ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

# Jupyter Notebook και JupyterLab στην Επιστήμη Δεδομένων: Εξέλιξη, Τεχνική Αρχιτεκτονική και Σύγκριση

Σφέτσιος Χαράλαμπος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Δαδαλιάρης Αντώνιος
Επίκουρος Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ
(εφόσον υπάρχει)

-

Λαμία 15/10 έτος 2025

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

# Jupyter Notebook και JupyterLab στην Επιστήμη Δεδομένων: Εξέλιξη, Τεχνική Αρχιτεκτονική και Σύγκριση

Σφέτσιος Χαράλαμπος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Δαδαλιάρης Αντώνιος
Επίκουρος Καθηγητής

ΣΥΝΕΠΙΒΛΕΠΩΝ
(εφόσον υπάρχει)

-

Λαμία 15/10 έτος 2025


UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

# Jupyter Notebook and JupyterLab in Data Science: Evolution, Technical Architecture and Comparison

Sfetsios Charalampos

FINAL THESIS

ADVISOR

Dadaliaris Antonios
Assistant Professor

CO ADVISOR

-

Lamia 15/10 year 2025

*«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:*

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

*3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

*4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: 22/10/2025

Ο – Η Δηλ.

*(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση*
*του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων*
*σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»*

## ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία εξετάζει τα εργαλεία Jupyter Notebook και Jupyter Lab, τα οποία έχουν καθιερωθεί ως βασικές πλατφόρμες στην επιστήμη δεδομένων. Αναλύεται η ιστορική τους εξέλιξη, η τεχνική αρχιτεκτονική και η σύνδεσή τους με την κοινότητα ανοιχτού κώδικα, ενώ παρουσιάζονται οι ομοιότητες και οι διαφορές τους. Επιπλέον, εστιάζουμε στις εφαρμογές τους στην προεπεξεργασία δεδομένων, στην ανάλυση με βιβλιοθήκες όπως οι NumPy και Pandas, καθώς και στην οπτικοποίηση με εργαλεία όπως οι Matplotlib και Seaborn. Ιδιαίτερη αναφορά γίνεται στη χρήση διαδραστικών στοιχείων (widgets) και στην ανάπτυξη dashboards, που μετατρέπουν τα notebooks σε δυναμικές εφαρμογές. Μέσα από τη συγκριτική αξιολόγηση καταδεικνύεται ότι το Jupyter Lab, αν και πιο σύνθετο, προσφέρει επεκτασιμότητα και καλύτερη οργάνωση, ενώ το Notebook παραμένει ελαφρύ και ιδανικό για διδακτική χρήση και γρήγορα πειράματα. Συνολικά, η εργασία αναδεικνύει τον κεντρικό ρόλο του οικοσυστήματος Jupyter στην ανοιχτή επιστήμη, την αναπαραγωγιμότητα και την καθημερινή πρακτική των data scientists.

## ABSTRACT

This thesis explores the **Jupyter Notebook** and **Jupyter Lab** environments, which have become key platforms in modern data science. It investigates their historical evolution, technical architecture, and close connection with the open-source community, while highlighting their similarities and differences. Furthermore, it discusses their main applications in data preprocessing, analysis with libraries such as **NumPy** and **Pandas**, and visualization through **Matplotlib** and **Seaborn**. Special emphasis is placed on interactive elements (widgets) and dashboard development, which transform notebooks into dynamic applications. The comparative study shows that while **Jupyter Lab** provides extensibility and better project organization, the classic **Notebook** remains lightweight and particularly effective for teaching and exploratory tasks. Overall, the study underlines the central role of the Jupyter ecosystem in open science, reproducibility, and the everyday workflows of data scientists.

Περιεχόμενο

# ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

## 1.1 Παρουσίαση του θέματος και της σημασίας του

Τα **Jupyter Notebook** και **Jupyter Lab** αποτελούν σύγχρονα εργαλεία υπολογιστικών σημειωματαρίων, τα οποία έχουν αναδειχθεί σε θεμελιώδεις πλατφόρμες για την επιστημονική έρευνα και την ανάλυση δεδομένων. Ένα υπολογιστικό σημειωματάριο (computational notebook) είναι ένα περιβάλλον που επιτρέπει την ανάμιξη εκτελέσιμου κώδικα, επεξηγηματικού κειμένου, μαθηματικών εξισώσεων και γραφημάτων σε ένα ενιαίο *έγγραφο* – διευκολύνοντας έτσι την έννοια του «αφηγηματικού» προγραμματισμού (narrative computing) [1]. Η ευκολία με την οποία οι επιστήμονες μπορούν να αναπτύσσουν κώδικα, να βλέπουν άμεσα αποτελέσματα και να τεκμηριώνουν τη διαδικασία σε πραγματικό χρόνο καθιστά αυτά τα εργαλεία εξαιρετικά σημαντικά. Πράγματι, υπολογιστικά σημειωματάρια όπως το **Jupyter** χρησιμοποιούνται πλέον από **εκατομμύρια** επιστήμονες δεδομένων, μηχανικούς μηχανικής μάθησης και ερευνητές για διερευνητικό προγραμματισμό και ανάλυση [2]. Η δημοτικότητά τους έχει εκτιναχθεί την τελευταία δεκαετία, καθιστώντας το **Jupyter Notebook** σχεδόν συνώνυμο με τον όρο «επιστημονικό σημειωματάριο» στην κοινότητα της επιστήμης δεδομένων [3].

Η σημασία των εργαλείων αυτών έγκειται τόσο στη **διαδραστικότητα** που προσφέρουν όσο και στη δυνατότητα **τεκμηρίωσης και αναπαραγωγιμότητας** των αποτελεσμάτων. Σε αντίθεση με τα παραδοσιακά σενάρια εκτέλεσης κώδικα, όπου ο κώδικας και τα αποτελέσματά του είναι διακριτά, τα σημειωματάρια επιτρέπουν την άμεση παρουσίαση των αποτελεσμάτων (πίνακες, γραφήματα, κ.λπ.) δίπλα στον κώδικα και τις επεξηγήσεις. Αυτός ο **συνδυασμός κώδικα και αφήγησης** υποστηρίζει αποτελεσματικά την ιδέα της «επαναλήψιμης έρευνας», όπου άλλοι ερευνητές (ή και ο ίδιος ο επιστήμονας σε μεταγενέστερο χρόνο) μπορούν να κατανοήσουν ακριβώς τη μεθοδολογία και να επαναλάβουν τα πειράματα [1]. Παράλληλα, η ανοιχτή φύση του οικοσυστήματος Jupyter και η υποστήριξη πολλαπλών γλωσσών προγραμματισμού (Python, R, Julia κ.ά.) διεύρυναν δραστικά την απήχησή του στην επιστημονική και προγραμματιστική κοινότητα [4].

## 1.2 Ο ρόλος των Jupyter Notebook και Jupyter Lab στην επιστήμη δεδομένων

Στο πεδίο της **επιστήμης δεδομένων**, η διαδραστική ανάλυση και ο πειραματισμός με δεδομένα είναι κεντρικής σημασίας. Τα **Jupyter Notebook** έχουν καθιερωθεί ως το κατεξοχήν εργαλείο για αυτό το σκοπό, καθώς επιτρέπουν στους επιστήμονες δεδομένων να πραγματοποιούν **διερευνητική ανάλυση δεδομένων**, να υλοποιούν γρήγορα μοντέλα μηχανικής μάθησης και να οπτικοποιούν αποτελέσματα μέσα σε ένα ενιαίο, κατανοητό πλαίσιο [2]. Η δυνατότητα άμεσης εκτέλεσης κώδικα σε κελλοειδείς δομές (cells) και άμεσης προβολής των αποτελεσμάτων δίπλα στον κώδικα επιταχύνει τον **κύκλο πειραματισμού** (write-evaluate-refine) που χαρακτηρίζει την εργασία των επιστημόνων δεδομένων. Τα notebooks υποστηρίζουν εξάλλου πρακτικές **επαναλήψιμου υπολογισμού** (reproducible computing), αφού καταγράφουν κάθε βήμα της ανάλυσης μαζί με τα αποτελέσματά του [3]. Αυτό τα καθιστά ιδιαίτερα χρήσιμα για τη διαμοίραση επιστημονικών αποτελεσμάτων, τη συνεργασία μεταξύ ομάδων και τη διδασκαλία.

