüfung und den sich daraus ergebenden Datenschutzbestimmungen ebenfalls einverstanden.

Course timetable

Always check StudIP for up-to-date information on this seminar. This seminar has the following structure: 1) introduction to the seminar 2) Introduction to machine learning 3) Discussion and preparation of seminar work 4) Deadline for the submission of notebooks. 5) Presentations of results. If there are too many students that have signed up, we may not have presentations. The actual time table can be found on StudIP. You should be aware that this seminar requires you to do some work on your own.

During the first meeting each student will be assigned to one case study (described below). The deadline for the submission of the Jupyter Notebooks will be set during the first session (Send these to saptarshi.bej@uni-rostock.de).

The seminar language is English.

What does it take to learn Machine Learning?

Almost all of what you need to learn Machine Learning, can be found on the Internet, almost all for free. And yet, there is something that is hard to find on the Internet, mostly missing from all written material.

For any one problem, there are usually several algorithms, models, or approaches to choose from. Each comes with a set of parameters, and there are usually different ways to evaluate and validate an approach as well. Which method should I choose; How do the methods compare? How do I go about choosing parameter values, and how do I evaluate my approach? These questions are most important in machine learning and to answer them, one requires experience. For most real world applications, it is impossible to decide in advance what will work and what doesn’t. For most practical purposes, one has to devise a strategy to experiment, test, compare and evaluate.

Machine learning is thus also quite ‘experimental’. A successful machine learning expert is not one that has heard of all methods that exist, but has an idea of how they compare. This requires practical experience, and ideally some theoretical understanding of how the algorithms work, what their basis is. Since this can heavily depend on the context of a question, problem and application, you will probably not find this knowledge in written material. This is where the personal interaction with an expert comes in. In the second session of this seminar, Saptarshi will provide you with an introduction to machine learning, touching on the questions above. You are invited and encouraged to join that session.

Learning outcomes

With this seminar, we are pursuing several learning outcomes. The goal is to introduce you to:

Python

Python is a popular and powerful interpreted language. Unlike R, which is also widely used for data analysis, Python is a complete general-purpose language and platform that can be used for both research and general software development. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented, and functional programming. Python's Wikipedia entry provides a nice overview and history. It is fair to say that Python, across many areas of science and industry has become the most popular language in recent years.

Jupyter Notebooks

Project Jupyter is a nonprofit organization created that supports execution environments for programming languages including Julia, Python and R. A Jupyter Notebook is an interactive computational environment, in which you can combine code execution, rich text, mathematics, plots and rich media. The Jupyter Notebook is a web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data processing, numerical simulation, statistical modeling, data visualization, machine learning. For our purposes we focus on using it for data analysis with Python. Jupyter Notebooks use the Markdown language for formatting the text. Markdown has become a popular choice and is used in an increasing number of contexts. Note: There is also something called JupyterLab, which is a 'next version' Jupyter Notebook. Both are browser-based and pretty much the same for the purpose of this seminar. If you want a stand-alone Python programming environment, that can also edit Jupyter Notebooks, PyCharm by JetBrains is an option. They offer a free edu version.

Data Science

Data Science is an interdisciplinary field that combines programming and computer science methodologies with data analysis and statistical data. A data scientist explores datza for real world applications, drawing from a wide range of tools and methodologies. The most important skill of a data scientist is to have an appreciation for a wide range of techniques, from computer science, statistics, and machine learning. The processing of data, analysis and visualisation has become a core competency in information or knowledge-based societies and business. A data scientist has knowledge of the mathematical and statistical foundations, and is yet not afraid to get his/her hands dirty with real, messy data.

Machine Learning

Machine learning (ML) is the study of computer algorithms that can learn from data. Machine learning algorithms are also at the core of Artificial Intelligence. Given a set of "training data", machine learning algorithms build a model that can be used for decision making and predictions. Machine learning approaches can be roughly divided into four broad categories: Supervised learning, Unsupervised learning, Reinforcement learning and Deep learning. Dimensionality reduction, clustering, classification and regression analysis are key concepts required for practical applications. Machine learning and artificial intelligence have become dominant fields, driving a variety of businesses, with spectacular developments over the last ten years or so.

Scientific writing and presentation

To some extent you are only as clever as other people believe you are. We have met numerous people with exceptional technical skills, who struggled with their career, for only one reason -

communicating their work effectively. Whether you become a scientist in the academic world, or you work in industry, presenting ideas and results in a concise format is an essential skill. For most forms of communications - presenting a project idea, project results, a publication, a poster or introducing yourself to someone else, you will have only a few minutes available to make the decisive impression. We want this seminar to be an opportunity to practice your scientific writing and presentation skills. Following the first meeting, where we introduce the case studies on which you will work, we share in a second meeting our experience in effective communication.

