Τελευταία ενημέρωση 9/11/2025

Ανάλυση δεδομένων με τη γλώσσα προγραμματισμού Python ΕΚΔΔΑ

7.1 Επισκόπηση βασικών αλγορίθμων μηχανικής μάθησης

Εβδομάδα 9/9

Προετοιμασία διαφανειών: Γκόγκος Χρήστος

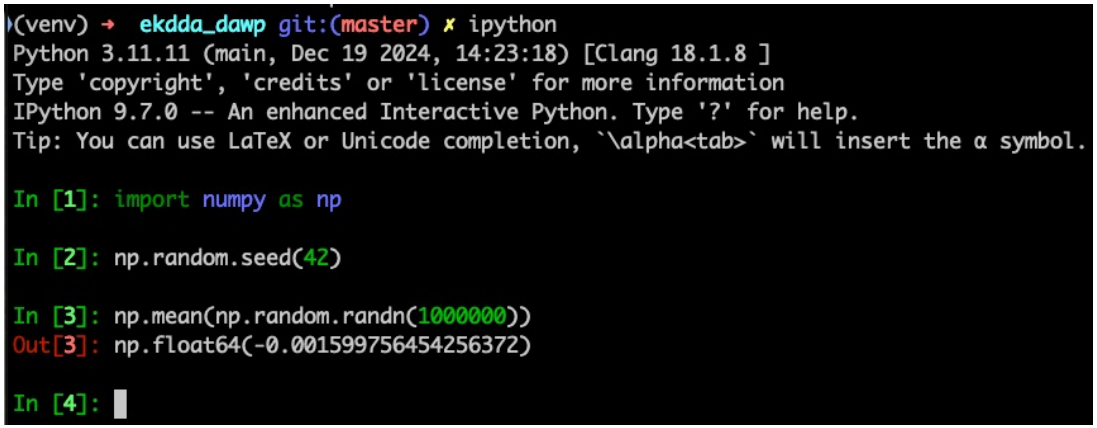
Βασικές βιβλιοθήκες της Python για ανάλυση δεδομένων

- **Jupyter**: διαδραστικό περιβάλλον σε μορφή διαδικτυακού σημειωματαρίου
- **IPython**: διαδραστικό περιβάλλον γραμμής εντολών
- **NumPy**: αποδοτική επεξεργασία αριθμητικών δεδομένων
- **Pandas**: μετασχηματισμός και ανάλυση δεδομένων που βασίζεται στο NumPy
- **Polars**: ταχύτερη επεξεργασία δεδομένων σε σχέση με το Pandas, κατάλληλη για μεγάλα δεδομένα
- **Matplotlib** και **Pyplot**: οπτικοποίηση δεδομένων (το pyplot είναι subpackage του matplotlib και μπορεί να προσπελαστεί απευθείας από το pandas)
- **Seaborn**: βιβλιοθήκη οπτικοποίησης δεδομένων πάνω από το matplotlib, βελτιωμένη αισθητική γραφημάτων
- **Plotly:** διαδραστική βιβλιοθήκη για δημιουργία dashboards και EDA (Exploratory Data Analysis)
- **Scikit-Learn**: βασίζεται στο NumPy και περιέχει SoTA υλοποιήσεις αλγόριθμων μηχανικής μάθησης, συμπεριλαμβανομένων βασικών νευρωνικών δικτύων
- **Statsmodels**: στατιστικά μοντέλα, έλεγχοι υποθέσεων, ανάλυση παλινδρόμησης
- **XGBoost / LightGBM / CatBoost**: βιβλιοθήκες για αλγορίθμους boosting υψηλής απόδοσης
- **PyTorch**: βιβλιοθήκη βαθιάς μάθησης (neural networks) από τη Meta
- **TensorFlow**: βιβλιοθήκη βαθιάς μάθησης (deep learning) από τη Google

```
$ pip install jupyter numpy pandas matplotlib seaborn scikit-learn
```

IPython

- Διαδραστική κονσόλα της Python με προχωρημένα χαρακτηριστικά όπως αυτόματη συμπλήρωση κώδικα και χρωματισμό σύνταξης (syntax highlighting)
- Προσφέρει αριθμημένη είσοδο/έξοδο, π.χ. In [1]:, Out [1]:



Σημειωματάρια (notebooks)

- Τα Jupyter Notebooks μπορούν να εκτελούνται είτε τοπικά (localhost) είτε σε κάποιο απομακρυσμένο υπολογιστή
- Το JupyterLab είναι ο διάδοχος του Jupyter Notebook
- Το VS Code δίνει τη δυνατότητα της απευθείας εκτέλεσης Jupyter notebooks (όπως και περιβάλλοντα ανάπτυξης κώδικα όπως το Spyder, PyCharm κ.α.)
  - Απλά δημιουργήστε ένα αρχείο με κατάληξη .ipynb
- Υπάρχουν ελεύθερα διαθέσιμα Jupyter Notebooks που εκτελούνται σε υποδομές cloud (δεν απαιτούν τοπική εγκατάσταση):
  - Google Colab - https://colab.research.google.com/
  - Jupyter Lite - https://jupyter.org/try
    - https://jupyter.org/try-jupyter/lab/
  - Kaggle Notebooks - https://www.kaggle.com/code
- Μια μοντέρνα εκδοχή (reactive) notebooks: marimo - https://marimo.io/

Binder

- Το Binder (mybinder.org) είναι μια ελεύθερα διαθέσιμη cloud-υπηρεσία που επιτρέπει την εκτέλεση Jupyter Notebooks online, χωρίς τοπική εγκατάσταση
- Το binder:
  - δέχεται ένα GitHub αποθετήριο (repository) που περιέχει Jupyter Notebooks και ένα αρχείο διαμόρφωσης (configuration file), π.χ. requirements.txt, environment.yml
  - δημιουργεί ένα προσωρινό περιβάλλον (docker container) με όλες τις απαιτούμενες εξαρτήσεις
  - επιστρέφει μια διεπαφή (live Jupyter Notebook interface) που μπορεί να εκτελεστεί σε έναν φυλλομετρητή για την εκτέλεση υπολογισμών και την οπτικοποίηση αποτελεσμάτων
- Δείτε οδηγίες χρήσης του binder στο:
  - https://aaltoscicomp.github.io/python-for-scicomp/binder/

Anaconda

- Το Anaconda είναι μια διανομή της Python που περιλαμβάνει διάφορα εργαλεία και πολλά pre-built πακέτα
  - Εγκαθίσταται από το https://www.anaconda.com/download
  - Επιλέξτε εγκατάσταση σε διαδρομή χωρίς κενά (π.χ. c:\anaconda, μπορεί να χρειαστεί να εκτελέσετε την εγκατάσταση με δικαιώματα διαχειριστή), ενεργοποιήστε τις επιλογές:
    - Add Anaconda3 to my PATH environment variable
    - Register Anaconda3 as my default Python 3.x
- Η εγκατάσταση του Anaconda, εγκαθιστά το conda που είναι λογισμικό δημιουργίας απομονωμένων περιβαλλόντων ανάπτυξης και package manager (εναλλακτικό του venv και του pip)
- Μέσα από το Anaconda η έναρξη Jupyter notebook γίνεται με την επιλογή "Jupyter Notebook" στο μενού εφαρμογών που έχει δημιουργηθεί κατά την εγκατάσταση ή από τη γραμμή εντολών με την εντολή jupyter notebook

Εκδόσεις βιβλιοθηκών και έκδοση της Python

- Μπορεί να περάσουν μήνες πριν μια μεταγλωττισμένη βιβλιοθήκη όπως το NumPy γίνει διαθέσιμη στην τελευταία έκδοση της Python
- Χρησιμοποιήστε μια λίγο παλαιότερη έκδοση της Python

Δεδομένα

- Δεδομένα είναι μια συλλογή από αντικείμενα με χαρακτηριστικά / ιδιότητες / γνωρίσματα (attributes / properties / features)
- Τα χαρακτηριστικά μπορούν να λαμβάνουν:
  - Ονομαστικές (nominal) τιμές: ονόματα που απλά χρησιμοποιούνται για να διαχωρίσουν μια τιμή από μια άλλη τιμή (π.χ. φύλο, χρώμα ματιών, χώρα κ.α.), χωρίς να υποδηλώνουν σειρά
  - Διατεταγμένες (ordinal) τιμές: τιμές κατηγοριών για τις οποίες έχει νόημα η σειρά (π.χ., επίπεδο εκπαίδευσης, βαθμός ικανοποίησης κ.α.)
  - Αριθμητικές τιμές: αναπαριστούν ποσότητες που η σύγκρισή τους με άλλες ποσότητες έχει νόημα

Τύποι δεδομένων

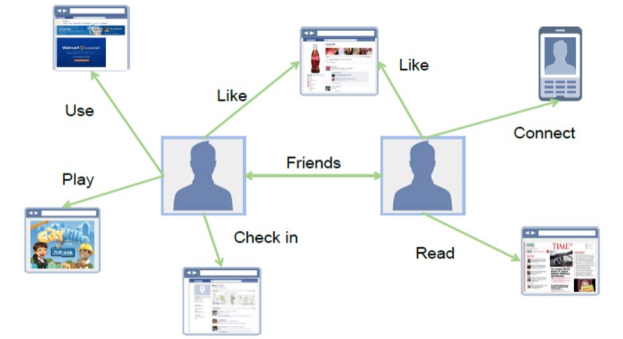
Εγγραφές Βάσεων Δεδομένων

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

Δοσοληψίες

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

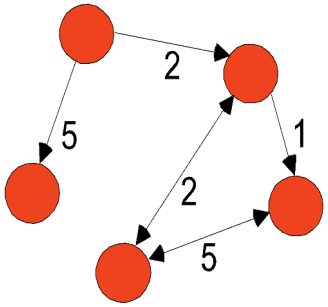
Κοινωνικά Δίκτυα



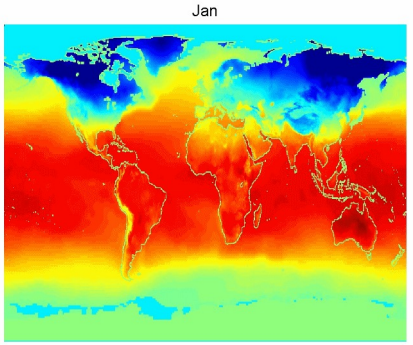
Αναπαράσταση κειμένου

	team	coach	play	ball	score	game	win	lost	timeout	season
Document 1	3	0	5	0	2	6	0	2	0	2
Document 2	0	7	0	2	1	0	0	3	0	0
Document 3	0	1	0	0	1	2	2	0	3	0