Το **Jupyter Lab**, η νεότερη γενιά του περιβάλλοντος Jupyter, επεκτείνει ακόμα περισσότερο αυτές τις δυνατότητες, παρέχοντας ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE-like) εντός του φυλλομετρητή. Με το **Jupyter Lab**, ο χρήστης μπορεί να ανοίγει πολλαπλά σημειωματάρια, αρχεία κώδικα, τερματικά και παράθυρα διαγράφηματων σε καρτέλες, αξιοποιώντας μια **ευέλικτη διεπαφή εργασίας** που θυμίζει σύγχρονα IDE [4]. Ο ρόλος του Jupyter Lab στην επιστήμη δεδομένων είναι να υποστηρίξει πιο σύνθετες ροές εργασίας: για παράδειγμα, ένας επιστήμονας μπορεί να επεξεργάζεται ένα dataset σε ένα φύλλο CSV, να γράφει κώδικα ανάλυσης σε ένα notebook, και ταυτόχρονα να παρακολουθεί τα γραφήματα σε πραγματικό χρόνο – όλα στο ίδιο παράθυρο. Επιπλέον, το **Jupyter Lab** βελτιώνει τη συνεργασία και την επεκτασιμότητα, καθώς επιτρέπει την εγκατάσταση προσθέτων (extensions) που μπορούν να προσθέσουν λειτουργίες (όπως διαχειριστές Git, εργαλεία visual debugging, κ.ά.) στο περιβάλλον. Με τον τρόπο αυτό, το **Jupyter Lab** λειτουργεί ως μια **ενοποιημένη πλατφόρμα** που καλύπτει ολόκληρο τον κύκλο ζωής της επιστήμης δεδομένων: από την αρχική διερεύνηση μέχρι την παραγωγή αναπαράξιμων αναφορών και εφαρμογών.

## 1.3 **Στόχοι της εργασίας**

Σκοπός της παρούσας πτυχιακής εργασίας είναι να μελετήσει σε βάθος τα εργαλεία **Jupyter Notebook** και **Jupyter Lab**, αναδεικνύοντας την εξέλιξή τους, την τεχνική τους αρχιτεκτονική, καθώς και τις ομοιότητες και διαφορές τους ως προς τη λειτουργικότητα και τις δυνατότητες που προσφέρουν. Συγκεκριμένα, στους επόμενους κεφάλαιους θα παρουσιαστεί: (α) το ιστορικό υπόβαθρο των εργαλείων, από την απαρχή του **IPython Notebook** έως τη μετάβαση στο έργο **Jupyter** και την ανάπτυξη του **Jupyter Lab**, (β) η δομή και η αρχιτεκτονική του συστήματος Jupyter – πώς το υποσύστημα του server, των πυρήνων (kernels) και των αρχείων notebook συνεργάζονται – καθώς και η σχέση του έργου με την κοινότητα ανοιχτού κώδικα, και (γ) μια λεπτομερής παρουσίαση των τεχνικών χαρακτηριστικών του **Jupyter Notebook** και του **Jupyter Lab**, με συγκριτική αξιολόγηση σε τομείς όπως η εμπειρία χρήστη, η απόδοση, η **επεκτασιμότητα** και η διαδραστικότητα. Μέσα από αυτή τη συγκριτική μελέτη, στόχος είναι να αναδειχθεί πώς το **Jupyter Lab** επιχειρεί να υπερβεί περιορισμούς του κλασικού Notebook, καθώς και να δοθούν κατευθύνσεις για την καταλληλότερη χρήση κάθε εργαλείου σε διαφορετικά σενάρια στην επιστήμη δεδομένων.

Τέλος, η εργασία θα επικεντρωθεί στην **εφαρμογή των εργαλείων στην επιστήμη δεδομένων**, παρουσιάζοντας παραδείγματα και περιπτώσεις χρήσης όπου το **Jupyter Notebook** και το **Jupyter Lab** έχουν συμβάλει στην ανάλυση δεδομένων, στη μηχανική μάθηση και στην αναπαραγωγή επιστημονικών αποτελεσμάτων. Θα δοθεί έμφαση στην τεχνική ακρίβεια των περιγραφών, υποστηρίζοντας κάθε βασική έννοια με βιβλιογραφικές αναφορές από την επιστημονική βιβλιογραφία (IEEE, ACM κλπ.), ώστε ο αναγνώστης να αποκτήσει μια πλήρη και αξιόπιστη εικόνα των δυνατοτήτων και των προκλήσεων που συνοδεύουν τη χρήση των **Jupyter Notebook** και **Jupyter Lab**.

# ΚΕΦΑΛΑΙΟ 2 Ιστορικό και Θεωρητικό Υπόβαθρο

## 2.1 Εξέλιξη των εργαλείων: Από το **IPython Notebook** στο **Jupyter Notebook** και τελικά στο **Jupyter Lab**

Η ιστορία του **Jupyter Notebook** ξεκινά από το πρότερο έργο **IPython**, ένα εγχείρημα που ξεκίνησε στις αρχές της δεκαετίας του 2000 με στόχο τη βελτίωση του διαδραστικού υπολογισμού στην Python. Το **IPython** (Interactive Python) παρείχε αρχικά ένα επεκτάσιμο κέλυφος εντολών (interactive shell) και σταδιακά εξελίχθηκε ώστε να συμπεριλάβει ένα web-based σημειωματάριο (το λεγόμενο *IPython Notebook*) που επέτρεπε στους χρήστες να γράφουν και να εκτελούν κώδικα Python μέσω ενός φυλλομετρητή. Το **IPython Notebook** παρουσιάστηκε δημόσια γύρω στο 2011-2012, προσφέροντας για πρώτη φορά τη δυνατότητα να συνδυάζεται κώδικας, κείμενο Markdown, μαθηματικοί τύποι και οπτικοποιήσεις σε μια *σειρά κελιών* μέσα από περιβάλλον web.

Μέχρι το 2014, η δημοτικότητα του IPython Notebook είχε αυξηθεί σημαντικά, οδηγώντας την ομάδα ανάπτυξης (με επικεφαλής τον Fernando Pérez και τον Brian Granger) να προχωρήσει σε μια σημαντική αναδιάρθρωση του έργου. Συγκεκριμένα, αναγνωρίζοντας ότι το μοντέλο των σημειωματαρίων είναι χρήσιμο όχι μόνο για την Python αλλά και για άλλες γλώσσες, αποφασίστηκε ο "μεγάλος διαχωρισμός" (Big Split): οι γλώσσα-αγνωστικές λειτουργίες του IPython (όπως η μορφή αρχείου notebook, το πρωτόκολλο επικοινωνίας με τους πυρήνες, η web εφαρμογή του notebook κ.λπ.) αποσπάστηκαν σε ένα νέο αυτόνομο έργο με την ονομασία **Project Jupyter [5]**. Έτσι, από την έκδοση IPython 4.0 (τέλος 2015) και έπειτα, το ίδιο το IPython συνεχίζει ως μια βιβλιοθήκη για την Python (παρέχοντας π.χ. τον Python **kernel** στο οικοσύστημα Jupyter), ενώ το **Jupyter Notebook** καθιερώθηκε ως η *διάδοχη πλατφόρμα* του IPython Notebook, ικανή να υποστηρίζει πολλαπλές γλώσσες προγραμματισμού εξίσου (Python, R, Julia, κ.ά.) [6]. Αυτή η εξέλιξη σηματοδότησε τη μετάβαση σε ένα πιο ανοιχτό και καθολικό οικοσύστημα: το όνομα "Jupyter" προκύπτει συμβολικά από τα αρχικά των Julia, Python και R – τριών εκ των βασικών γλωσσών που στοχεύει να υποστηρίζει.

Κατά την ίδια περίοδο, η δημοτικότητα των notebooks εκτοξεύθηκε περαιτέρω, με παράγοντες όπως η ενσωμάτωση της δυνατότητας προεπισκόπησης notebooks στο GitHub (2015) να παίζουν σημαντικό ρόλο [4]. Η δυνατότητα των χρηστών να μοιράζονται αρχεία `.ipynb` στο GitHub και να τα βλέπουν άμεσα σε μορφή HTML (μέσω του nbviewer) έκανε τα notebooks ένα ευρέως χρησιμοποιούμενο μέσο ανταλλαγής κώδικα και αποτελεσμάτων στην κοινότητα. Μέχρι τα τέλη της δεκαετίας του 2010, είχε ήδη δημιουργηθεί ένας πλούσιος **οικοσύστημα εργαλείων γύρω από το Jupyter**: υπηρεσίες όπως το **Binder** για κοινή χρήση εκτελέσιμων notebooks online, πλατφόρμες cloud (π.χ. Google Colab, Azure Notebooks), και πληθώρα επεκτάσεων για το κλασικό Notebook.