Note: The list of objectives for this seminar is long. The links with background material provided below, can be overwhelming. Learning Python can easily fill a whole semester, and this seminar gives you about one month to use Python for Machine Learning … We should thus be clear that this seminar will be a challenge, even for second semester computer science students. Remember therefore that you are embarking on a learning process and that errors, and error messages in particular, are perfectly normal. They are part of the learning process. You are not implementing or coding machine learning algorithms, but using existing functions to analyse data. Nevertheless, you should know that error messages are fine. Everyone gets them ... all the time. Often it is a syntax issue like missing brackets or a missing space. You can trust the "error message", it will give you a lead to its solution. If you are stuck, speak to fellow students, or add stack overflow as a resource. You may copy paste the error message into Google or add a new thread on stackoverflow. Most of us never had to create a new thread in Stackoverflow ... any error they may run into - someone else had before and you can find solutions online.

Useful Links & Materials

There are plenty of guides available on how to start with Python programming, including this guide by Kerry Parker.

The data scientist workflow we have in mind for this seminar has been described nicely in a Python tutorial by Jason Brownlee. If you want to dig deeper, learning Python and/or data analysis, machine learning and AI techniques, we recommend looking at Jason Brownlee's webpage for free tutorials but also excellent eBooks, with many practical examples.

It is not necessary to study all the links below. To start with, you need to install Python and Jupyter Notebooks. This can be done in various ways (Installing Jupyter Notebook, Anaconda Python/R Distribution - Free Download) and here is a quick guide:

How to Setup Your Python Environment for Machine Learning with Anaconda

There is also something called JupyterLab, which is a 'next version' Jupyter Notebook. Both are browser-based and pretty much the same for the purpose of this seminar. If you want a stand-alone Python programming environment, that can also edit Jupyter Notebooks, PyCharm by JetBrains is an option. They offer a free edu version.

More recently, we have successfully used Google Colab to edit and run Jupyter Notebooks - with not need to install an IDE or Python! If you run into problem with the installations, or you do not wish to install anything, you can complete this seminar using just a web browser.

Python

Welcome to Python.org

Python For Beginners

Learn Python - Free Interactive Python Tutorial

The Python Tutorial — Python 3.8.2 documentation

Python Programming Tutorials

Jupyter notebooks

The Jupyter Notebook

The Jupyter Notebook — Jupyter Notebook 6.0.3 documentation

Notebook Examples — Jupyter Notebook 6.0.3 documentation

Jupyter Notebook for Beginners Tutorial

(Tutorial) Jupyter Notebook: The Definitive Guide

Jupyter Notebook Best Practices for Data Science

Machine learning with Python

scikit-learn: machine learning in Python — scikit-learn 0.22.2 documentation

Beginner's Guide to Machine Learning with Python

Machine Learning Tutorial for Beginners

pandas - Python Data Analysis Library

NumPy — NumPy

Data Visualisation with Python

Bokeh 2.0.1 Documentation

Matplotlib: Python plotting — Matplotlib 3.2.1 documentation

Crisp python plots based on visualization theory

Tutorial Example: Iris flower data set

The data scientist workflow we have in mind for this seminar has been described nicely in a Python tutorial by Jason Brownlee The main difference to the seminar work is, that we like you to write the notebook as if you are writing a textbook example. We therefore like you to expand on the choices made in programming and in choosing the methods for data analysis. The most important idea in writing is thus to compare and contrast.

The dataset in Jason Brownlee's tutorial contains 150 observations of iris flowers. There are four columns of measurements of the flowers in centimeters. The fifth column is the species of the flower observed. All observed flowers belong to one of three species. You can learn more about the "Iris flower data set" on Wikipedia. This dataset has become a famous example, widely used in data science training. For the case studies below, we shall also use well known benchmark datasets. You can load many well known benchmark datasets directly from the UCI Machine Learning Repository.

Jason Brownlee has made available another tutorial, for a step-by-step coding of the k nearest neighbor algorithm, also using the Iris data set. This tutorial demonstrates nicely the workflow we would like you to develop, in which the notebook becomes a documentation of your learning and experimentation.

Tips for all modules

Python libraries: The installation of Python and Jupyter Notebooks was already discussed above. The Anaconda distribution automatically installs the Jupyter Notebook and most of the Python libraries that you will need. For machine learning, the most important Python libraries are scikit-learn, NumPy and Pandas. Pandas is designed to import the datasets in your Jupyter notebook. NumPy is designed to convert lists/arrays to Python objects called numpy arrays. The Numpy library provides easy ways to manipulate such arrays. The library scikit-learn enables you to execute machine learning algorithms. Moreover, it helps you to tune your models by adjusting hyper-parameters and test the performance of your model. To get a basic understanding of how this works, begin by exploring the Jupyter notebook: Train your own model.ipynb. For the visualisation of data, Matplotlib is the most important library.