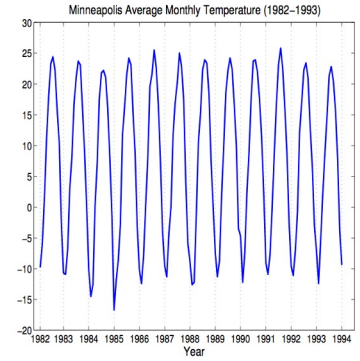
Δεδομένα γράφων



Χωροχρονικά



Χρονοσειρές



Αποθετήρια δεδομένων για πειραματισμό με δοκιμή αλγορίθμων ανάλυσης δεδομένων

- UCI machine learning repository - https://archive.ics.uci.edu/
- Scikit-Learn datasets
  - https://scikit-learn.org/stable/datasets/toy_dataset.html
  - https://scikit-learn.org/stable/datasets/real_world.html
- Seaborn datasets
  - https://www.kaggle.com/datasets/abdoomoh/all-seaborn-built-in-datasets
- OpenML datasets - https://www.openml.org/search?type=data&status=active
- Keras datasets - https://keras.io/api/datasets/
- Calmcode.io datasets - https://calmcode.io/datasets

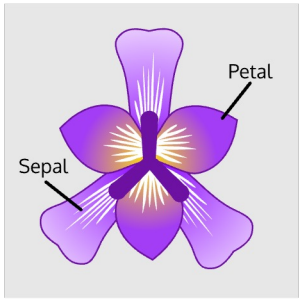
Σύνολα δεδομένων που θα χρησιμοποιήσουμε

- Iris flower data set
  - https://en.wikipedia.org/wiki/Iris_flower_data_set
  - https://scikit-learn.org/stable/datasets/toy_dataset.html#iris-plants-dataset
  - https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data
- The Boston Housing Dataset
  - https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html
  - https://lib.stat.cmu.edu/datasets/boston
  - https://www.kaggle.com/code/prasadperera/the-boston-housing-dataset
  - https://github.com/shridhar1504/Boston-House-Price-Prediction-Datascience-Project
- Titanic
  - https://seaborn.pydata.org/generated/seaborn.load_dataset.html
  - https://github.com/mwaskom/seaborn-data
  - https://www.kaggle.com/competitions/titanic
  - https://colab.research.google.com/drive/1EshABaip88itNX4vABJ1FHDmgRJSbanw

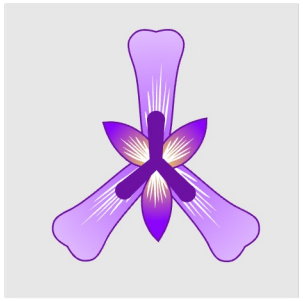
Iris dataset

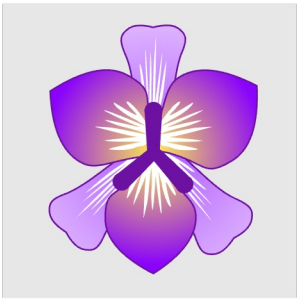
- Περιέχει μετρήσεις για 150 φυτά iris
- 3 διαφορετικά είδη με 50 φυτά για κάθε είδος
- Ιδιότητες που υπάρχουν στο Iris dataset:
  - species name (όνομα είδους)
  - sepal length
  - sepal width
  - petal length
  - petal width

```
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
...
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
...
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
...
```



Iris Versicolor


Iris Setosa


Iris Virginica

The Boston Housing Dataset

The Boston Housing Dataset

A Dataset derived from information collected by the U.S. Census Service concerning housing in the area of Boston Mass.

Delve

This dataset contains information collected by the U.S Census Service concerning housing in the area of Boston Mass. It was obtained from the StatLib archive (http://lib.stat.cmu.edu/datasets/boston), and has been used extensively throughout the literature to benchmark algorithms. However, these comparisons were primarily done outside of **Delve** and are thus somewhat suspect. The dataset is small in size with only 506 cases.

The data was originally published by Harrison, D. and Rubinfeld, D.L. `*Hedonic prices and the demand for clean air*', J. Environ. Economics & Management, vol.5, 81-102, 1978.

https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html

Dataset Naming

The name for this dataset is simply **boston**. It has two prototasks: **nox**, in which the nitrous oxide level is to be predicted; and **price**, in which the median value of a home is to be predicted

Miscellaneous Details

- **Origin**
  The origin of the boston housing data is **Natural**.
- **Usage**
  This dataset may be used for **Assessment**.
- **Number of Cases**
  The dataset contains a total of **506** cases.
- **Order**
  The order of the cases is **mysterious**.
- **Variables**
  There are **14** attributes in each case of the dataset. They are:
  1. CRIM - per capita crime rate by town
  2. ZN - proportion of residential land zoned for lots over 25,000 sq.ft.
  3. INDUS - proportion of non-retail business acres per town.
  4. CHAS - Charles River dummy variable (1 if tract bounds river; 0 otherwise)
  5. NOX - nitric oxides concentration (parts per 10 million)
  6. RM - average number of rooms per dwelling
  7. AGE - proportion of owner-occupied units built prior to 1940
  8. DIS - weighted distances to five Boston employment centres
  9. RAD - index of accessibility to radial highways
  10. TAX - full-value property-tax rate per $10,000
  11. PTRATIO - pupil-teacher ratio by town
  12. B - 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
  13. LSTAT - % lower status of the population
  14. MEDV - Median value of owner-occupied homes in $1000's

- **Note**
  Variable #14 seems to be censored at 50.00 (corresponding to a median price of $50,000); Censoring is suggested by the fact that the highest median price of exactly $50,000 is reported in 16 cases, while 15 cases have prices between $40,000 and $50,000, with prices rounded to the nearest hundred. Harrison and Rubinfeld do not mention any censoring.

Last Updated 10 October 1996
Comments and questions to: delve@cs.toronto.edu

Titanic dataset

Titanic

ID: 40945 verified ARFF Public 2017-10-16 v.1 Version history

Joaquin Vanschoren 3 likes 0 issues 45 downloads

Data Science History Statistics text_data

Description

Author: Frank E. Harrell Jr., Thomas Cason
Source: Vanderbilt Biostatistics
Please cite:

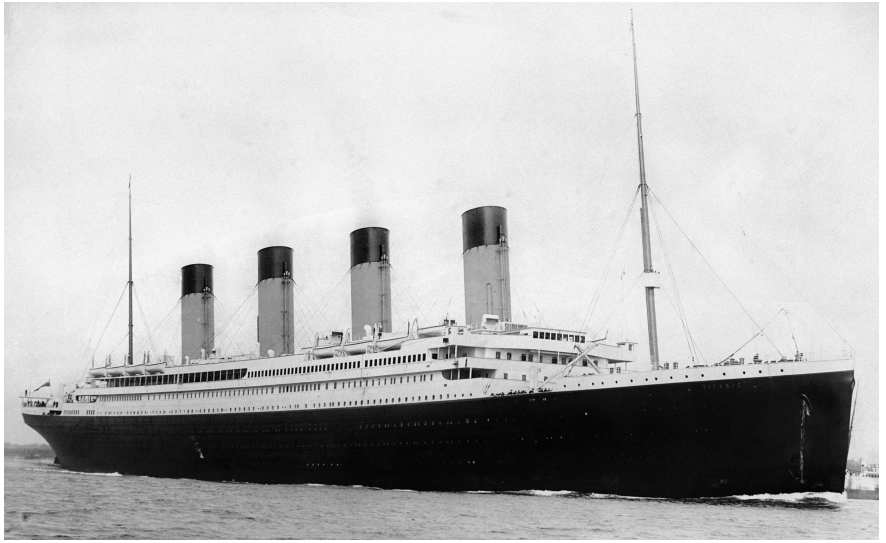
The original Titanic dataset, describing the survival status of individual passengers on the Titanic. The titanic data does not contain information from the crew, but it does contain actual ages of half of the passengers. The principal source for data about Titanic passengers is the Encyclopedia Titanica. The datasets used here were begun by a variety of researchers. One of the original sources is Eaton & Haas (1994) Titanic: Triumph and Tragedy, Patrick Stephens Ltd, which includes a passenger list created by many researchers and edited by Michael A. Findlay.

Thomas Cason of UVa has greatly updated and improved the titanic data frame using the Encyclopedia Titanica and created the dataset here. Some duplicate passengers have been dropped, many errors corrected, many missing ages filled in, and new variables created.