Παράλληλα με αυτή την ανάπτυξη, έγινε αντιληπτό ότι η διεπαφή του κλασικού Jupyter Notebook, αν και απλή και αποτελεσματική, είχε ορισμένους περιορισμούς. Ήταν βασισμένη σε ένα μονο-σελιδοποιημένο μοντέλο: κάθε notebook άνοιγε στη δική του σελίδα, με περιορισμένες δυνατότητες διαχείρισης πολλαπλών αρχείων ή σύνθετων έργων. Για να αντιμετωπιστούν αυτοί οι περιορισμοί, η κοινότητα του Project Jupyter ξεκίνησε την ανάπτυξη μιας νέας διεπαφής, που θα εξελισσόταν στο **Jupyter Lab**. Το Jupyter Lab άρχισε να αναπτύσσεται γύρω στο 2016 και κυκλοφόρησε σε σταθερή έκδοση περίπου το 2018-2019. Σε αντίθεση με το κλασικό Notebook, το **Jupyter Lab** σχεδιάστηκε εξαρχής ως ένα **πολυ-παραθυρικό** περιβάλλον, όπου ο χρήστης μπορεί να έχει ανοιχτά πολλαπλά notebooks και

άλλα αρχεία ταυτόχρονα, οργανωμένα σε καρτέλες και πλαίσια εργασίας. Η πρώτη επίσημη έκδοση (JupyterLab 1.0) κυκλοφόρησε τον Ιούνιο του 2019, σηματοδοτώντας την ωρίμανση του εργαλείου και την προοπτική να αντικαταστήσει πλήρως το κλασικό Notebook στις περισσότερες χρήσεις.

Το **Jupyter Lab** μπορεί να θεωρηθεί ως η φυσική εξέλιξη του Jupyter Notebook, διατηρώντας όλα τα βασικά του στοιχεία αλλά παρέχοντας μια πιο **ισχυρή και επεκτάσιμη πλατφόρμα**. Μάλιστα, η έκδοση **JupyterLab 4** (κυκλοφόρησε το 2023) χαρακτηρίζεται ως "νέας γενιάς διεπαφή σημειωματαρίων" βασισμένη στην κλασική λειτουργικότητα του Notebook [8]. Η μετάβαση από το κλασικό Notebook στο JupyterLab έγινε σταδιακά· και τα δύο εργαλεία συνυπάρχουν μέχρι σήμερα, με το κλασικό **Jupyter Notebook** (εκδ. 6.x και 7.x) να συνεχίζει να χρησιμοποιείται ευρέως, ενώ το **Jupyter Lab** κερδίζει ολοένα και περισσότερη αποδοχή ως το προεπιλεγμένο περιβάλλον για πιο σύνθετες εργασίες. Στα επόμενα τμήματα, θα δούμε λεπτομερέστερα πώς δομείται το σύστημα **Jupyter** και ποιες είναι οι διαφορές μεταξύ Notebook και Lab.

## 2.2 Δομή και αρχιτεκτονική του **Jupyter**

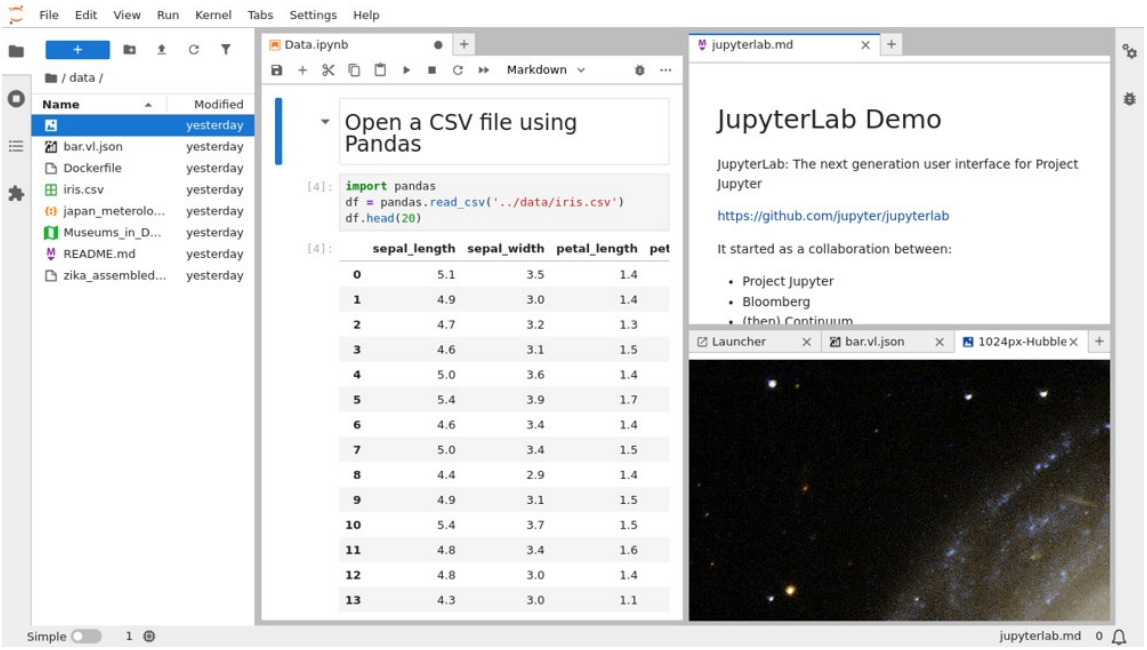
Η αρχιτεκτονική του Jupyter στηρίζεται σε έναν πελάτη-διακομιστή (client-server) σχεδιασμό, ο οποίος επιτρέπει την αλληλεπίδραση του χρήστη με κώδικα μέσω ενός φυλλομετρητή ιστού (web browser). Ειδικότερα, ένα **Jupyter Notebook** σύστημα αποτελείται από τέσσερα κύρια συστατικά:

- **Web browser (φυλλομετρητής)**: Ο φυλλομετρητής λειτουργεί ως το περιβάλλον χρήστη. Ο χρήστης ανοίγει τον φυλλομετρητή του (π.χ. Chrome, Firefox) και φορτώνει τη σελίδα του Jupyter Notebook ή Lab. Όλη η αλληλεπίδραση (πληκτρολόγηση κώδικα, εντολών, προβολή αποτελεσμάτων) γίνεται μέσω αυτής της διεπαφής.

- **Notebook server (διακομιστής σημειωματαρίου)**: Είναι μια εφαρμογή (υλοποιημένη σε Python, βασισμένη στο πλαίσιο Tornado) που τρέχει τοπικά ή σε απομακρυσμένο μηχάνημα και διαχειρίζεται τα αιτήματα του χρήστη. Ο server αυτός αναλαμβάνει να σερβίρει τη διεπαφή (HTML/JavaScript) στον browser, να διαχειρίζεται τα αρχεία notebook, και – πολύ σημαντικό – να διαμεσολαβεί στην επικοινωνία μεταξύ του browser και των πυρήνων (kernels).

- **Notebook file (αρχείο σημειωματαρίου)**: Κάθε notebook αντιστοιχεί σε ένα αρχείο με κατάληξη `.ipynb`. Το αρχείο αυτό αποθηκεύεται σε μορφή **JSON** και περιέχει τη διάρθρωση των κελιών, τον κώδικα, το κείμενο Markdown, καθώς και τα παραγόμενα αποτελέσματα (output) σε μορφή που μπορεί να σειριακοποιηθεί (συνήθως επίσης JSON για γραφήματα, κείμενο για εξόδους κονσόλας κ.λπ.). Ο notebook server διαβάζει και γράφει αυτά τα αρχεία από/προς το σύστημα αρχείων, επιτρέποντας στον χρήστη να αποθηκεύει το έργο του.

- **Kernel (πυρήνας εκτέλεσης)**: Ο *kernel* είναι η μηχανή εκτέλεσης που τρέχει τον κώδικα ενός notebook. Για παράδειγμα, για κώδικα Python χρησιμοποιείται ο **IPython kernel** (μέρος του πακέτου IPython), ενώ για R χρησιμοποιείται ο **IRkernel** κ.ο.κ. Κάθε ανοιχτό notebook συνδέεται με έναν kernel συγκεκριμένης γλώσσας. Ο notebook server εκκινεί και διαχειρίζεται αυτούς τους kernels ως ανεξάρτητες διεργασίες. Η επικοινωνία μεταξύ του notebook server και του kernel γίνεται μέσω ενός καθορισμένου πρωτοκόλλου μηνυμάτων (Jupyter message protocol) πάνω από **WebSockets**, επιτρέποντας την αποστολή εντολών κώδικα από τον χρήστη προς τον kernel και την επιστροφή των αποτελεσμάτων (ή σφαλμάτων) πίσω στον χρήστη.