NOTE: For any ML related tool, a google search of 'how to use XX using sklearn' should guide you to examples of using the tool XX.

What we recommend

- Watching youtube tutorials. For example
  - Scikit-Learn Tutorial | Machine Learning With Scikit-Learn | Sklearn | Python Tutorial | Simplilearn
  - Complete Python Pandas Data Science Tutorial! (Reading CSV/Excel files, Sorting, Filtering, Groupby)
  - Complete Python NumPy Tutorial (Creating Arrays, Indexing, Math, Statistics, Reshaping)
  - Matplotlib Tutorial (Part 1): Creating and Customizing Our First Plots

  Also watch tutorials relevant to your topics.
- Interaction among module mates. Keep in touch with your colleagues dealing with the same model. Some models are basic and some are more advanced. We tried to distribute them in a way such that the workload is uniform. Nevertheless, interacting with your colleagues dealing with the same module will make your job easier.

We we expect

- Within each notebook, at least a one page discussion on each model, based on your literature research and understanding. You might not be able to grasp all the mathematics and theory for the models but you should be able to grasp the philosophy of the model - what it is intended for, how it relates to and compares with other algorithms.

- Mention two or three pros and two or three cons of the models you are studying.
- Provide reference (online articles or research papers) to the statements you write. This is a good scientific practice.
- *What we do not expect but would appreciate - no, value A LOT - is if you decide to find or create your own datasets or examples. Yes, there is a lot out there but if you come up with something that helps understanding the topic and which you created or discovered yourself, beyond what we write here, this is the extra bit that we are hoping for* ;)

Preparing your Jupyter Notebook

One learning outcome for the seminar is to practice scientific writing. Usually, this means writing a project report or manuscript for publication. The tips we give below and the guidelines for marking have this type of scientific writing in mind.

However, the beauty of Jupyter Notebooks is that it combines coding with documentation and writing. That means, using Jupyter Notebooks allows you to document your coding and analysis workflow. This makes everything you programme more accessible and reproducible. Code used to be something tugged to the end of a report. With Jupyter Notebooks you can tell the story of your coding and analysis, while coding and running algorithms.

The structure of project reports and scientific publication, however, have one characteristic that contradicts our idea of using Jupyter Notebooks: In reports and publications you summarise the results of your work, without spending much effort on documenting the actual workflows, including mistakes made, falsely chosen directions, and dead alleys. Often these experiments are what provide the greatest learning opportunities.

Ensure there is a red-line going through your notebook, connecting each section. Ideally, your notebook is read like a story, from beginning to end, with no need to jump forth or back. Remember to connect sections, focussing on the last and first paragraphs. Each paragraph should contain one and only one message. The first sentence of a paragraph is the “topic position”, giving context, motivation and background, while the last sentence of a paragraph is the “stress position”, where the reader expects the central message that is arising from the paragraph. Try to raise interest, wherever possible, consider an engaging conversational style. It is often good to imagine you explain something to your partner, friend or colleague. We want you to create textbook examples that would show a learner how data are analysed with Python. We hope that you enjoy this experience - not just copying any report together but sharing your learning experience and also code and analyse data!

Module I: Supervised Learning

Goal: Module I is about attaining working knowledge on Supervised Learning algorithms. The dataset you will use to test the performance of your model is the MNIST dataset which is one of the basic datasets to be explored by beginners in the field of Machine learning. This dataset is about classifying handwritten digits (learn more from MNIST database). You can download the dataset from MNIST in CSV. Each of you have been assigned to work on two ML models.

Experiment Design: Machine learning models are prone to overfitting. The experiments have to be carefully designed to ensure that your model’s predictive ability is robust to randomness. You will first train your models with the 5x5-fold stratified cross validation protocol (using only the training data and using parameter optimization using RandomizedSearchCV sklearn.model_selection.RandomizedSearchCV — scikit-learn 0.22.2 documentation). Once you ensure that your model is not overfitting, you will train your model with the entire training data (with best parameter settings obtained using RandomizedSearchCV) and then validate your model with the test data available. The performance measures you will use are: Accuracy score and Confusion matrix.

Expected output from you (module specific):

- Proper documetened parameter optimization in your Jupyter notebook for each model.
- Research about the key parameters of your model
- Self explanatory labelled bar plots comparing the performance accuracy for both of your models with details of parameter settings.
- A one page note on each model based on your literature research and understanding. You might not be able to grasp all the mathematics and theory for the models. In that case trying to capture the philosophy of the model through your writing would be crucial.
- Mention two-three pros and two-three cons of the models you are studying.
- Provide reference (online articles or research papers) to the statements you write. This is a good scientific practice.