For more information about how this dataset was constructed: http://biostat.mc.vanderbilt.edu/wiki/pub/Main/DataSets/titanic3info.txt

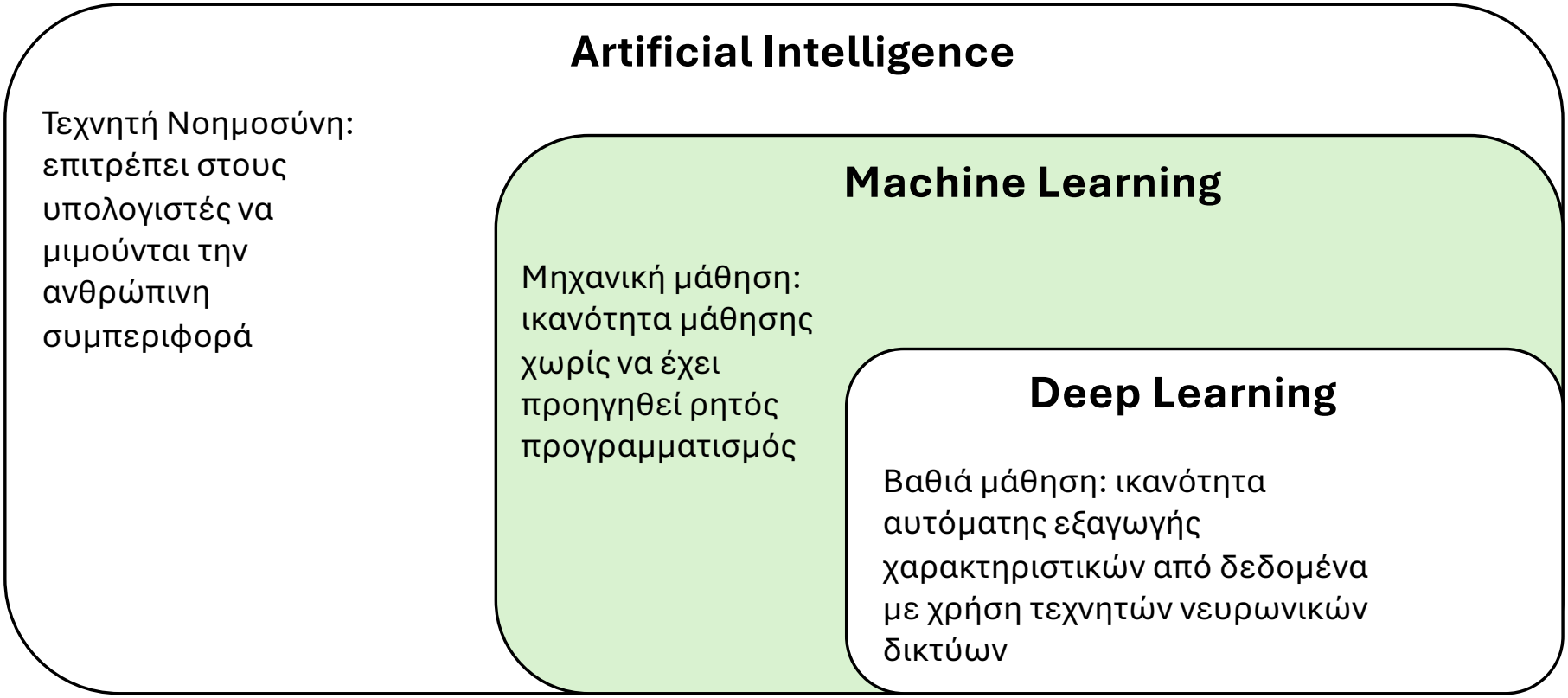
Attribute information

The variables on our extracted dataset are pclass, survived, name, age, embarked, home.dest, room, ticket, boat, and sex. pclass refers to passenger class (1st, 2nd, 3rd), and is a proxy for socio-economic class. Age is in years, and some infants had fractional values. The titanic2 data frame has no missing data and includes records for the crew, but age is dichotomized at adult vs. child. These data were obtained from Robert Dawson, Saint Mary's University, E-mail. The variables are pclass, age, sex, survived. These data frames are useful for demonstrating many of the functions in Hmisc as well as demonstrating binary logistic regression analysis using the Design library. For more details and references see Simonoff, Jeffrey S (1997): The "unusual episode" and a second statistics course. J Statistics Education, Vol. 5 No. 1.

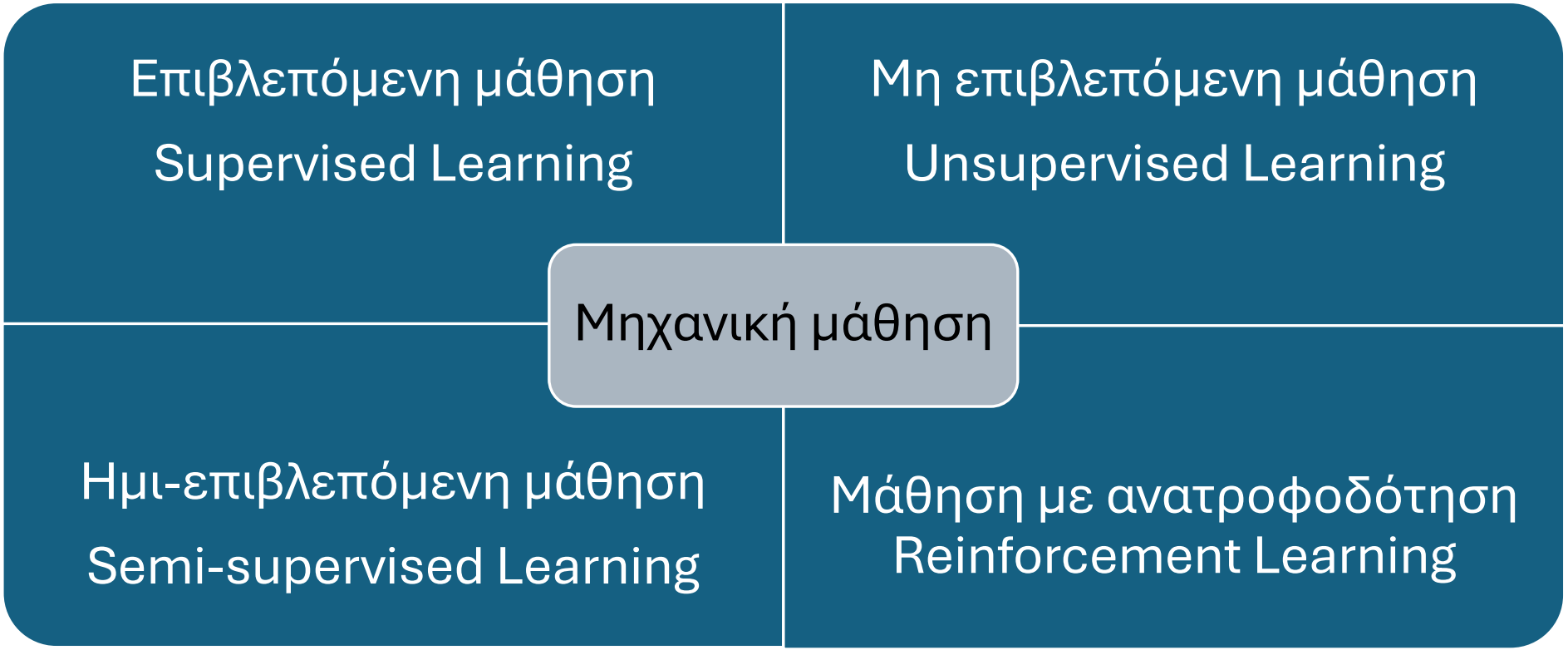


https://www.openml.org/search?type=data&sort=runs&id=40945&status=active

AI-ML-DL


Artificial Intelligence
Τεχνητή Νοημοσύνη: επιτρέπει στους υπολογιστές να μιμούνται την ανθρώπινη συμπεριφορά
Machine Learning
Μηχανική μάθηση: ικανότητα μάθησης χωρίς να έχει προηγηθεί ρητός προγραμματισμός
Deep Learning
Βαθιά μάθηση: ικανότητα αυτόματης εξαγωγής χαρακτηριστικών από δεδομένα με χρήση τεχνητών νευρωνικών δικτύων

Κατηγορίες αλγορίθμων μηχανικής μάθησης



Επιβλεπόμενη μάθηση και μη επιβλεπόμενη μάθηση

- Στην επιβλεπόμενη μάθηση το μοντέλο μαθαίνει από τα παραδείγματα
  - Τα δεδομένα εμφανίζονται σε ζεύγη **(X, y)**
  - **X** είναι οι λεγόμενες **ερμηνευτικές μεταβλητές** που αναφέρονται συχνά ως explanatory variables ή predictors ή features ή independent variables
  - **y** είναι η λεγόμενη **εξαρτημένη μεταβλητή** που αναφέρεται συχνά ως dependent variable ή response ή label ή target (μπορούν να υπάρχουν περισσότερες από 1 εξαρτημένες μεταβλητές)
- Στην μη-επιβλεπόμενη μάθηση εντοπίζονται πρότυπα στα δεδομένα
  - Δεν υπάρχει **εξαρτημένη μεταβλητή** στα δεδομένα εκπαίδευσης

Υποκατηγορίες προβλημάτων επιβλεπόμενης μάθησης

- **Παλινδρόμηση (regression)**: ανάθεση αριθμητικών τιμών σε αριθμητικά δεδομένα εισόδου
  - Παράδειγμα προβλήματος παλινδρόμησης: πρόβλεψη τιμών ακινήτων
  - Αλγόριθμοι: γραμμική παλινδρόμηση, πολυωνυμική παλινδρόμηση, νευρωνικά δίκτυα, κ.α.
- **Κατηγοριοποίηση (classification)**: ανάθεση ετικετών σε αριθμητικά δεδομένα εισόδου
  - Παράδειγμα προβλήματος κατηγοριοποίησης: αναγνώριση email ως κανονικού ή ως spam
  - Αλγόριθμοι: νευρωνικά δίκτυα, k-πλησιέστεροι γείτονες (k-NN), δένδρα αποφάσεων, λογιστική παλινδρόμηση, Naive Bayes, Support Vector Machines (SVM), κ.α.

Υποκατηγορίες προβλημάτων μη επιβλεπόμενης μάθησης

- **Συσταδοποίηση (clustering)**: αναγνώριση συστάδων για δεδομένα εισόδου
  - Παράδειγμα προβλήματος συσταδοποίησης: ομαδοποίηση πελατών μιας επιχείρησης με βάση τα χαρακτηριστικά τους
  - Αλγόριθμοι: K-Means, Hierarchical Clustering, DBScan, κ.α.
- **Μείωση διαστάσεων (dimension reduction)**: αντιστοίχιση σημείων ενός n-διάστατου χώρου σε ένα m-διάστατο χώρο όπου το m είναι σημαντικό μικρότερο του n
  - Παράδειγμα προβλήματος μείωσης διαστάσεων: οπτικοποίηση δεδομένων πελατών με πολλά χαρακτηριστικά (ηλικία, εισόδημα, συχνότητα αγορών κ.α.) σε 2 διαστάσεις
  - Αλγόριθμοι: Principal Components Analysis (PCA), t-SNE, κ.α.