**Εικόνα 2.1 1 : Η αρχιτεκτονική του συστήματος Jupyter – Ο χρήστης αλληλεπιδρά μέσω φυλλομετρητή (αριστερά), ο οποίος επικοινωνεί με τον διακομιστή Jupyter (στο κέντρο)· ο διακομιστής φορτώνει/αποθηκεύει αρχεία notebook στο σύστημα αρχείων και προωθεί τον κώδικα προς τον κατάλληλο *kernel* (δεξιά) για εκτέλεση, επιστρέφοντας τα αποτελέσματα πίσω στον φυλλομετρητή για εμφάνιση. (Πηγή: Τεκμηρίωση JupyterLab)**

Στην πράξη, όταν ο χρήστης εκτελεί ένα κελί κώδικα στο notebook, η ακολουθία είναι η εξής: το αίτημα φεύγει από τον browser προς τον server, ο server το προωθεί μέσω **WebSocket** στον kernel, ο kernel εκτελεί τον κώδικα και επιστρέφει τα αποτελέσματα, και ο server τα στέλνει πίσω στον browser για να εμφανιστούν άμεσα κάτω από το κελί. Αυτή η αρχιτεκτονική επιτρέπει μεγάλη ευελιξία: ο browser μπορεί να βρίσκεται ακόμη και σε άλλον υπολογιστή από τον kernel (π.χ. χρήση Jupyter σε απομακρυσμένο server στο cloud), καθότι η επικοινωνία γίνεται μέσω δικτύου. Επίσης, επειδή τα αρχεία notebook είναι **JSON** και ανοιχτού προτύπου, μπορούν να μετατραπούν εύκολα σε άλλες μορφές ή να **συνδεθούν με άλλα εργαλεία**. Για παράδειγμα, το εργαλείο `nbconvert` μπορεί να μετατρέπει notebooks σε HTML, PDF, διαφάνειες κ.λπ. για διανομή [7], ενώ υπηρεσίες όπως το `nbviewer` παρέχουν δημόσια φιλοξενία και προβολή notebooks χωρίς να απαιτείται εγκατάσταση Jupyter από τον αναγνώστη.

Στο οικοσύστημα Jupyter, κάθε υποστηριζόμενη γλώσσα προγραμματισμού υλοποιείται μέσω ενός αντίστοιχου kernel. Υπάρχουν δεκάδες kernels διαθέσιμοι (Python, R, Julia, JavaScript, C++, Octave, κ.ά.), καθιστώντας το Jupyter πραγματικά πολυγλωσσικό περιβάλλον. Αυτός ο σχεδιασμός – όπου το frontend (διεπαφή notebook) είναι αποσυνδεδεμένο από τον backend (kernel) – είναι που επιτρέπει την επεκτασιμότητα σε πολλαπλές γλώσσες [4]. Το **IPython kernel** παραμένει ο προεπιλεγμένος πυρήνας για την Python (αποτελώντας τη συνέχεια του αρχικού IPython για Jupyter) [7], όμως το σύστημα είναι ανοιχτό να ενσωματώνει οποιονδήποτε άλλον kernel ακολουθεί το πρωτόκολλο της Jupyter.

Τέλος, αξίζει να σημειωθεί ότι το Jupyter παρέχει ένα σύνολο *υπηρεσιών και υποσυστημάτων* που πλαισιώνουν την κύρια λειτουργία του notebook. Για παράδειγμα, υπάρχει το **JupyterHub** – μια υπηρεσία που επιτρέπει τη φιλοξενία πολλαπλών χρηστών notebooks σε έναν server (συχνά χρησιμοποιείται σε πανεπιστήμια ή εταιρίες για παροχή περιβάλλοντος Jupyter σε πολλούς χρήστες). Επίσης, το νεότερο **JupyterLite** αποτελεί μια ελαφριά διανομή Jupyter που τρέχει εξ' ολοκλήρου στον browser (χρησιμοποιώντας WebAssembly) χωρίς να απαιτείται server, δίνοντας μια γεύση του μέλλοντος όπου τα notebooks μπορούν να λειτουργούν παντού με ελάχιστες εξαρτήσεις [8]. Όλα αυτά τα εργαλεία βασίζονται στην ίδια κεντρική αρχιτεκτονική που περιγράφηκε: *ανοιχτά πρότυπα*, *επεκτασιμότητα* και έμφαση στη διάδραση ανθρώπου-υπολογιστή μέσω μιας ενιαίας, ευέλικτης διεπαφής.

## 2.3 Σύνδεση με τον ανοιχτό κώδικα και την επιστημονική κοινότητα

Ένας από τους καθοριστικούς παράγοντες της επιτυχίας του Jupyter υπήρξε εξαρχής η φιλοσοφία του ανοιχτού κώδικα και η στέρεη σύνδεσή του με την επιστημονική κοινότητα. Το Project Jupyter είναι ένα εγχείρημα 100% ανοιχτού κώδικα, με άδεια BSD (για το IPython) και παρόμοιες άδειες για τα υπόλοιπα συστατικά, γεγονός που επέτρεψε σε χιλιάδες προγραμματιστές και ερευνητές να συνεισφέρουν σε αυτό. Αυτή η ανοιχτή φύση έχει οδηγήσει στη δημιουργία ενός **εκτεταμένου οικοσυστήματος κοινοτικών εργαλείων** γύρω από το Jupyter [9]. Για παράδειγμα, εργαλεία για τη δημιουργία βιβλίων από notebooks (όπως το Jupyter Book), για τη δημιουργία διαφανειών παρουσίασης (π.χ. RISE), και για την κατασκευή web εφαρμογών από notebooks (π.χ. Voilà) έχουν αναπτυχθεί από την κοινότητα, επεκτείνοντας τη χρηστικότητα των notebooks πέρα από τα όρια της βασικής εφαρμογής [9]. Η ίδια η ευελιξία των notebooks – ως ανοιχτής μορφής εγγράφων – ευνόησε τέτοιες καινοτομίες, καθώς οποιοσδήποτε μπορούσε να αξιοποιήσει το format ή το Jupyter kernel/protocol για νέες χρήσεις.

Η επιστημονική κοινότητα έχει αγκαλιάσει το Jupyter ως εργαλείο **“open science”**. Πολλές δημοσιεύσεις πλέον συνοδεύονται από Jupyter notebooks που περιέχουν τον κώδικα ανάλυσης και τα δεδομένα (ή συνδέσμους σε αυτά), επιτρέποντας στους αναγνώστες να αναπαράγουν και να εξερευνήσουν περαιτέρω τα αποτελέσματα. Μελέτες έχουν δείξει ότι τα notebooks βελτιώνουν την **αναπαραγωγιμότητα** της επιστημονικής εργασίας – κάτι κρίσιμο σε πεδία όπως η βιοπληροφορική, η αστρονομία, ή η κοινωνική επιστήμη δεδομένων [10]. Η δυνατότητα να μοιραστεί ένας ερευνητής ένα πλήρες “computational narrative” (υπολογιστική αφήγηση) – δηλαδή, όχι μόνο τα τελικά αποτελέσματα αλλά και τη διαδικασία, βήμα-βήμα –

είναι επαναστατική. Πλατφόρμες όπως το **NBViewer** και το **GitHub** (με native render υποστήριξη για `.ipynb`) αλλά και το **Zenodo** (για DOI σε σύνολα notebooks) διευκολύνουν τη διάχυση της γνώσης μέσω notebooks.