Case Study 1: kNN and LGBM models

For Case Study 1 you will learn two ML models. k-Nearest Neighbours and Light Gradient Boosting Machine (LGBM). k-Nearest Neighbours is a basic algorithm but can be important in certain scenarios. You can begin your journey on this successful algorithm here: KNN Algorithm - How KNN Algorithm Works With Example | Data Science For Beginners | Simplilearn. Even if this algorithm is basic, make no mistake, there has been a lot of research on this.

LGBM is a more recent and advanced model from Microsoft. You might not get too many materials for understanding its theory. However, it is related to models like ADABoost, Gradient Boosting and XG-Boost. Interact with your colleagues dealing with these to understand it better. Focus on the applications this model has found. Begin here: Boosting Machine Learning Tutorial | Adaptive Boosting, Gradient Boosting, XGBoost | Edureka.

You may also refer to:

- Decision Tree Algorithm With Example | Decision Tree In Machine Learning | Data Science |Simplilearn
- https://www.youtube.com/watch?v=3CC4N4z3GJc&t=81s
- Boosting Machine Learning Tutorial | Adaptive Boosting, Gradient Boosting, XGBoost | Edureka
- Welcome to LightGBM's documentation! — LightGBM 2.3.2 documentation

Case Study 2: Naive Bayes and SVM models

For Case Study 2 you will learn two ML models - Naive Bayes and SVM. The Naive Bayes model is a ML model that is also one of the easier models using basic probability theory. Begin your journey here:

- Naive Bayes Classifier in Python | Naive Bayes Algorithm | Machine Learning Algorithm | Edureka
- Naive Bayes Classifier | Naive Bayes Algorithm | Naive Bayes Classifier With Example | Simplilearn.

There can be variations of this model like Gaussian Naive Bayes. You might want to explore these as well.

Support Vector Machine (SVM) is a more advanced and usually better performing model. There has been a lot of research on this model. Learn about the Kernel-trick and several kernels for this model and implement them in your code. Learn how to make SVM faster. Begin your journey with these:

- Support Vector Machine - How Support Vector Machine Works | SVM In Machine Learning | Simplilearn
- Support Vector Machines, Clearly Explained!!!
- 16. Learning: Support Vector Machines

Case Study 3: Random Forest and ADABoost models

For Case Study 3 you will learn two ML models: Random forest and ADABoost. For understanding both these models you would like to understand how decision trees work. Once you know the concept of Decision trees and the concept of Boosting, this will not be a difficult task.

Start with these:

- Decision Tree Algorithm With Example | Decision Tree In Machine Learning | Data Science |Simplilearn
- Random Forest Algorithm - Random Forest Explained | Random Forest in Machine Learning | Simplilearn
- AdaBoost, Clearly Explained
- Boosting Machine Learning Tutorial | Adaptive Boosting, Gradient Boosting, XGBoost | Edureka

Case Study 4: Decision Trees and Gradient Boosting models

For Case Study 4 you will learn two ML models: Decision Tree and Gradient Boosting. We recommend you to begin with Decision Trees. Once you know the concept of Decision trees, Random forest and the concept of Boosting, this will not be a difficult task.

Start with these:

- Decision Tree Algorithm With Example | Decision Tree In Machine Learning | Data Science |Simplilearn
- Random Forest Algorithm - Random Forest Explained | Random Forest in Machine Learning | Simplilearn
- AdaBoost, Clearly Explained
- https://www.youtube.com/watch?v=3CC4N4z3GJc&t=81s
- Boosting Machine Learning Tutorial | Adaptive Boosting, Gradient Boosting, XGBoost | Edureka

Case Study 5: Logistic regression and XGBoost

Logistic Regression is a basic supervised learning algorithm. It is a clever modification of the idea of linear regression to be used properly for classification tasks. Learn more on this from the tutorial link: https://www.youtube.com/watch?v=OCwZyYH14uw. XGBoost is an advanced concept for which you might need to understand the concept of boosting properly.

You may want to refer to:

- Decision Tree Algorithm With Example | Decision Tree In Machine Learning | Data Science |Simplilearn
- AdaBoost, Clearly Explained
- https://www.youtube.com/watch?v=3CC4N4z3GJc&t=81s
- Boosting Machine Learning Tutorial | Adaptive Boosting, Gradient Boosting, XGBoost | Edureka

Module II: Unsupervised Learning

Goal: Module I is about attaining working knowledge on Unsupervised Learning algorithms. For Unsupervised learning you do not need to care about fitting your data to the labels. However, if the labels are known, you can colour-code your plots with the labels (Visualising high-dimensional datasets using PCA and t-SNE in Python). Each of you will apply one/two dimension reduction algorithms on three datasets. Students having one method as a case study are likely to encounter more complicated and advanced methods. Moreover, they are expected to go in real detail of parameter estimation while implementing the algorithm.