Παλινδρόμηση

Απλή γραμμική παλινδρόμηση, παλινδρόμηση με πολλές ερμηνευτικές μεταβλητές, πολυωνυμική παλινδρόμηση, βελτίωση αποτελεσμάτων παλινδρόμησης

Γραμμική παλινδρόμηση

- Στη γραμμική παλινδρόμηση (linear regression) υποθέτουμε ότι η εξαρτημένη μεταβλητή μπορεί να εκφραστεί ως γραμμικός συνδυασμός ενός αριθμού ερμηνευτικών μεταβλητών
- Στη γραμμική παλινδρόμηση μια γραμμική συνάρτηση προσαρμόζεται (γίνεται fit) σε γνωστά σημεία δεδομένων, συνήθως χρησιμοποιώντας τη μέθοδο ελαχίστων τετραγώνων (least squares)
  - OLS (Ordinary Least Squares): το άθροισμα των τετραγώνων των σφαλμάτων (αποκλίσεων) θα πρέπει να είναι το ελάχιστο δυνατό
- Η εφαρμογή της γραμμικής παλινδρόμησης με το scikit-learn είναι εύκολη υπόθεση
- Εναλλακτικά, για τη γραμμική παλινδρόμηση, μπορούν να χρησιμοποιηθούν άλλες βιβλιοθήκες όπως η statsmodel

Πως λειτουργεί η OLS;

- Η OLS (Ordinary Least Squares) εντοπίζει τη γραμμή (ή το υπερεπίπεδο) που ελαχιστοποιεί το άθροισμα των διαφορών στο τετράγωνο μεταξύ παρατηρήσεων y_i και προβλέψεων $\hat{y}_i$
- Τυπικά, ελαχιστοποιεί το:

$$SSE = \sum_{i=1}^{n} (y_i - (b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_p x_{ip}))^2$$

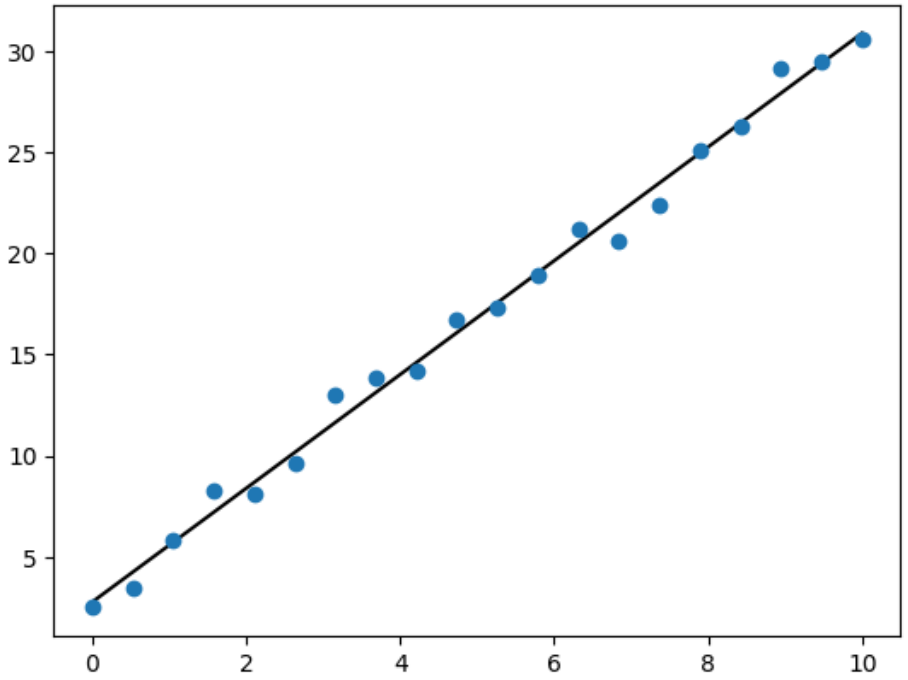
- Υποθέτει γραμμικότητα, ανεξαρτησία μεταβλητών, σταθερή διακύμανση (ομοσκεδαστικότητα) και σφάλματα που ακολουθούν την κανονική κατανομή
- Για τον υπολογισμό των τιμών τελικά λύνεται ένα σύστημα εξισώσεων

Απλή γραμμική παλινδρόμηση

- Στην περίπτωση που η ερμηνευτική μεταβλητή είναι μόνο μια, όπως και η εξαρτημένη μεταβλητή, η γραμμική παλινδρόμηση ονομάζεται απλή γραμμική παλινδρόμηση (simple linear regression)
- Η οπτικοποίηση της απλής γραμμικής παλινδρόμησης είναι εύκολη καθώς εμπλέκονται 2 μόνο μεταβλητές

Απλή γραμμική παλινδρόμηση

- Στο παράδειγμα αυτό θα χρησιμοποιήσουμε τη γραμμική σχέση **y = 3x+2** και για διάφορες τιμές του x στο διάστημα [0,10] θα δημιουργήσουμε τιμές του y που βρίσκονται κοντά στην τιμή που θα είχε το y (δηλαδή, θα προσθέσουμε λίγο θόρυβο)
- Ο σκοπός είναι η απλή γραμμική παλινδρόμηση, που θα εφαρμοστεί στη συνέχεια, να υπολογίσει για τις παραμέτρους της σχέσης **y = ax+b**, τιμές για τις παραμέτρους a και b που θα βρίσκονται κοντά στο 3 και 2 αντίστοιχα



https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_simple_linear_regression.ipynb

Άσκηση 1: Απλή γραμμική παλινδρόμηση

- Δίνονται τα ακόλουθα σημεία: (1,2), (2,3), (3,5), (4,7), (5,11)
- Σχεδιάστε τα σημεία σε ένα γράφημα με το matplotlib
- Πραγματοποιήστε απλή γραμμική παλινδρόμηση
- Εκτυπώστε τους συντελεστές παλινδρόμησης
- Σχεδιάστε τη γραμμή παλινδρόμησης

Γραμμική παλινδρόμηση με πολλαπλές ερμηνευτικές μεταβλητές (multiple features)

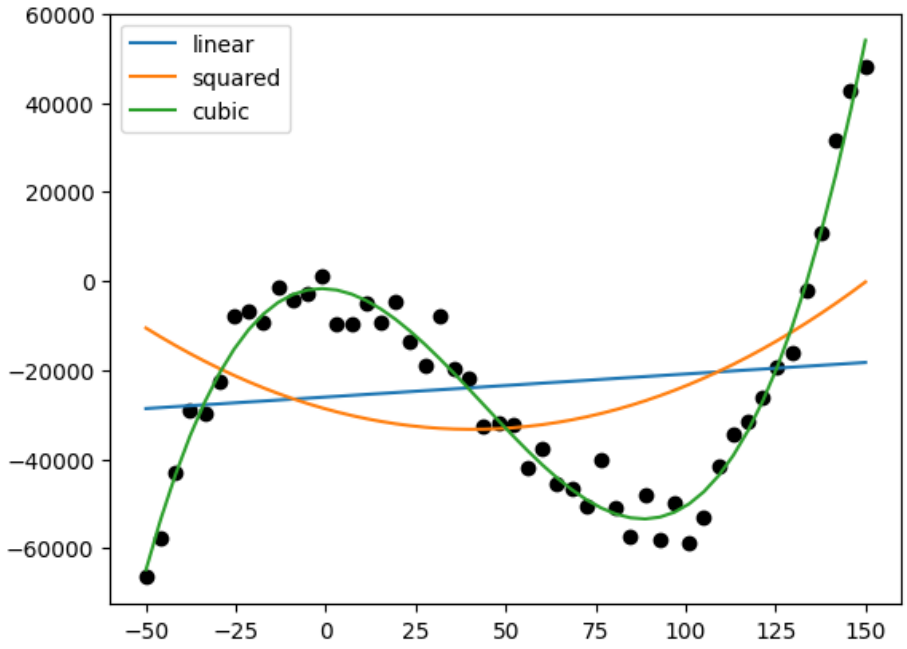
- Οι ερμηνευτικές μεταβλητές μπορεί να είναι πολλές (x_1,x_2, …, x_n)
- Σε αυτή την περίπτωση προκύπτει μια γραμμική σχέση της μορφής: $y = a_1x_1 + a_2x_2 + \ldots + a_nx_n + b$
- Η παλινδρόμηση γίνεται με τον ίδιο τρόπο, όπως και στην απλή γραμμική παλινδρόμηση, από το scikit-learn, απλά οι ερμηνευτικές μεταβλητές είναι ένας πίνακας m x n με m το πλήθος των δειγμάτων και n το πλήθος των ερμηνευτικών μεταβλητών

Άσκηση 2: Γραμμική παλινδρόμηση με πολλαπλές μεταβλητές

- Δίνονται τα ακόλουθα δεδομένα:
  - 0 λίτρο γάλα, 0 κιλό ψωμί: 0 ευρώ
  - 1 λίτρο γάλα, 1 κιλό ψωμί: 5 ευρώ
  - 2 λίτρα γάλα, 3 κιλά ψωμί: 13,5 ευρώ
  - 3 λίτρα γάλα, 2 κιλά ψωμί: 10.90 ευρώ
- Πραγματοποιήστε γραμμική παλινδρόμηση και υπολογίστε το προβλεπόμενο κόστος για 4 λίτρα γάλα και 2 κιλά ψωμί

Πολυωνυμική παλινδρόμηση (Polynomial Regression)