Επιπλέον, το Jupyter έχει λάβει σημαντική υποστήριξη από φορείς χρηματοδότησης και τεχνολογικές εταιρείες, ακριβώς λόγω της αξίας του στην προώθηση της ανοιχτής επιστήμης. Οργανισμοί όπως το Moore Foundation, το Sloan Foundation και άλλοι παρείχαν επιχορηγήσεις για την ανάπτυξη του Jupyter και του **JupyterHub**, συνειδητοποιώντας ότι αποτελούν υποδομές-κλειδιά για την επιστημονική έρευνα στην ψηφιακή εποχή. Η κοινότητα χρηστών και συνεισφερόντων επεκτείνεται από ακαδημαϊκά ιδρύματα έως τμήματα δεδομένων μεγάλων επιχειρήσεων. Ως αποτέλεσμα, το Jupyter εξελίσσεται διαρκώς με γνώμονα τις ανάγκες των χρηστών του: από την προσθήκη της δυνατότητας **συνεργατικής επεξεργασίας σε πραγματικό χρόνο** (ένα χαρακτηριστικό που αναπτύσσεται για το JupyterLab) έως τη δημιουργία εύχρηστων διανομών (Anaconda, Kaggle Notebooks κ.λπ.) που καθιστούν ευκολότερη την πρόσβαση στα εργαλεία αυτά.

Συνολικά, η στενή **διασύνδεση με την κοινότητα** είναι εμφανής σε κάθε πτυχή του έργου Jupyter. Το γεγονός ότι είναι **ανοιχτού κώδικα (open source)** σημαίνει ότι ερευνητές μπορούν να επιθεωρήσουν τον κώδικα για να κατανοήσουν πώς λειτουργεί (κάτι κρίσιμο για εμπιστοσύνη στα εργαλεία τους), να επεκτείνουν τις δυνατότητες του συστήματος προσθέτοντας δικές τους επεκτάσεις ή συμβάλλοντας απευθείας στο κεντρικό έργο, και να προσαρμόσουν το περιβάλλον στις δικές τους εξειδικευμένες ανάγκες. Η κουλτούρα αυτή οδήγησε σε καινοτομίες όπως η δημιουργία εξειδικευμένων *notebook-friendly* βιβλιοθηκών για οπτικοποίηση (π.χ. Plotly, Altair), για interactive widgets (**ipywidgets**), και πολλών άλλων που ενισχύουν την παραγωγικότητα των επιστημόνων δεδομένων. Δεν είναι υπερβολή να πούμε ότι το Jupyter έχει γίνει ο *de facto* τρόπος με τον οποίο μοιράζεται και επικοινωνεί η γνώση στα data science projects – σε συνέδρια, ιστολόγια τεχνολογίας, εκπαιδευτικά σεμινάρια και μαθήματα, τα notebooks είναι πανταχού παρόντα.

Συμπερασματικά, η εξέλιξη του **Jupyter Notebook** και η μετάβασή του στο **Jupyter Lab** δεν μπορούν να ιδωθούν ανεξάρτητα από την κοινότητα που τα περιβάλλει. Η δυναμική αλληλεπίδραση ανάμεσα στους δημιουργούς/συντηρητές του Jupyter και στους χρήστες/επιστήμονες ανά τον κόσμο οδήγησε σε ένα εργαλείο που ανταποκρίνεται άμεσα στις απαιτήσεις της σύγχρονης επιστήμης δεδομένων. Αυτή η συνεργατική δυναμική είναι εμφανής και στη συνεχιζόμενη ανάπτυξη: νέες ιδέες (όπως η ενσωμάτωση **real-time collaboration** ή η βελτίωση της **αξιοπιστίας και ποιότητας** των notebooks) αναδύονται από την κοινότητα και υιοθετούνται στο έργο [11]. Με αυτό το υπόβαθρο, προχωρούμε στην εις βάθος τεχνική εξέταση των δύο εργαλείων (Notebook και Lab) και στην συγκριτική αξιολόγησή τους.

# ΚΕΦΑΛΑΙΟ 3 Τεχνικά Χαρακτηριστικά και Σύγκριση

## 3.1 Jupyter Notebook – Βασικά χαρακτηριστικά, πλεονεκτήματα και περιορισμοί

Το **Jupyter Notebook** (γνωστό και ως "κλασικό notebook") είναι μια web-εφαρμογή που παρέχει ένα απλό και αποτελεσματικό περιβάλλον για διαδραστικό προγραμματισμό. Στο βασικό του περιβάλλον, το Notebook αποτελείται από μια διαδοχή κελιών, όπου κάθε κελί μπορεί να περιέχει είτε κώδικα (π.χ. Python) είτε επεξηγηματικό κείμενο (γραμμένο σε Markdown), καθώς και άλλα στοιχεία όπως μαθηματικές φόρμουλες (με **LaTeX**) ή ενσωματωμένα γραφήματα. Ο χρήστης μπορεί να εκτελεί κάθε κελί ξεχωριστά και να βλέπει αμέσως κάτω από αυτό την έξοδο ή το γράφημα που παράγεται. Αυτή η ροή εργασίας, γνωστή και ως *literate programming*, επιτρέπει την παράλληλη ανάπτυξη κώδικα και τεκμηρίωσης, καθιστώντας το Notebook ιδανικό για **πειραματισμό** και **παρουσίαση** αποτελεσμάτων.

**Βασικά χαρακτηριστικά του Jupyter Notebook:**

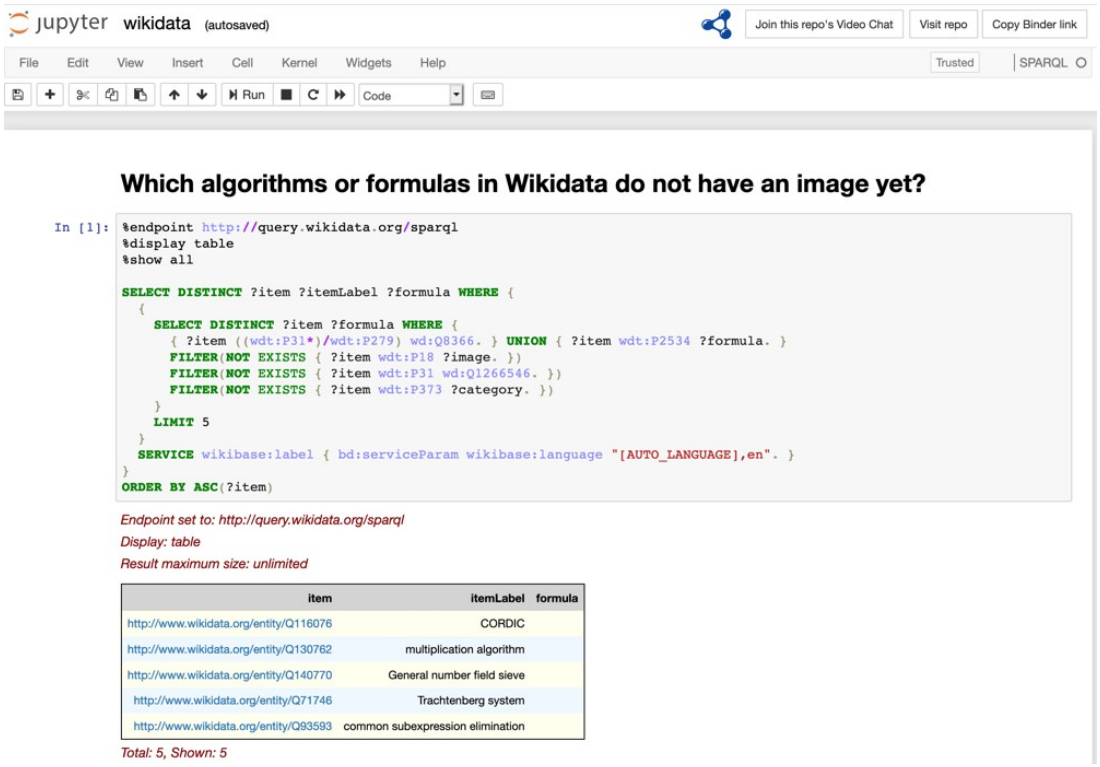
- **Διεπαφή μέσω φυλλομετρητή:** Το Notebook τρέχει σε έναν server (τοπικό ή απομακρυσμένο) και η αλληλεπίδραση γίνεται μέσω browser. Η διεπαφή είναι λιτή: μια γραμμή εργαλείων για βασικές λειτουργίες (αποθήκευση, εκτέλεση, προσθήκη κελιών), ένα μενού, και η κύρια περιοχή που περιέχει τα κελιά.

- **Κύτταρα κώδικα και Markdown:** Τα κελιά κώδικα εκτελούνται κατά παραγγελία και μοιράζονται κατάσταση μέσω του υποκείμενου kernel (π.χ. μεταβλητές που ορίζονται σε ένα κελί είναι διαθέσιμες σε επόμενα). Τα Markdown κελιά επιτρέπουν μορφοποίηση κειμένου, εισαγωγή εικόνων, συνδέσμων, καθώς και LaTeX για μαθηματικά, διευκολύνοντας την **τεκμηρίωση**.