- **MNIST dataset:** This dataset is about classifying handwritten digits (learn more from MNIST database). You can download the dataset from MNIST in CSV. Each of you have been assigned to work on two ML models.
- **Credit Fraud dataset:** This dataset is from a kaggle competition. This is an imbalanced dataset with very few examples of credit fraud. You can find the details dataset in Credit Card Fraud Detection
- **Swiss Roll dataset:** Swiss roll dataset can be produced using scikitlearn using Swiss Roll reduction with LLE — scikit-learn 0.22.2 documentation. You can produce such a dataset specifying the number of data points you would like to produce. You can try this with 2000 data points.

Experiment Design: For this module, you are expected to generate 2-D and 3-D reduction plots for the given data sets. In case the datasets are labelled, you have to use the unlabelled versions to create the plots. But you can use the labels to colour code the plots to view the clusters in your data.

Please refer to the links:
- https://scikit-learn.org/stable/modules/manifold.html#spectral-embedding
- https://scikit-learn.org/stable/auto_examples/manifold/plot_compare_methods.html#sphx-glr-auto-examples-manifold-plot-compare-methods-py

Expected as output from you (module specific):
- Proper documented parameter optimization in your Jupyter notebook for each model.
- Research about the key parameters of your model.
- Self-explanatory labelled scatter plots (2-D and 3-D) showing the colour coded data in reduced dimensions.

Case Study 6: PCA and t-SNE

Principal component analysis is the most basic dimension reduction algorithm. Learn more about it from:
- StatQuest: Principal Component Analysis (PCA), Step-by-Step

t-SNE is a more advanced algorithm. You can learn more about the algorithm from the following link:

- t-SNE: Clearly Explained
- Visualizing Data Using t-SNE

Try and implement the algorithms for the given three datasets and try to visualize the difference you see. Comment on this in your Jupyter notebook.

Case Study 7: ISOMAP and MDS

ISOMAP is one of the earlier modifications on PCA. You can find more on ISOMAP and MDS algorithm in the links:

- StatQuest: Principal Component Analysis (PCA), Step-by-Step
- Ali Ghodsi, Lec 4: MDS, Isomap, LLE
- StatQuest: MDS and PCoA

Implement the algorithms for your datasets and compare the results.

Case Study 8: UMAP

UMAP is an advanced algorithm for dimension reduction. For this know a bit about PCA and ISOMap. Then study a bit about t-SNE.

- StatQuest: Principal Component Analysis (PCA), Step-by-Step
- Ali Ghodsi, Lec 4: MDS, Isomap, LLE
- Visualizing Data Using t-SNE
- UMAP Uniform Manifold Approximation and Projection for Dimension Reduction | SciPy 2018 |

Understand what the algorithm adopts from its predecessors like PCA, ISOMAP, t-SNE. Use it for the three datasets with several parameter settings for 'metric', 'n-neighbours' etc and comment on your experiences with these. This link is also going to be very useful:

How UMAP Works — umap 0.4 documentation

Since you have one model to deal with, you are expected to submit a two-page report on this model. It is quite an advanced model, so you will have enough materials

Case Study 9: LLE

Locally linear embedding(LLE) is a vastly popular dimension reduction method. It has several variants such as Hessian EigenMapping, Local tangent space alignment. You might try coding for these variants using the 'method' parameter in scikit-learn documentation and compare their performances. Read more about the algorithms from

- Locally Linear Embedding
- Ali Ghodsi, Lec 4: MDS, Isomap, LLE
- 23 Reducing dimensions Local Linear Embedding
- Locally Linear Embedding Python tutorial

Since you have one model to deal with, you are expected to submit a two-page report on this model. It is quite an advanced model, so you will have enough materials

Module III: Foundations of imbalanced data analysis

Goal: Module III is about attaining knowing and understanding several aspects of imbalance datasets. To get a preliminary overview on imbalance datasets please see the articles:

- https://machinelearningmastery.com/what-is-imbalanced-classification
- SMOTE - Synthetic Minority Oversampling Technique

In this module we provide you several case studies that help you understand several aspects of imbalance datasets. You will be exploring two datasets using two supervised ML models.

- **Credit Fraud dataset:** This dataset is from a Kaggle competition. This is an imbalanced dataset with very few examples of credit fraud. You can find the details dataset in Credit Card Fraud Detection
- **Mammography dataset:** You can access this dataset using imblearn.datasets. fetch_datasets function. Hint: for this module you need to learn more about the imblearn library: imblearn.datasets.fetch_datasets

For the credit fraud dataset you need to use the Random Forest classification algorithm with default parameters. Although this is not needed for this module, in case you want to learn more about the algorithms interact with your Module I colleagues or see the videos:

- Decision Tree Algorithm With Example | Decision Tree In Machine Learning | Data Science |Simplilearn
- Random Forest Algorithm - Random Forest Explained | Random Forest in Machine Learning | Simplilearn

For the mammography dataset use the knn classification algorithm with k=30. In case you are curious about the model, watch this : KNN Algorithm - How KNN Algorithm Works With Example | Data Science For Beginners | Simplilearn.