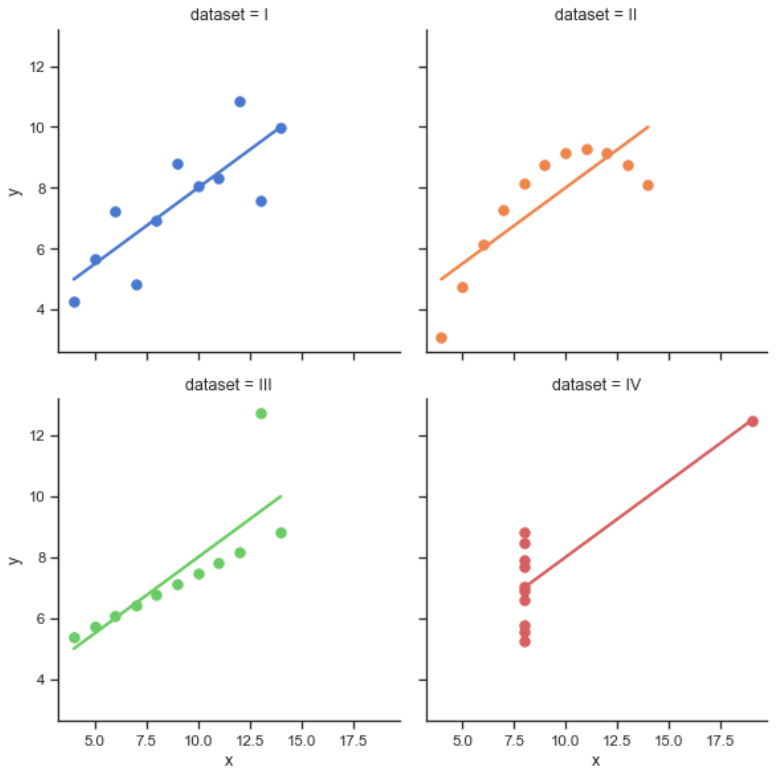
- Σε ορισμένα δεδομένα δεν ταιριάζει μια γραμμική σχέση της μορφής **y = a*x+b**
- Αντί για γραμμική σχέση, μπορεί να υποτεθεί πολυωνυμική σχέση μεταξύ των ερμηνευτικών μεταβλητών και της εξαρτημένης μεταβλητής, για παράδειγμα:
  - **y = a*x^2 + b*x + c**
  - **y = a*x^3 + b*x^2 + c*x + d**
- Το scikit-learn διαθέτει έναν preprocessor που ονομάζεται PolynomialFeatures
  - from sklearn.preprocessing import PolynomialFeatures



https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_polynomial_regression.ipynb
https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_polynomial_regression2.ipynb

Άσκηση 3: Πολυωνυμική παλινδρόμηση

- Φορτώστε το dataset II από το https://seaborn.pydata.org/examples/anscombes_quartet.html
- Εφαρμόστε πολυωνυμική παλινδρόμηση



Γραμμική παλινδρόμηση σε ένα μεγαλύτερο παράδειγμα

- Θα εξετάσουμε το παράδειγμα Boston House Pricing
- Τα δεδομένα βρίσκονται στο https://lib.stat.cmu.edu/datasets/boston

```
price = predict_price(model, CRIM=0.05, ZN=18.0, INDUS=2.31, CHAS=0, NOX=0.54,
                      RM=6.8, AGE=60.0, DIS=4.0, RAD=1, TAX=296, PTRATIO=15.3,
                      B=390.5, LSTAT=5.0)
print(price
```

1. Φόρτωση δεδομένων
2. Μετασχηματισμός δεδομένων σε dataframe με 13 στήλες για τις ερμηνευτικές μεταβλητές και 1 στήλη για την εξαρτημένη μεταβλητή (τιμή σπιτιού)
3. Διαχωρισμός δεδομένων (506 δειγμάτων) σε 80% train και 20% test
4. Δημιουργία μοντέλου γραμμικής παλινδρόμησης και εκπαίδευση του στο σύνολο train
5. Εφαρμογή του μοντέλου στα δείγματα που έχουν διατηρηθεί στο test
6. Αποτίμηση απόδοσης με τις μετρικές MSE (Mean Square Error) και R^2 (R-Squared)
7. Εκτύπωση συντελεστών παλινδρόμησης
8. Εκτίμηση τιμής για ένα "νέο" σπίτι, συγγραφή συνάρτησης predict_price που δέχεται ως ορίσματα το μοντέλο και ένα **kwargs με όσες ερμηνευτικές μεταβλητές γνωρίζουμε (στις υπόλοιπες συμπληρώνει τιμή μηδέν)

https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_LR_boston.ipynb

Πως θα μπορούσε να βελτιωθεί η πρόβλεψη στο προηγούμενο παράδειγμα;

- Μελέτη των δεδομένων
- Αφαίρεση ή μετασχηματισμός ακραίων τιμών
- Κλιμάκωση μεταβλητών (scaling)
- Εφαρμογή κανονικοποίησης (regularization) Lasso ή Ridge
- Καλύτερη αξιολόγηση με Cross-Validation
- Δοκιμή ισχυρότερων αλγορίθμων (τα μη γραμμικά μοντέλα μπορούν να περιγράψουν περίπλοκες σχέσεις)

Σχόλιο για τα δεδομένα Boston House Pricing

- Η τιμή MEDV (median value of owner-occupied homes) μετράται σε χιλιάδες δολάρια
- Ωστόσο, όλες οι τιμές πάνω από $50,000 έχουν περιοριστεί τεχνητά στο 50.0
- Το αποτέλεσμα είναι μια εξαρτημένη μεταβλητή λογοκριμένη (censored) προς τα πάνω
- Οδηγεί σε υποεκτίμηση τιμών για τα υψηλής αξίας σπίτια

- Τρόποι αντιμετώπισης:
  - Αφαίρεση δειγμάτων με MEDV = 50.0
  - Εφαρμογή μοντέλου που είναι ανθεκτικότερο σε λογοκριμένα δεδομένα (π.χ. tobit regression που υπάρχει στο statsmodels)

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
161	1.46336	0.0	19.58	0.0	0.6050	7.489	90.8	1.9709	5.0	403.0	14.7	374.43	1.73	50.0
162	1.83377	0.0	19.58	1.0	0.6050	7.802	98.2	2.0407	5.0	403.0	14.7	389.61	1.92	50.0
163	1.51902	0.0	19.58	1.0	0.6050	8.375	93.9	2.1620	5.0	403.0	14.7	388.45	3.32	50.0
166	2.01019	0.0	19.58	0.0	0.6050	7.929	96.2	2.0459	5.0	403.0	14.7	369.30	3.70	50.0
186	0.05602	0.0	2.46	0.0	0.4880	7.831	53.6	3.1992	3.0	193.0	17.8	392.63	4.45	50.0
195	0.01381	80.0	0.46	0.0	0.4220	7.875	32.0	5.6484	4.0	255.0	14.4	394.23	2.97	50.0
204	0.02009	95.0	2.68	0.0	0.4161	8.034	31.9	5.1180	4.0	224.0	14.7	390.55	2.88	50.0
225	0.52693	0.0	6.20	0.0	0.5040	8.725	83.0	2.8944	8.0	307.0	17.4	382.00	4.63	50.0
257	0.61154	20.0	3.97	0.0	0.6470	8.704	86.9	1.8010	5.0	264.0	13.0	389.70	5.12	50.0
267	0.57834	20.0	3.97	0.0	0.5750	8.297	67.0	2.4216	5.0	264.0	13.0	384.54	7.44	50.0
283	0.01501	90.0	1.21	1.0	0.4010	7.923	24.8	5.8850	1.0	198.0	13.6	395.52	3.16	50.0
368	4.89822	0.0	18.10	0.0	0.6310	4.970	100.0	1.3325	24.0	666.0	20.2	375.52	3.26	50.0
369	5.66998	0.0	18.10	1.0	0.6310	6.683	96.8	1.3567	24.0	666.0	20.2	375.33	3.73	50.0
370	6.53876	0.0	18.10	1.0	0.6310	7.016	97.5	1.2024	24.0	666.0	20.2	392.05	2.96	50.0
371	9.23230	0.0	18.10	0.0	0.6310	6.216	100.0	1.1691	24.0	666.0	20.2	366.15	9.53	50.0
372	8.26725	0.0	18.10	1.0	0.6680	5.875	89.6	1.1296	24.0	666.0	20.2	347.88	8.88	50.0

Μετρικές για γραμμική παλινδρόμηση

- MSE: μέσο τετραγωνικό σφάλμα
- RMSE: τετραγωνική ρίζα του MSE
- R^2: συντελεστής προσδιορισμού (coefficient of determination) ή R squared
  - Πόση από τη διακύμανση της εξαρτημένης μεταβλητής μπορεί να εξηγηθεί από τις ανεξάρτητες μεταβλητές;
  - Εκφράζει το σχετικό σφάλμα λαμβάνοντας τιμές από το 0 μέχρι το 1
  - 1 = τέλεια παρεμβολή (interpolation)
  - 0 = η παρεμβολή είναι το ίδιο καλή με το έχουμε χρησιμοποιήσει το μέσο όρο των δεδομένων
  - <0 = η παρεμβολή είναι χειρότερη από το αν είχαμε χρησιμοποιήσει το μέσο όρο των δεδομένων
- Επιπλέον μετρικές:
  - F-statistic
  - BIC (Bayesian Information Criterion)
  - AIC (Akaike Information Criterion)

Κανονικοποίηση

- Η τεχνική της κανονικοποίησης (Regularization) χρησιμοποιείται στην παλινδρόμηση για την αποφυγή μεγάλων συντελεστών
- L_1 regularization (Lasso): οι απόλυτες τιμές των συντελεστών λαμβάνουν ποινή
- L_2 regularization (Ridge): τα τετράγωνα τιμών των συντελεστών λαμβάνουν ποινή

Άσκηση 4

- Στο πρόβλημα Boston House Pricing αφαιρέστε τα δείγματα που έχουν MEDV = 50.0
- Υπολογίστε τις μετρικές R^2 και MSE ξανά
- Εφαρμόστε scaling στις ερμηνευτικές μεταβλητές με το StandardScaler όπως φαίνεται στον κώδικα δεξιά, τι αποτελέσματα λαμβάνετε;

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Κατηγοριοποίηση

Κατηγοριοποίηση Naive Bayes, θεώρημα του Bayes, μετρικές προβλημάτων κατηγοριοποίησης, δένδρα απόφασης, λογιστική παλινδρόμηση, k-πλησιέστεροι γείτονες, Support Vector Machines

Κατηγοριοποίηση Naive Bayes

- Ο αλγόριθμος Naive Bayes classification είναι ένας απλός και γρήγορος αλγόριθμος κατηγοριοποίησης
- Η ταχύτητα του οφείλεται σε απλουστεύσεις για τις υποκείμενες κατανομές πιθανοτήτων και ιδιαίτερα σε ότι αφορά τη υπόθεση περί ανεξαρτησίας των ερμηνευτικών μεταβλητών
- Ειδικά όταν το πλήθος των ερμηνευτικών μεταβλητών είναι μεγάλο, μπορεί να επιστρέφει πολύ καλά αποτελέσματα
- Αλγόριθμοι όπως ο Naive Bayes που κάνουν κατηγοριοποίηση, ονομάζονται κατηγοριοποιητές ή ταξινομητές (classifiers)