- **Inline γραφήματα και πολυμέσα:** Χάρη σε βιβλιοθήκες όπως η **Matplotlib** ή η **Plotly** το Notebook μπορεί να εμφανίζει γραφήματα, πίνακες και άλλα πλούσια μέσα ακριβώς κάτω από τον κώδικα που τα παράγει. Αυτό σημαίνει ότι η οπτικοποίηση δεδομένων γίνεται άμεσα αντιληπτή μέσα στη ροή ανάλυσης.

- **Ανοιχτή μορφή αρχείου (.ipynb):** Όπως αναφέρθηκε, τα notebooks αποθηκεύονται σε JSON, επιτρέποντας την εύκολη έκδοση (versioning) και μετατροπή τους. Εργαλεία όπως το `nbconvert` μπορούν να παραγάγουν HTML ή PDF εκδοχές, διευκολύνοντας τη δημοσίευση. Επίσης, πλατφόρμες όπως το GitHub μπορούν να εμφανίσουν το περιεχόμενο ενός notebook απευθείας.

- **Υποστήριξη πολλαπλών γλωσσών:** Αν και συνηθέστερα χρησιμοποιείται με Python, το Jupyter Notebook υποστηρίζει πάνω από 40 γλώσσες μέσω των kernels, δίνοντας ευελιξία σε χρήστες διαφορετικών οικοσυστημάτων (π.χ. **R**, **Julia**, **C++**, **Octave**, **Scala** κ.ά.) [4]




**Εικόνα 3.1 1 : Παράδειγμα περιβάλλοντος Jupyter Notebook – Το πάνω μέρος περιλαμβάνει γραμμή μενού και εργαλειών, ενώ το σώμα αποτελείται από κελιά κώδικα (εδώ με εντολές SPARQL) και εξόδους (π.χ. πίνακες αποτελεσμάτων). Η απλότητα της διεπαφής του Notebook διευκολύνει τον χρήστη να εστιάσει στον κώδικα και τα δεδομένα.**

**Πλεονεκτήματα του Jupyter Notebook:**

- *Ευκολία χρήσης και μαθησιακή καμπύλη:* Το περιβάλλον του Notebook είναι φιλικό ακόμη και για αρχάριους. Ένας χρήστης με βασικές γνώσεις Python μπορεί εύκολα να ξεκινήσει ένα notebook και να αρχίσει να γράφει κώδικα, βλέποντας άμεσα αποτελέσματα. Αυτή η αμεσότητα το έκανε εξαιρετικά δημοφιλές στην εκπαίδευση (π.χ. πανεπιστημιακά μαθήματα προγραμματισμού/ανάλυσης δεδομένων).

- *Συνδυασμός κώδικα και αφήγησης:* Όπως τονίστηκε, το Notebook ευνοεί την δημιουργία αυτοτελών "ιστοριών δεδομένων" όπου το κείμενο εξηγεί τα βήματα της ανάλυσης και ο κώδικας τα υλοποιεί. Αυτό το χαρακτηριστικό self-documentation προωθεί την κατανόηση και την συνεργασία [10].

- *Διαδραστικότητα και πειραματισμός:* Οι χρήστες μπορούν να εκτελούν τμήματα του κώδικα ανεξάρτητα, να τροποποιούν παραμέτρους και να ξανατρέχουν τα κελιά άμεσα. Αυτή η μη-γραμμική ροή εργασίας επιταχύνει τον πειραματισμό σε αναλύσεις δεδομένων – δεν χρειάζεται να επανατράχει ολόκληρο το πρόγραμμα για να δοκιμάσει κανείς μια αλλαγή.

- *Κοινότητα και διαθέσιμα παραδείγματα:* Υπάρχουν χιλιάδες διαθέσιμα notebooks σε αποθετήρια (GitHub, Kaggle κ.λπ.) που οι νέοι χρήστες μπορούν να μελετήσουν και να τροποποιήσουν. Η κοινότητα γύρω από το Jupyter Notebook είναι τεράστια και ενεργή, παρέχοντας υποστήριξη και μοιράζοντας βέλτιστες πρακτικές.

- *Ενσωμάτωση με εργαλεία ανάλυσης:* Πολλά βιβλιογραφικά εργαλεία δεδομένων (π.χ. **pandas** για dataframes, **scikit-learn** για machine learning, **TensorFlow**/**PyTorch** για deep learning) δουλεύουν άριστα μέσα σε notebooks. Επιπλέον, βιβλιοθήκες για **interactive widgets** (ipywidgets) επιτρέπουν τη δημιουργία απλών διεπαφών (κουμπιά, sliders) μέσα στο notebook για διαδραστικό έλεγχο των παραμέτρων.

**Περιορισμοί και προκλήσεις του Jupyter Notebook:**

Παρά τα πλεονεκτήματά του, το Notebook έχει επισημανθεί και για ορισμένα μειονεκτήματα, ειδικά όταν χρησιμοποιείται σε πιο σύνθετες ή συνεργατικές ροές εργασίας :

- *Διαχείριση κατάστασης και «κρυφή» κατάσταση:* Επειδή τα κελιά μπορούν να εκτελεστούν με αυθαίρετη σειρά, είναι εύκολο να δημιουργηθεί σύγχυση ως προς την κατάσταση του περιβάλλοντος (ποιες μεταβλητές έχουν ποια τιμή, ποια κελιά έχουν εκτελεστεί). Αυτό μπορεί να οδηγήσει σε **απρόβλεπτη συμπεριφορά** αν ο χρήστης δεν εκτελεί τα κελιά με τη σωστή σειρά ή αν παραλείψει κάποιο. Με άλλα λόγια, η ροή εκτέλεσης δεν είναι πάντα γραμμική, γεγονός που μπορεί να δυσκολέψει την αναπαραγωγή των αποτελεσμάτων εάν κάποιος άλλος απλώς «τρέξει όλα τα κελιά από την αρχή».

- *Ενθάρρυνση κακών πρακτικών κώδικα:* Έχει επισημανθεί ότι η ευκολία του Notebook μπορεί να οδηγήσει σε λιγότερο δομημένο κώδικα. Για παράδειγμα, κάποιος μπορεί να εκτελεί πολλές δοκιμές στο ίδιο notebook, αφήνοντας πίσω μεταβλητές ή κώδικα που δεν χρησιμοποιείται, αντί να οργανώσει τον κώδικα σε συναρτήσεις ή modules. Αυτό μπορεί να **υποβαθμίσει την ποιότητα του κώδικα** και να κάνει δύσκολη την συντήρησή του [11].

- *Δυσκολία στον έλεγχο εκδόσεων (version control):* Αν και το `.ipynb` είναι απλό κείμενο (JSON), η σύγκριση εκδόσεων (diff) δύο notebooks δεν είναι εύκολη υπόθεση, ειδικά λόγω των ενσωματωμένων outputs που μπορεί να περιέχονται στο αρχείο. Υπάρχουν εργαλεία (π.χ. nbdime) που βελτιώνουν την κατάσταση, αλλά σε μεγάλα έργα λογισμικού πολλοί προγραμματιστές θεωρούν τα notebooks δύσχρηστα στη συνεργασία μέσω Git.

- *Φόρτωση και απόδοση:* Ένα notebook που περιέχει μεγάλο αριθμό εξόδων (π.χ. πολλά γραφήματα ή μεγάλο όγκο δεδομένων σε εκτύπωση) μπορεί να γίνει βαρύ για τον browser. Καθώς όλο το περιεχόμενο (inputs/outputs) είναι ενσωματωμένο, αρχεία notebooks πολλών MB μπορεί να καθυστερούν κατά το άνοιγμα. Επίσης, το κλασικό Notebook δεν έχει μηχανισμό οργάνωσης σε καρτέλες – κάθε notebook ανοίγει σε δικό του παράθυρο browser, που μπορεί να οδηγήσει σε "παραφωνία παραθύρων" αν δουλεύει κανείς με πολλά notebooks ταυτόχρονα.

- *Περιορισμένη λειτουργικότητα IDE:* Το περιβάλλον notebook υστερεί σε σχέση με ένα πλήρες IDE ως προς ορισμένες δυνατότητες, όπως η προηγμένη αυτόματη συμπλήρωση κώδικα, ο εντοπισμός σφαλμάτων (debugging) με breakpoint, η πλοήγηση σε πολλαπλά αρχεία κ.λπ. Αν και υπάρχουν επεκτάσεις και εργαλεία (π.χ. το **JupyterLab** βελτιώνει πολλά

από αυτά, και IDEs όπως Visual Studio Code υποστηρίζουν notebooks), το κλασικό Notebook από μόνο του παραμένει βασικό σε αυτά που προσφέρει.