There are several approaches to handle class imbalance. You will be exploring oversampling approaches. Each of you will explore two oversampling algorithms. You will read about the algorithms from respective research papers and then implement the oversampling algorithms on the above mentioned datasets using respective classification algorithms as instructed above. You DO NOT need to tune parameters for the classification algorithms. For your respective oversampling algorithms, if there is a parameter: n_neighbour or k_neighbour, please use 5 as its value. For this module you need to explore three special libraries:

- smote-variants
- Welcome to imbalanced-learn documentation! — imbalanced-learn 0.5.0 documentation
- imblearn.datasets.fetch_datasets

These libraries will be sufficient for the implementation of all algorithms assigned to you (especially the first link(smote-variants)). To make coding easier follow: SMOTE with Imbalance Data

Case Study 10: Performance measures on imbalance datasets

For this case study you will be exploring several performance measures on imbalance datasets. You will use only the Card Fraud Detection dataset for this study. You will use the random forest classifier to detect credit frauds from the dataset. Use a train test split of 80:20. For use random forest classifier with the following specifications:

- Baseline training with the original data

- SMOTE oversampling on the data
- DBSMOTE oversampling on the data
- Random undersampling on the data
- Without data preprocessing (oversampling or undersampling), but using cost sensitive random forest (fix cost of minority class: majority class 100:1) (see https://www.kaggle.com/christiantheilhave/class-imbalance-with-weighted-random-forest)

Report the following performance measures for each of these cases:
- Accuracy
- Balanced accuracy
- Precision
- Recall
- F1-Score
- G-Mean
- Kappa Score
- AUC score
- Average Precision Score (APS)

Plot the F1-Score vs G-mean for the 5 cases mentioned above. Plot the F1-Score vs Kappa Score for the 5 cases mentioned above. Plot the AUC vs APS for the 5 cases mentioned above.

Case Study 11: Cost sensitive learning

For this case study you will be exploring several performance measures on imbalance datasets. You will use both the Card Fraud Detection and Mammography dataset (imblearn.datasets.fetch_datasets) for this study. You will use the random forest classifier to detect credit frauds from the dataset. For the mammography dataset use the knn classification algorithm with k=30, as instructed before. Use a train test split of 80:20. For use random forest classifier with the following specifications:
- Baseline training with the original data
- No oversampling and Using cost sensitive random forest (fix cost of minority class: majority class 10:1)
- No oversampling and Using cost sensitive random forest (fix cost of minority class: majority class 100:1)
- SMOTE oversampling and Using cost sensitive random forest (fix cost of minority class: majority class 10:1)
- SMOTE oversampling and Using cost sensitive random forest (fix cost of minority class: majority class 100:1)

Read the following article in kaggle to get a preliminary idea on cost sensitive learning to begin: https://www.kaggle.com/christiantheilhave/class-imbalance-with-weighted-random-forest)

Report the following performance measures for each of these cases:
- Balanced accuracy
- F1-Score
- G-Mean
- Average Precision Score (APS)

Compare and comment on the scores.

Expected output from you (for the following case studies 12-14):

- Use default parameters for the Oversampling model wherever applicable. You DO NOT need to do parameter optimizations for the classifiers knn and random forest. Use the values as instructed (k=30 for knn and default for random forest).
- Use the library smote-variants to access the oversampling algorithms
- Self-explanatory labelled bar plots comparing the performance measures for both of your oversampling models.
- Generate without PCA plots for the training dataset for the cases 1) without oversampling 2) with oversampling
- Brownie points for mentioning one-two pros and one-two cons of the models you are studying.
- Provide reference (online articles or research papers) to the statements you write. This is a good scientific practice.

Experiment Design (for the following case studies 12-14): Machine learning models are prone to overfitting. The experiments have to be carefully designed to ensure that your model's predictive ability is robust to randomness. You will first train and test your models with the 5x5-fold stratified cross validation protocol. Present the average performance for all the folds. Use performance measures: F1-Score, G-Mean, Balanced Accuracy, Average precision score see: sklearn.metrics. Generate necessary plots to compare the three scores between two models you used.