Πως λειτουργεί ο Naive Bayes;

- Από τα δεδομένα του προβλήματος γνωρίζουμε για κάθε τιμή L (ετικέτα) της εξαρτημένης μεταβλητής που θέλουμε να προβλέψουμε, μια κατανομή πιθανότητας
- Δηλαδή γνωρίζουμε για κάθε πιθανό συνδυασμό ερμηνευτικών μεταβλητών την πιθανότητα να παρατηρηθεί η τιμή L

$$P(features|L)$$

- Η βασική ιδέα του Naive Bayes είναι να αντιστραφεί η κατεύθυνση της εξάρτησης, θέλουμε δηλαδή να προβλέπουμε την ετικέτα με βάση τις ερμηνευτικές μεταβλητές

$$P(L|features)$$

- Τη δυνατότητα αυτή μας τη δίνει το θεώρημα του Bayes

Θεώρημα του Bayes

- Το θεώρημα του Bayes περιγράφει πως μπορεί να ενημερωθεί η πιθανότητα ενός γεγονότος Α δεδομένης της εμφάνισης ενός άλλου γεγονότος Β
- P(A|B) είναι η υπό συνθήκη πιθανότητα του Α, δεδομένου ότι έχει συμβεί το Β
- P(B|A) είναι η υπό συνθήκη πιθανότητα του Β, δεδομένου ότι έχει συμβεί το Α
- P(A) είναι η απλή πιθανότητα του Α
- P(B) είναι η απλή πιθανότητα του Β

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Παράδειγμα

- Οι 3 στους 4 νέους (75%) κάτω των 30 είναι άνεργοι.
- Από τα δημογραφικά στοιχεία γνωρίζουμε ότι:
  - Το ποσοστό ανεργίας είναι 37% για όλες τις ηλικίες που θα μπορούσαν να εργάζονται
  - Οι Έλληνες κάτω των 30 είναι το 35% του πληθυσμού που μπορεί να εργάζεται
- Από τα 200 άτομα που βρίσκονται στο ταμείο ανεργίας πόσοι είναι νέοι κάτω από 30 ετών;

- Λύση
  - Έστω Α το ενδεχόμενο να είναι κάποιος κάτω από 30 έτη => P(A)=0.35
  - Έστω Β το ενδεχόμενο κάποιος να είναι άνεργος => P(B)=0.37
  - Η πιθανότητα να είναι κάποιος άνεργος δεδομένου ότι είναι νέος είναι P(B|A) = 0.75
  - Άρα, η πιθανότητα να είναι κάποιος νέος δεδομένου ότι είναι άνεργος είναι $P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{0.35*0.75}{0.37} = 0.71$
  - Συνεπώς, από τα 200 άτομα στο ταμείο ανεργίας οι 71*200=142 αναμένεται να είναι άνεργοι

Naive Bayes Classification

- Συνεπώς, από το θεώρημα του Bayes προκύπτει:

$$P(L|features) = \frac{P(features|L)P(L)}{P(features)}$$

- Αν υποθέσουμε ότι έχουμε δύο ετικέτες L1 και L2 και γνωρίζουμε τα P(features|L1) και P(features|L2)
- Έστω ένα νέο δείγμα για το οποίο γνωρίζουμε τα features (τιμές ερμηνευτικών μεταβλητών) αλλά δεν γνωρίζουμε την ετικέτα του

- Μπορούμε να επιχειρήσουμε να προβλέψουμε την ετικέτα του νέου δείγματος χρησιμοποιώντας τους λόγους των εκ των υστέρων πιθανοτήτων

$$\frac{P(L1|features)}{P(L2|features)} = \frac{P(features|L1)P(L1)}{P(features|L2)P(L2)}$$

- Αν το κλάσμα είναι μεγαλύτερο του 1 τότε το νέο δείγμα λαμβάνει την ετικέτα L1, ενώ διαφορετικά λαμβάνει την ετικέτα L2
- Προσέξτε ότι οι τιμές των P(features|L1), P(features|L2), P(L1) και P(L2) υπάρχουν όλες στα δεδομένα εισόδου που διαθέτουμε

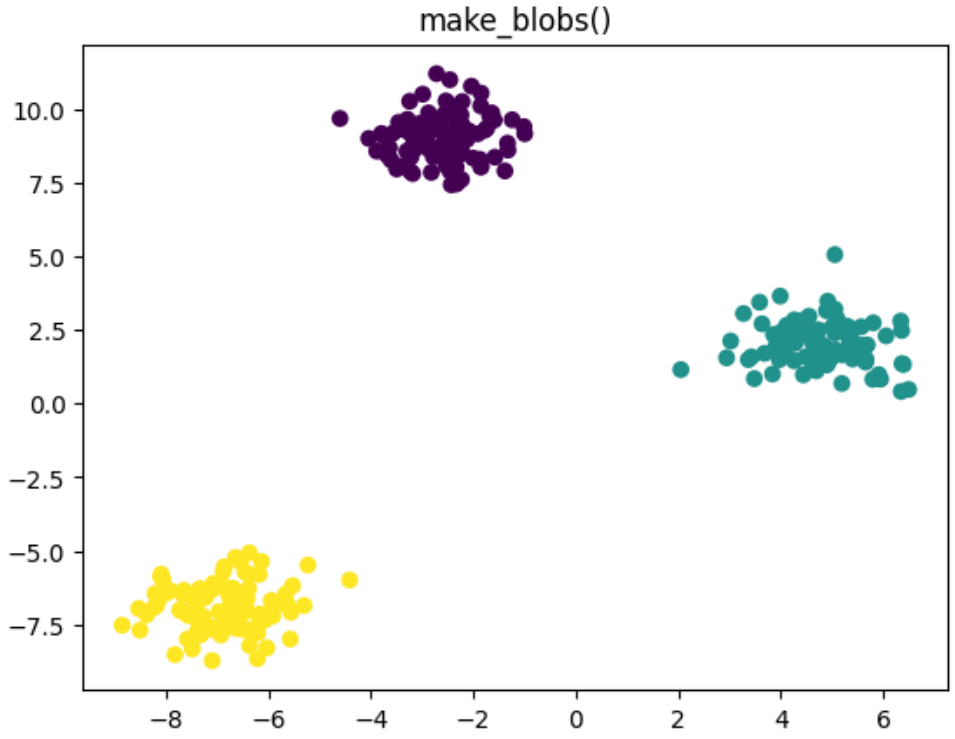
Δημιουργία συνθετικών δεδομένων για κατηγοριοποίηση

- Το scikit-learn διαθέτει τη συνάρτηση make_blobs που δημιουργεί συνθετικά δεδομένα για τη δοκιμή αλγορίθμων κατηγοριοποίησης και συσταδοποίησης
- Αυξήστε την τιμή του cluster_std στον ακόλουθο κώδικα, ποιο θα είναι το αποτέλεσμα;

```python
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# Δημιουργία 3 συστάδων σημείων 2D
X, y = make_blobs(n_samples=300, centers=3, n_features=2,
cluster_std=0.8, random_state=42)

plt.scatter(X[:, 0], X[:, 1], c=y, cmap='viridis')
plt.title("make_blobs()")
plt.show()
```



Παράδειγμα εφαρμογής της Naive Bayes σε συνθετικά δεδομένα

- Στο παράδειγμα αυτό θα χρησιμοποιήσουμε τη make_blobs για να δημιουργήσουμε 100 δείγματα καλά διαχωρισμένα μεταξύ τους (cluster_std=1.0) με 2 features και 1 label (που θα μπορεί να λάβει 1 από 2 τιμές) για κάθε δείγμα
- Θα απεικονίσουμε γραφικά τα δείγματα με το matplotlib
- Θα εφαρμόσουμε τον αλγόριθμο GaussianNB από το sklearn.naive_bayes
- Τέλος, θα υπολογίσουμε την ακρίβεια των προβλέψεων με το from accuracy_score από το sklearn.metrics καλώντας την accuracy_score(y, y_fitted)
- Γιατί το accuracy είναι 100%;

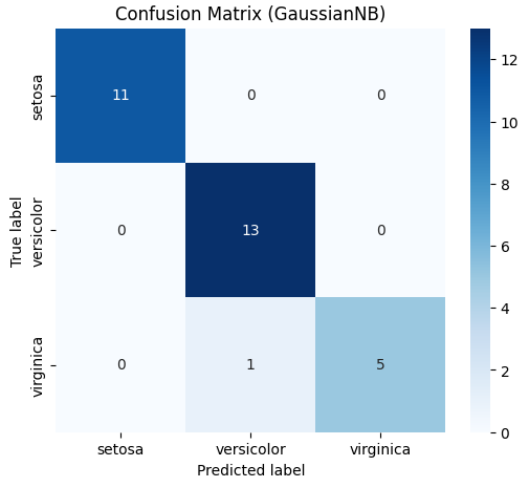
https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_GaussianNB.ipynb

Άσκηση 5

- Γράψτε κώδικα που να δημιουργεί, με την make_blobs, 1000 τυχαία δείγματα με 4 features και 1 label με 3 πιθανές τιμές και να τοποθετεί τα features σε έναν πίνακα X και τα labels σε ένα διάνυσμα y
- Πραγματοποιήστε πρόβλεψη με GaussianNB και χρησιμοποιήστε τη συνάρτηση train_test_split για να διαχωρίσετε τα δείγματα σε training set και testing set με 70% και 30% αντίστοιχα
- Ορίστε όπου χρειάζεται seed=7 για επαναληψιμότητα στην τυχαία επιλογή
- Ποιο είναι το accuracy;