Παρά τα παραπάνω, το **Jupyter Notebook** παραμένει ένα πανίσχυρο εργαλείο που έχει αλλάξει τον τρόπο που γίνεται η επιστήμη δεδομένων. Οι **σημαντικές ωφέλειες στην τεκμηρίωση και την αναπαραγωγιμότητα** συχνά υπερκερούν τους κινδύνους, ειδικά όταν οι χρήστες ακολουθούν καλές πρακτικές (όπως η τακτική επανεκκίνηση του kernel και εκτέλεση όλων των κελιών για έλεγχο, η διαχωρισμός μεγάλων έργων σε αρθρωτά notebooks ή modules Python κ.λπ.). Πολλές από τις προκλήσεις που αναφέρθηκαν έχουν αποτελέσει αντικείμενο έρευνας και βελτίωσης [10], οδηγώντας και στην ανάπτυξη του JupyterLab ως λύσης σε ορισμένα από αυτά. Στην επόμενη ενότητα, εξετάζουμε το **Jupyter Lab** ακριβώς υπό αυτό το πρίσμα: ως την επόμενη γενιά σημειωματαρίων που επιχειρεί να διατηρήσει τα πλεονεκτήματα του Notebook ενώ αντιμετωπίζει κάποιους από τους περιορισμούς του.

## 3.2 **Jupyter Lab** – Δομή, επεκτασιμότητα και λειτουργίες

Το **Jupyter Lab** αποτελεί τη νεότερη εξέλιξη του Project Jupyter και μπορεί να θεωρηθεί ως ένα **ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για δεδομένα** μέσα στον browser. Σχεδιάστηκε ώστε να προσφέρει όλα τα γνώριμα στοιχεία του κλασικού **Jupyter Notebook** (κελιά, kernels, κώδικας/κείμενο, outputs) αλλά σε μια πολύ πιο ευέλικτη και επεκτάσιμη διεπαφή [5]. Το JupyterLab εισάγει μια δομή όπου ο χρήστης δεν περιορίζεται σε ένα μόνο notebook ανά παράθυρο: αντιθέτως, υπάρχει μια *κεντρική περιοχή εργασίας* με καρτέλες και παράθυρα που μπορούν να διαταχθούν κατά βούληση.

**Βασικά χαρακτηριστικά και δομή του Jupyter Lab:**

- **Πολλαπλά έγγραφα και πάνελ:** Στο κέντρο της οθόνης, το Jupyter Lab διαθέτει μια περιοχή όπου μπορούν να ανοιχτούν πολλαπλά notebooks, αρχεία κώδικα (script π.χ. .py), αρχεία κειμένου/Markdown, ακόμη και αρχεία CSV ή εικόνες. Ο χρήστης μπορεί να τα βλέπει παράλληλα, π.χ. δύο notebooks δίπλα-δίπλα για σύγκριση, ή ένα notebook αριστερά και έναν επεξεργαστή κώδικα δεξιά.

- **Πλευρική στήλη (sidebar):** Στα αριστερά, υπάρχει μια πτυσσόμενη πλαϊνή στήλη που περιλαμβάνει εργαλεία όπως τον **file browser** (πλοηγό αρχείων στον τρέχοντα κατάλογο εργασίας), τη λίστα **running kernels and terminals** (όλοι οι ενεργοί πυρήνες και τερματικά), το **command palette** (μια γραμμή εντολών όπως στα IDE για γρήγορη πρόσβαση σε λειτουργίες) και άλλα. Αυτή η στήλη επιτρέπει γρήγορη εναλλαγή μεταξύ αρχείων και επισκόπηση του περιβάλλοντος εργασίας.

- **Μενού και συντομεύσεις:** Το κορυφαίο μενού του Jupyter Lab είναι πιο πλούσιο από του κλασικού Notebook, συμπεριλαμβάνοντας επιλογές για προβολή (π.χ. διάταξη των πάνελ), ενέργειες σε πολλαπλές καρτέλες, κ.ά. Το Jupyter Lab υιοθετεί επίσης *συντομεύσεις πληκτρολογίου* παρόμοιες με ενός επιτραπέζιου IDE, επιταχύνοντας την εργασία.

- **Launcher και tabs:** Όταν ανοίγει το JupyterLab, εμφανίζει έναν πίνακα εκκίνησης (Launcher) με κουμπιά για δημιουργία νέου Notebook (επιλογή kernel), τερματικού, ή ανεβάσματος αρχείου. Κάθε ανοιχτό στοιχείο (notebook, αρχείο, κλπ) εμφανίζεται ως καρτέλα στην κορυφή. Η διάταξη των καρτελών είναι *drag-and-drop*: μπορείτε να σύρετε μια καρτέλα για να ανοίξει δίπλα σε μια άλλη (κάνοντας split view) ή σε ξεχωριστό παράθυρο.

- **Ενσωματωμένος τερματικός και κειμεδογράφος:** Σε αντίθεση με το κλασικό Notebook, το JupyterLab ενσωματώνει πλήρως τερματικό (bash/sh) και απλό κειμεδογράφο. Έτσι, ο χρήστης μπορεί μέσα στο ίδιο περιβάλλον να επεξεργάζεται έναν κώδικα Python module (.py file) ή να εκτελεί εντολές κελύφους, χωρίς να φύγει από την εφαρμογή.

- **Σύνδεση καρτελών (linking):** Μια ενδιαφέρουσα δυνατότητα είναι ότι μπορείτε να ανοίξετε το ίδιο notebook ή dataset σε δύο θέες (views) ταυτόχρονα. Π.χ. να έχετε έναν κώδικα Python script ανοιχτό σε έναν editor και παράλληλα ένα console που χρησιμοποιεί το ίδιο kernel για δοκιμές.

Η αρχιτεκτονική του **Jupyter Lab** "κάθεται πάνω" στην αρχιτεκτονική του Jupyter Notebook. Δηλαδή, χρησιμοποιεί κι αυτό έναν server (ο ίδιος server Jupyter, με μερικές επιπλέον επεκτάσεις) και kernels. Η κύρια διαφορά είναι στο frontend, που είναι μια **μοντέρνα εφαρμογή JavaScript (UI)** γραμμένη με το πλαίσιο **Lumino** (TypeScript) η οποία φορτώνεται στον browser. Αυτή η εφαρμογή είναι σχεδιασμένη ως *modular plugin system*. Συγκεκριμένα, το JupyterLab αποτελείται από πολλαπλά **plugins** που υλοποιούν κάθε τμήμα της λειτουργικότητας (π.χ. ένα plugin για το file browser, ένα για το notebook panel, ένα για το terminal, κ.ο.κ.) [5]. Αυτή η αρχιτεκτονική προσφέρει δύο μεγάλα οφέλη:

1. **Επεκτασιμότητα:** Οι προγραμματιστές τρίτων μπορούν να γράψουν δικά τους plugins/επεκτάσεις που ενσωματώνονται αρμονικά στο JupyterLab. Για παράδειγμα, υπάρχει επέκταση που προσθέτει περιβάλλον διαγράμμισης Git, επέκταση για integration με το Google Drive, θεματικές επεκτάσεις (dark theme) κ.ά. Το JupyterLab παρέχει APIs για να δημιουργεί κανείς νέες καρτέλες, να προσθέτει στοιχεία στο μενού, στο command palette, κλπ. Έτσι, θεωρητικά, το περιβάλλον μπορεί να προσαρμοστεί πλήρως στις ανάγκες του χρήστη ή του οργανισμού.

2. **Συντήρηση και απόδοση:** Ο κώδικας του frontend είναι πιο οργανωμένος σε modules, διευκολύνοντας τη συντήρηση. Παράλληλα, επειδή φορτώνονται μόνο τα απαραίτητα κομμάτια, το JupyterLab μπορεί να είναι αποδοτικότερο καθώς μεγαλώνει το project. Αξίζει να σημειωθεί ότι μεταγενέστερες εκδόσεις (JupyterLab 3 και 4) έχουν επικεντρωθεί πολύ στην **βελτιστοποίηση απόδοσης** – π.χ. η φόρτωση μεγάλων notebooks είναι ταχύτερη σε Lab, και η χρήση μνήμης στο browser μειωμένη σε σχέση με παλαιότερες εκδόσεις [12].