Case Study 12: Borderline-SMOTE and CURE-SMOTE

To get a headstart see the articles:

- Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning
- CURE-SMOTE algorithm and hybrid algorithm for feature selection and parameter optimization based on random forests

Case Study 13: SVM-SMOTE and ADASYN

To get a headstart see the articles:

- Tackling class imbalance with SVM-SMOTE - vclab
- Kernel-based SMOTE for SVM classification of imbalanced datasets - IEEE Conference Publication
- ADASYN: Adaptive Synthetic Sampling Method for Imbalanced Data
- ADASYN: Adaptive synthetic sampling approach for imbalanced learning - IEEE Conference Publication

Case Study 14: k_MeansSMOTE and AND-SMOTE

To get a headstart see the articles:

- Automatic Determination of Neighborhood Size in SMOTE
- k-Means SMOTE

Note

Classification with imbalance datasets is an important real-world problem, with applications in numerous application domains. This topic is close to our heart and is part of our research programme. If you find this topic interesting as well, you may find reading the following useful:

B Krawczyk: Learning from imbalanced data: open challenges and future directions. *Prog Artif Intell* (2016) 5:221-232

S Bej, N Davtyan, M Wolfien, M Nassar, O Wolkenhauer: LoRAS: An oversampling approach for imbalanced datasets. Accepted for publication in *Machine Learning* (2020)

Communicating your work effectively

Effective communication in science implies a concise presentation (of a project idea, project progress, results, publication). The most common format of scientific communication is that of an 'abstract', with a length of around 200-250 words, which corresponds to an elevator pitch. Abstracts are found on posters, in scientific publications, project reports, grant proposals and on websites, describing your own work, person or project. 150-250 words fills about half an A4 page or a single page of a website. In poster presentations, or reporting progress on a project in a corporate environment, the time you have available to present your results is equivalent to writing an abstract.

It turns out that all these forms of communication (elevator pitch, poster presentations, paper abstract etc) can be structured by answering the following questions:

1. What is the context?
    a. One sentence providing a basic introduction to the area, comprehensible to a scientist in any discipline.
    b. One or two sentences of more detailed background, comprehensible to scientists in a related discipline.

2. What is the focus?
    a. One sentence clearly stating the general problem.
    b. Identify a research gap, and raise interest through selected wording.

3. What will be done? (For papers, posters: What was done?)
    a. One sentence stating the methodology, technology or approach being used.

4. What will come out? (For papers, posters: What has come out?)
    a. One sentence summarizing the expected outcomes. (For papers, posters: What are the results?)
    b. One sentence explaining what the main results reveals in direct comparison to what was thought to be the case previously, or how the main result adds to previous knowledge.

5. What is the value?
    a. One sentence to put the results into a more general context, indicating the (potential) impact of the work.
    b. One sentence to provide a broader perspective, readily comprehensible to a scientist in any discipline.

If you answer these questions, you will never again worry about summarizing your work. At the end of this seminar, you should be ready to give an oral summary of your project.

During the last meeting each student, or group, will present their Case Study with one slide only, and max 250 words presentation. We urge you to follow the structure above.

Scientific Writing

We here provide guidance for scientific writing in general. See below our comments on how for this seminar, things differ slightly.

Title of the case study

Author

- Name, affiliation, Email address.

Abstract

- Try to structure the abstract by answering the following question (appr one sentence per question):
    - What is the context or background of the case study?
    - What is the challenge, research question, addressed?
    - What methods, tools and methodologies are used?
    - What are the results from the analysis?
    - What is the use, value, or application?
- The abstract should be about 150 words, max 200 words.

Keywords

- max 5.

Introductory Part

- State the question you wish to address.
- Describe how this question arises from its context.
- State how you are going to focus on the subject.
- State the objectives and the approach taken.
- Outline the report.

State-of-the-Art (Literature Review)

- This should reflect the introductory part with more factual details.
- A critical approach of the literature is encouraged..
- Highlight achievements, progress, open problems and gaps in the literature.
- Consider a range of sources, journals, Internet pages, books and make use of online search tools.

Methodology

- Argue a roadmap for the project, decide upon the approach taken and the methodologies employed.
- Justify the approach by relating to the literature. Compare and contrast methods, algorithms and tools.

Results and Discussion

- State what you have found.
- Determine the reliability of the conclusions made.

- How do your results compare to the literature, mention assumptions and provide an honest answer to possible weaknesses.
- Have you uncovered any surprises, did problems emerge, do the results confirm a hypothesis or known knowledge?
- State clearly the disadvantages and weaknesses of your approach/results in comparison to other works.

Conclusions
- What can be concluded from the case study and the results of the analysis?
- Given the opportunity, what could/should be done differently?
- State open problems, make recommendations for future directions of the work.

Structure of the Seminar Jupyter Notebook

The structure above is common with scientific reports in a number of fields. For computer science, and this seminar in particular, not all aspects of the structure above applies. Since one objective of this seminar is to learn scientific writing, we have provided you with the structure above.

The beauty of Jupyter Notebooks is that it combines coding with documentation and writing. Some of the best known Python textbooks are entirely written as Jupyter Notebooks. The structure for a scientific report or publication, given above, has one characteristic that does not fit our purpose well. When one is presenting a work, say in a publication, the process through which methods were implemented, the construction of analysis pipelines (workflows) is not described. What is described are the methods and results, but not how one actually got to this outcome.