Κατηγοριοποίηση του Iris dataset με το GaussianNB

- Φόρτωση του iris_dataset από το scikit-learn και επισκόπηση
- Διαχωρισμός των δειγμάτων σε training set και testing set χρησιμοποιώντας το train_test_split έτσι ώστε το training set να είναι 80% των συνολικών δεδομένων (random_state=0 για επαναληψιμότητα)
- Εφαρμογή του GaussianNB
- Πρόβλεψη ετικετών για τα δεδομένα testing
- Εύρεση και εκτύπωση των μετρικών:
  - confusion matrix
  - accuracy
  - precision
  - recall
  - F1-score
- Σχεδίαση heatmap για το confusion matrix με το seaborn



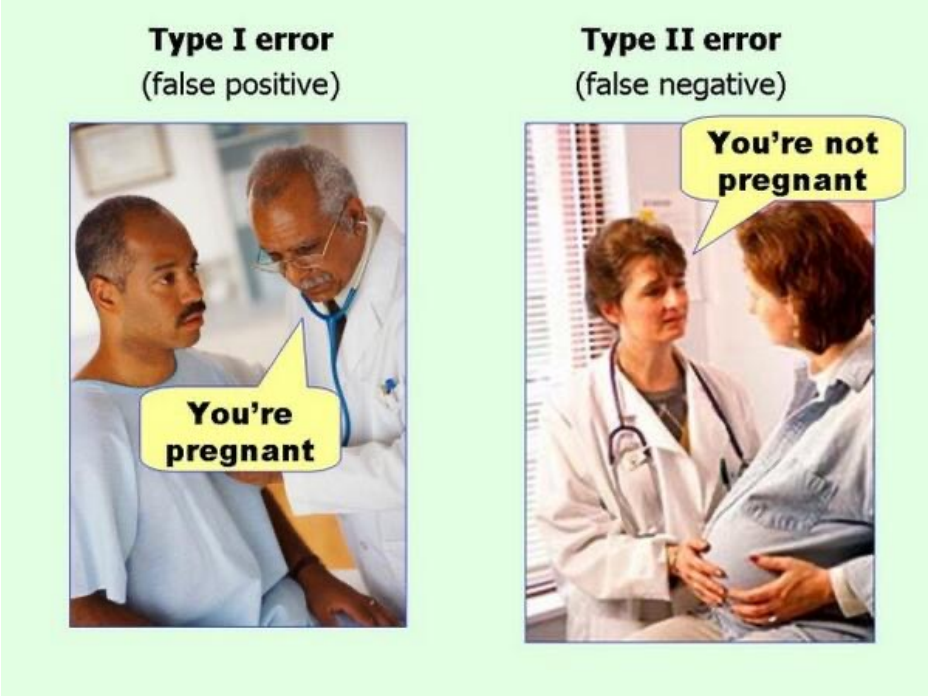
https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_GaussianNB_iris.ipynb

Μετρικές για προβλήματα κατηγοριοποίησης

- Μετρικές ακρίβειας
  - accuracy (ορθότητα)
  - confusion matrix (πίνακας σφαλμάτων)
- Μετρικές που βασίζονται σε True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN)
  - precision (ακρίβεια)
  - recall (ανάκληση)
  - f-score
  - ROC
  - AUC
- Πιθανοτικές μετρικές
  - cross entropy

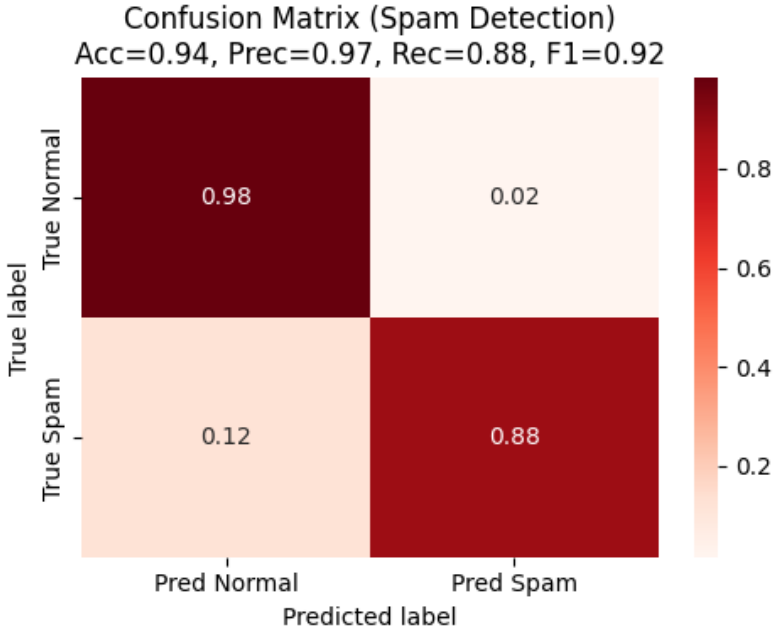
Μετρικές που βασίζονται σε TP, TN, FP, FN

- Παράδειγμα: αναγνώριση αν ένα μήνυμα είναι SPAM
  - TP (True Positive): ένα spam μήνυμα κατηγοριοποιείται ως spam
  - TN (True Negative): ένα κανονικό μήνυμα κατηγοριοποιείται ως κανονικό
  - FP (False Positive): ένα κανονικό μήνυμα κατηγοριοποιείται ως spam (type I error)
  - FN (False Negative): ένα spam μήνυμα κατηγοριοποιείται ως κανονικό (type II error)



Μετρικές που βασίζονται σε TP, TN, FP, FN: precision, recall, F-score, F1-score

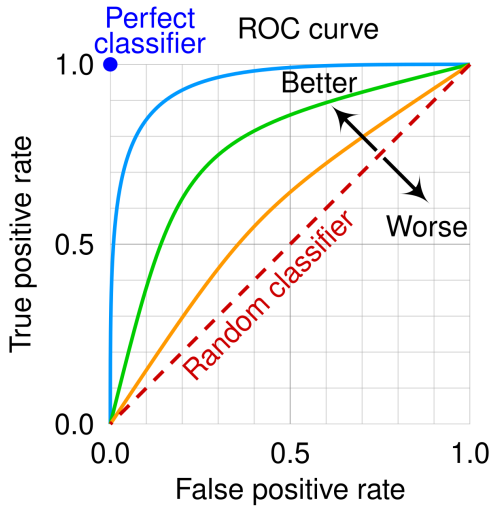
- Παράδειγμα:
  - 60 κανονικά μηνύματα, 40 μηνύματα spam
  - 1 κανονικό μήνυμα κατηγοριοποιήθηκε εσφαλμένα ως spam
  - 5 spam μηνύματα κατηγοριοποιήθηκαν εσφαλμένα ως κανονικά



- **precision**: 35/36=0,97 (35 από τα 36 μηνύματα που κατηγοριοποιήθηκαν ως spam ήταν πράγματι spam)
- **recall**: 35/40 = 0.88 (35 από τα 40 μηνύματα spam αναγνωρίστηκαν ως spam)
- **F-score**: αρμονικός μέσος των precision και recall $F_\beta = (1+\beta^2)\cdot\frac{\text{Precision}\cdot\text{Recall}}{(\beta^2\cdot\text{Precision})+\text{Recall}}$
- **F1-score**: $F_1 = 2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$
- Το precision και το recall έχουν διαφορετική βαρύτητα ανάλογα με την περίπτωση
- Για παράδειγμα η κατηγοριοποίηση ενός κανονικού μηνύματος ως spam είναι σημαντικό να μη συμβεί, συνεπώς η μετρική precision είναι πολύ σημαντική

Μετρικές που βασίζονται σε TP, TN, FP, FN: ROC, AUC

- ROC (Receiver Operating Characteristic)
- Χρησιμοποιείται για την αξιολόγηση δυαδικών κατηγοριοποιητών (binary classifiers)
- Το ROC είναι μια μετρική που αναπαριστά τη σχέση ανάμεσα σε True Positives και False Positives



- Το ROC αναπαρίσταται ως μια καμπύλη
- Όσο μεγαλύτερη είναι η περιοχή κάτω από την καμπύλη (AUC = Area Under the Curve), τόσο καλύτερη είναι η κατηγοριοποίηση
- Το AUC είναι μια τιμή που δείχνει πόσο καλά συμπεριφέρεται το μοντέλο:
  - AUC = 1.0: τέλειος κατηγοριοποιητής
  - AUC = 0.5: κατηγοριοποιητής ισοδύναμος με τυχαία επιλογή

Εφαρμογή πολλαπλών αλγορίθμων κατηγοριοποίησης στο πρόβλημα Titanic

Αλγόριθμοι Κατηγοριοποίησης

- GaussianNB
- DecisionTreeClassifier
- Logistic Regression
- KNN (χρειάζεται scaling)
- SVM (χρειάζεται scaling)

Η μέθοδος predict_proba() υπάρχει σε πολλούς αλγορίθμους κατηγοριοποίησης του scikit-learn και επιστρέφει την πιθανότητα να προβλεφθεί η κάθε ετικέτα αντί να επιστρέφει μόνο την προβλεπόμενη ετικέτα όπως η predict()

Βήματα που θα ακολουθηθούν

1. Φόρτωση δεδομένων προβλήματος από το seaborn
2. Διατήρηση μόνο των ερμηνευτικών μεταβλητών "age", "sex", "fare", "class", "survived"
3. Διαγραφή δειγμάτων που δεν έχουν τιμή (στη θέση της τιμής βρίσκεται NaN) σε κάποια από τις ερμηνευτικές μεταβλητές
4. Μετατροπή κατηγορικών μεταβλητών ("sex", "class") σε αριθμητικές
5. Ομαλοποίηση (scaling) με το StandardScaler
6. Διαχωρισμός σε training set και testing set
7. Για κάθε αλγοριθμο κατηγοριοποίησης:
   I. Δημιουργία μοντέλου πρόβλεψης και εκπαίδευση
   II. Λήψη προβλέψεων για το testing set
   III. Λήψη μετρικών accuracy, precision, recall, F1-Score
   IV. Σχεδίαση ROC, υπολογισμός AUC
8. Σύγκριση μοντέλων

https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_classification_titanic.ipynb

Δένδρο απόφασης

- Ένας ταξινομητής Δέντρου Απόφασης είναι ένας αλγόριθμος επιβλεπόμενης μάθησης που προβλέπει την κατηγορία ενός δείγματος μέσω διαδοχικών διαχωρισμών των δεδομένων με βάση τις τιμές των χαρακτηριστικών. Δημιουργεί μια δενδρική δομή όπου:
  - οι εσωτερικοί κόμβοι αντιστοιχούν σε ερωτήσεις/διαχωρισμούς
  - τα κλαδιά σε αποτελέσματα
  - και τα φύλλα σε τελικές κατηγορίες
- Στόχος: μεγιστοποίηση της «καθαρότητας» των κόμβων (π.χ. ελαχιστοποίηση Gini ή εντροπίας)
- Πλεονεκτήματα: εύκολη ερμηνεία, χειρίζεται αριθμητικά και κατηγορικά δεδομένα
- Μειονέκτημα: τάση για υπερεκπαίδευση (overfitting) αν δεν υπάρξει περιορισμός

Δένδρα απόφασης στο scikit-learn

- sklearn.tree.DecisionTreeClassifier — βασικό δέντρο απόφασης
- sklearn.ensemble.RandomForestClassifier — σύνολο πολλών δέντρων (bagging)
- sklearn.ensemble.ExtraTreesClassifier — δέντρα με επιπλέον τυχαιότητα για μείωση διασποράς
- sklearn.ensemble.GradientBoostingClassifier / sklearn.ensemble.HistGradientBoostingClassifier — σύνολα δέντρων με τεχνική boosting

Λογιστική παλινδρόμηση

- Χρησιμοποιεί τη λογιστική συνάρτηση (logistic function) για να καθορίσει πόσο πιθανό είναι ένα σημείο δεδομένων να ανήκει σε μια κλάση ή σε μια άλλη κλάση
- Η λογιστική συνάρτηση έχει την μορφή:

$$P(y = 1 \mid \mathbf{x}) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n)}}$$

- Οι συντελεστές β_i υπολογίζονται μέσω παλινδρόμησης ώστε να βελτιστοποιείται η πιθανότητα σωστής ταξινόμησης
- Είναι κατάλληλη για δυαδική κατηγοριοποίηση (0/1), π.χ. ΝΑΙ/ΟΧΙ
- Παρέχει και πιθανότητα πρόβλεψης, όχι μόνο την τελική κατηγορία
- Στο scikit-learn: sklearn.linear_model.LogisticRegression

Κ-πλησιέστεροι γείτονες

- Ο αλγόριθμος k-nearest neighbors (KNN) είναι ένα αλγόριθμος που αναθέτει μια κατηγορία σε ένα σημείο δεδομένων "κοιτάζοντας" παρόμοια σημεία δεδομένων με γνωστές κατηγορίες
- Είναι απλός αλγόριθμος και έχει καλή επίδοση σε μικρά δεδομένα
- Είναι όμως αργός σε μεγάλα σύνολα δεδομένων και είναι ευαίσθητος στην κλίμακα των χαρακτηριστικών
- Στο scikit-learn: sklearn.neighbors.KNeighborsClassifier

Support Vector Machines

- Διαχωρίζει τις κατηγορίες μέσω επιπέδων (planes) διασφαλίζοντας ότι τα επίπεδα αυτά έχουν την μέγιστη δυνατή απόσταση (margin) από τα σημεία που διαχωρίζουν
- Τα επίπεδα-διαχωριστές μπορούν να λαμβάνουν διάφορα σχήματα χρησιμοποιώντας κατάλληλα kernel functions (π.χ. καμπύλες διαφόρων σχημάτων)
- Είναι αποτελεσματικό σε υψηλές διαστάσεις
- Έχουν υψηλό υπολογιστικό κόστος σε μεγάλα δεδομένα
- Στο scikit-learn: sklearn.svm.SVC

Συσταδοποίηση

KMeans, ιεραρχική συσταδοποίηση

Γενικά περί συσταδοποίησης

- Η συσταδοποίηση είναι ένας τύπος μη-επιβλεπόμενης μάθησης
- Σε κάθε σημείο δεδομένων (δείγμα) πρέπει να ανατεθεί μια ετικέτα και πρέπει ο αλγόριθμος που θα χρησιμοποιηθεί να συμπεράνει από τα δεδομένα ποια σημεία δεδομένων ανήκουν στην ίδια συστάδα
- Για να συμβεί αυτό υπολογίζεται κάποιας μορφής απόσταση μεταξύ των σημείων δεδομένων και τα σημεία που ανήκουν στην ίδια συστάδα θα πρέπει κατά κάποια έννοια να είναι κοντά

KMeans

- Ο αλγόριθμος KMeans είναι από τους απλούστερους αλγορίθμους συσταδοποίησης
- Στον αλγόριθμο αυτό ισχύει ότι:
  - Το κέντρο κάθε συστάδας είναι ο μέσος όρος από όλα τα σημεία που ανήκουν στην συστάδα
  - Κάθε σημείο μιας συστάδας είναι πλησιέστερα στο κέντρο της συστάδας παρά στο κέντρο οποιασδήποτε άλλης συστάδας

- Ο αλγόριθμος KMeans δέχεται αρχικά τον επιθυμητό αριθμό συστάδων k
- Στη συνέχεια k σημεία από το σύνολο δεδομένων επιλέγονται ως κέντρα, με τυχαίο τρόπο
- Μετά επαναλαμβάνονται τα ακόλουθα:
  - κάθε σημείο ορίζεται να ανήκει στη συστάδα που το κέντρο της είναι πλησιέστερα στο σημείο
  - για κάθε συστάδα επιλέγεται ένα νέο κέντρο που είναι ο μέσος όρος των σημείων της συστάδας
- Η διαδικασία επαναλαμβάνεται μέχρι να μην αλλάζουν πλέον οι συστάδες
- Η παραπάνω διαδικασία αποδεδειγμένα συγκλίνει

Παράδειγμα εφαρμογής του KMeans σε συνθετικά δεδομένα

- Στο παράδειγμα αυτό θα χρησιμοποιήσουμε ξανά τη make_blobs, αυτή τη φορά με τις ακόλουθες παραμέτρους: centers=4, n_samples=200, random_state=42, cluster_std=1.7
- Θα απεικονίσουμε γραφικά τα δείγματα με το matplotlib, όλα με το ίδιο χρώμα
- Θα εφαρμόσουμε τον αλγόριθμο KMeans χρησιμοποιώντας ως τιμή της υπερπαραμέτρου (hyperparameter) k το 4
- Θα απεικονίσουμε γραφικά εκ νέου τα δείγματα με διαφορετικό χρώμα για κάθε συστάδα
- Θα υπολογίσουμε την μετρική accuracy
  - Έχει νόημα σε ένα πρόβλημα συσταδοποίησης να υπολογιστεί η μετρική accuracy;
- Θα υπολογίσουμε την μετρική silhouette_score - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html

https://mybinder.org/v2/gh/chgogos/ekdda/HEAD?filepath=dawp/w9/w9_KMeans.ipynb

Άσκηση 6: KMeans

1. Δημιουργήστε συνθετικά δεδομένα με την make_blobs επιλέγοντας τυχαία μια ακέραια τιμή για centers από 2 μέχρι και 5 (οι υπόλοιπες παράμετροι να είναι: n_samples=100, random_state=42, cluster_std=1.3)
2. Εφαρμόστε τον Kmeans για κάθε τιμή της υπερπαραμέτρου k από 2 μέχρι και 5 και υπολογίστε και εμφανίστε για κάθε εφαρμογή την μετρική silhouette_score
3. Εξετάστε αν η υψηλότερη τιμή της silhouette_score έδωσε το σωστό αριθμό συστάδων όπως αυτός επιλέχθηκε με την τυχαία ακέραια τιμή στο πρώτο βήμα
4. Απεικονίστε γραφικά τις συστάδες όπως τις δημιούργησε η make_blobs και όπως τις αναγνώρισε η "καλύτερη" εφαρμογή της Kmeans σύμφωνα με το silhouette_score

Ιεραρχική συσταδοποίηση

- Η ιεραρχική συσταδοποίηση ξεκινά τοποθετώντας αρχικά κάθε σημείο δεδομένων στην δική του συστάδα και σταδιακά συνενώνει συστάδες (με βάση κάποιον κανόνα), μέχρι να υπάρξει ο επιθυμητός αριθμός συστάδων
- Θα πρέπει να υπάρχει, όπως και στον KMEANS κάποιο μέτρο απόστασης d
- Με βάση το μέτρο απόστασης d ορίζεται ένα επιπλέον μέτρο απόστασης μεταξύ των συστάδων U και V με κάποια από τις ακόλουθες μεθόδους: single, complete, average, και ward
- Σε κάθε επανάληψη δύο συστάδες που είναι η μια πλησιέστερα στην άλλη συνενώνονται και οι αποστάσεις μεταξύ των συστάδων επαναυπολογίζονται

$$single:\ d(U,V) = \min_{u \in U, v \in V} d(u,v)$$

$$complete:\ d(U,V) = \max_{u \in U, v \in V} d(u,v)$$

$$average:\ d(U,V) = \sum_{u \in U, v \in V} \frac{d(u,v)}{|U|\,|V|}$$

$$ward:\ minimize\ variance\ in\ each\ cluster$$

Παράδειγμα εφαρμογής του Hierarchical Clustering σε συνθετικά δεδομένα

- Θα εφαρμόσουμε τον αλγόριθμο ιεραρχικής συσταδοποίησης που διαθέτει το scikit-learn για να λύσουμε ξανά το παράδειγμα που λύσαμε με τον KMeans
- Παρατηρήστε ότι ο αλγόριθμος ιεραρχικής συσταδοποίησης στο scikit-learn είναι ο AgglomerativeClustering από το sklearn.cluster
- Agglomerative σημαίνει “χτίζω προς τα πάνω συνδυάζοντας μικρά τμήματα”

Βιβλιογραφία

- https://jakevdp.github.io/PythonDataScienceHandbook/
- Data Analysis with Python 2024-2025 - https://courses.mooc.fi/org/uh-cs/courses/data-analysis-with-python-2024-2025
- https://marko-knoebl.github.io/slides/python-and-data-science-all-en.html
- https://www.fun-mooc.fr/en/courses/machine-learning-python-scikit-learn/