In our case, where we look at textbook examples for the analysis of datasets, the main emphasis will be on documenting the “workflow” and “train of thoughts” that a data scientist follows in analysing data. We want you to create textbook examples that would show a learner how data are analysed with Python.

The data scientist workflow we have in mind for this seminar has been described nicely in a Python tutorial by Jason Brownlee See also another tutorial by him, showing nicely how one can use Jupyter Notebooks for an engaging documentation of a data science workflow.

Marking of the seminar work

By participating in this seminar, you agree to submit a report. This report, in the form of a Jupyter Notebook, will be marked. During the last meeting each student, or group, will present their Case Study with one slide only, and max 250 words presentation. (If there are too many students in the seminar, we may decide to skip the presentations).

In order to give you an idea of what is expected, we are going to adapt the following general guidelines for marking written works, including dissertations, final year projects etc.

1. Preliminary Studies: background understanding, literature search, exploratory work.
2. Technical Achievement: difficulty of the project, embracing new ideas,originality, integrity of work, meeting the aims, practical considerations.
3. Analysis of Results, Conclusions: appropriate and thorough methods, clarity and accuracy of analysis, adequate acknowledgement of sources.

4. Organisation of the Report: abstract, clearly expressed aim, objectives and motivation, appropriate partitioning and structure of the document, references & citations.
5. Clarity of Text and Language: ease of understanding.
6. Diagrams and Tables: legibility, clarity of caption texts, visual clarity of information in figures, labeling, numbering, references in text.

All aspects are weighted equally, except technical achievements which are weighted twice as much as any other category.

- **100%** would correspond to work, in that aspect, that reaches the highest standards that could be expected of a professional scientist with experience. Contains all of the relevant information with no errors or only insignificant errors. Displays excellent understanding of the subject within a wider context. Gives extensive evidence of critical awareness and independent thinking.
- **80%** corresponds to work that is, in that aspect, mainly of professional standard, but has few shortfalls. Displays good understanding of the subject within a wider context. Has reached beyond the essential material.
- **60%** corresponds to work which has, in that aspect, both strong and weak features with the stronger features being in the majority. Less evidence for critical awareness and independent thinking.
- **40%** corresponds to work that is marginally satisfactory in that aspect. There are several failings, but there are also some achievements and positive features. Little evidence of critical awareness and independent thinking. Lack of evidence for a deeper understanding of the subject within a wider context.

Translation into course marks

Mark	**%**	**Criteria**
1	90-100	**Outstanding,** comprehensive, factually faultless answer, evidence of originality and extensive knowledge.
1.3	80-89	**Very good**, factually faultless, good evidence of supplementary reading.
1.7	70-79	**Good**, logical presentation, evidence of supplementary reading, good coverage.
2.0-2.3-2.7	60-69	**Comprehensive**, clear, thorough, evidence of comprehensive coverage of material.
3.0-3.3-3.7	50-59	**Adequate**, perhaps some errors or key facts missing, expression moderate.
4.0	40-49	**Incomplete**, sparse information, some inaccuracies, poor coverage of material, no sign of comprehensive preparation, difficulties in expressing knowledge.
failed	30-39	**Deficient**, many omissions or errors, expression and argument poor**.**

failed	15-29	**Poor**, largely irrelevant to the question, little or no substance/factual material**,** apparent lack of preparation.
failed	0-14	**Inadequate**, obvious lack of preparation, little or no relevance to question, wrong approach and presentation.

For Machine Learning, ‘very good’ or ‘outstanding’ are attributes that are mostly associated with work that justifies and explains things. For example, the choice of an algorithm/model is justified, e.g. through a quantitative comparison. Parameter values have to be chosen, and justifying the choice, the reasoning is often non-trivial. Transparency about assumptions being made are also an important factor. Finally, demonstrating an understanding of the mathematical basis of an algorithm, or providing a mathematical explanation for a result, is often challenging and hence a criteria to set a work apart.

Note: What we are looking for are notebooks that are fun and easy to read, provide clear instructions and encourage the reader to use Python. We want you to learn Python yourself and then share your learning experience. You may draw from examples and ideas related to these datasets on the Internet but try to be creative in how you describe and present the case studies and their analysis. Compare and contrast the method, tools and algorithms used.

Also note that many of the algorithms that you are to encounter in this course are mathematically quite advanced. We thus do not expect you to understand these in full mathematical rigor. The implementation however, is quite straightforward. If you refer to the provided materials properly and do a bit of research, then this should not be very difficult. To get a headstart refer to the materials

- https://www.youtube.com/watch?v=HW29067qVWk
- https://drive.google.com/open?id=11kGfiloxiCSAjD-iphWzY4xbg_U0zC8j

We assure you that 70% of your marks will come from the implementation and coding part which will be judged from your Jupyter notebooks. 30% of your marks will be judged for the effort that you put in understanding the algorithm that is the two page report that you are required to submit on your understanding of the algorithms.