**Ενδεικτικές λειτουργίες και επεκτάσεις στο JupyterLab:**

- *Σύστημα αρχείων:* Επιτρέπει άνοιγμα αρχείων διαφόρων τύπων. Για JSON, CSV κ.λπ. έχει ενσωματωμένους viewers (π.χ. πίνακας για CSV). Για εικόνες, εμφανίζει τις εικόνες. Υπάρχουν plugins για προβολή GeoJSON χαρτών, για μουσικά MIDI files, κ.ά.

- *Συντάκτης κώδικα:* Πλήρης editor με επισήμανση σύνταξης (syntax highlighting) για δεκάδες γλώσσες, αυτόματη εσοχή κώδικα, και (μέσω επεκτάσεων) υποστήριξη autocompletion με **LSP (Language Server Protocol)** ώστε να παρέχονται IDE-like βοήθειες (π.χ. intellisense).

- *Ολοκλήρωση με **debugger***: Στο JupyterLab 3 προστέθηκε υποστήριξη για γραφικό debugger (για κατάλληλους kernels π.χ. xeus-Python). Έτσι, μπορεί κανείς να θέτει breakpoints σε κώδικα μέσα στο notebook και να ελέγχει τη στοίβα κλήσεων, μεταβλητές κ.λπ. (κάτι αδύνατο στο κλασικό Notebook χωρίς εξωτερικό εργαλείο).

- *Real-time collaboration:* Πειραματικό αλλά σημαντικό – δυνατότητα πολλοί χρήστες να ανοίγουν το ίδιο notebook και να βλέπουν σε πραγματικό χρόνο τις αλλαγές

(παρόμοιο με Google Docs). Αυτό απευθύνεται σε εκπαιδευτικά σενάρια ή σύγχρονη συνεργασία data scientists.

- *Ενσωμάτωση τρίτων υπηρεσιών:* Π.χ. επεκτάσεις για AWS ή GCP που επιτρέπουν φόρτωση δεδομένων από cloud storage μέσα από το UI, ή σύνδεση με βάσεις δεδομένων (π.χ. ένα μικρό SQL browser plugin).

- *Themes και εμφάνιση:* Δυνατότητα αλλαγής θέματος (Light/Dark κ.λπ.) και διάταξης (π.χ. Zen Mode για εστίαση).

Είναι σαφές ότι το **Jupyter Lab** επιχειρεί να προσφέρει μια πιο **ολοκληρωμένη εμπειρία** για τον χρήστη που δουλεύει σοβαρά με notebooks. Αν το κλασικό Notebook ήταν ένα εξαιρετικό σημειωματάριο, το Lab είναι ένα ολόκληρο "σημειωματάριο εργασίας". Πολλοί το έχουν περιγράψει ως τον «VS Code των notebooks» λόγω της ευελιξίας του. Σημειώνεται ότι το Lab είναι πλήρως συμβατό με τα υπάρχοντα notebooks: χρησιμοποιεί ακριβώς το ίδιο format .ipynb και την ίδια υποδομή kernels. Έτσι, οι χρήστες μπορούν να εναλλάσσονται μεταξύ Notebook και Lab ανάλογα με τις ανάγκες τους – στην πραγματικότητα, το JupyterLab μπορεί να ανοίγει notebooks ακριβώς όπως το κλασικό περιβάλλον, απλώς παρέχει περισσότερη εργονομία.

**Πλεονεκτήματα του JupyterLab εν συγκρίσει με το Notebook:**

- *Βελτιωμένη εμπειρία χρήστη:* Η ικανότητα προβολής πολλαπλών αρχείων ταυτόχρονα αυξάνει την παραγωγικότητα. Για παράδειγμα, μπορεί κανείς να βλέπει τα δεδομένα σε ένα CSV, να γράφει κώδικα που τα επεξεργάζεται σε notebook δίπλα, και να καταγράφει παρατηρήσεις σε ένα Markdown αρχείο ταυτόχρονα.

- *Καλύτερη οργάνωση έργου:* Με το file browser και την ιεραρχική οπτική των φακέλων, το Lab επιτρέπει διαχείριση πολλών notebooks και scripts μέσα σε ένα έργο. Στο κλασικό Notebook αυτό ήταν πιο άβολο, καθώς έπρεπε κανείς να ανοίγει πολλά παράθυρα.

- *Επεκτασιμότητα/Παραμετροποίηση:* Όπως συζητήθηκε, το Lab είναι παραμετροποιήσιμο μέσω επεκτάσεων. Αυτό σημαίνει ότι μπορεί να εξελιχθεί με τον χρόνο χωρίς να περιμένουμε αλλαγές στον βασικό κώδικα – η κοινότητα ήδη έχει δημιουργήσει δεκάδες χρήσιμα extensions.

- *Αντιμετώπιση ορισμένων περιορισμών:* Π.χ. το πρόβλημα της «κρυφής κατάστασης» παραμένει εγγενές σε notebooks, αλλά το Lab βοηθά με το command palette και tools που κάνουν restart kernel & run all πιο εύκολα, ή το debugger που επιτρέπει καλύτερη κατανόηση της ροής. Επίσης, η απόδοση είναι καλύτερη με μεγάλα outputs.

- *Σύγχρονη τεχνολογική βάση:* Το JupyterLab, όντας νεότερο, εκμεταλλεύεται σύγχρονες τεχνικές web development. Αυτό σημαίνει καλύτερη υποστήριξη σε νέα πρότυπα, διορθώσεις ασφαλείας, και συμβατότητα στο μέλλον. Ο σχεδιασμός του επίσης το κάνει πιο έτοιμο να ενσωματώσει αιτήματα όπως η συνεργατική σύνταξη.

Φυσικά, το **Jupyter Lab** δεν είναι πανάκεια – φέρει και αυτό κάποια πολυπλοκότητα (μια ελαφρώς μεγαλύτερη καμπύλη εκμάθησης για τον εντελώς αρχάριο, λόγω των πολλών επιλογών). Επίσης, ορισμένοι χρήστες έχουν αναφέρει ότι σε ορισμένα σενάρια το Lab μπορεί να είναι βαρύτερο (π.χ. σε παλαιότερους υπολογιστές με ελάχιστη μνήμη, το κλασικό

Notebook ίσως τρέχει πιο ελαφρά). Ωστόσο, η εξέλιξη των εκδόσεων έχει μειώσει σημαντικά τέτοια προβλήματα και το JupyterLab θεωρείται πλέον ώριμο για γενική χρήση.

## 3.3 Σύγκριση μεταξύ **Jupyter Notebook** και **Jupyter Lab** (Απόδοση, εμπειρία χρήσης, επεκτασιμότητα, διαδραστικότητα)

Έχοντας περιγράψει τα χαρακτηριστικά των δύο εργαλείων, συνοψίζουμε συγκριτικά τα κύρια σημεία διαφοροποίησης:

- **Εμπειρία Χρήστη (UX):** Το κλασικό **Jupyter Notebook** προσφέρει έναν απλούστερο, μονοπαραθυρικό σχεδιασμό, που είναι εύκολος και άμεσος – ιδανικός για γρήγορες δοκιμές ή παρουσίαση ενός μόνο θέματος. Το **Jupyter Lab**, από την άλλη, προσφέρει έναν πολυπαραθυρικό, ολοκληρωμένο χώρο εργασίας. Για έναν αρχάριο, το Notebook μπορεί να είναι λιγότερο αποθαρρυντικό. Για έναν προχωρημένο χρήστη ή ένα μεγάλο project, το Lab παρέχει καλύτερη οργάνωση και εργονομία. Ενδεικτικά, σε έρευνες ικανοποίησης χρηστών, πολλοί εκτιμούν την δυνατότητα του Lab να έχουν "όλα σε ένα μέρος" και θεωρούν ότι, αφού το συνηθίσουν, δύσκολα επιστρέφουν στο παλαιό περιβάλλον λόγω της βελτιωμένης ροής εργασίας.

- **Απόδοση:** Σε επίπεδο υπολογιστικής εκτέλεσης, δεν υπάρχει διαφορά – και τα δύο χρησιμοποιούν τους ίδιους kernels. Ωστόσο, στην απόδοση της διεπαφής υπάρχουν μικρές διαφοροποιήσεις. Το **Jupyter Lab** με τις νεότερες βελτιστοποιήσεις του είναι αρκετά αποδοτικό και μπορεί να διαχειριστεί μεγάλα notebooks με πολλαπλά outputs καλύτερα από το κλασικό Notebook. Παρ’ όλα αυτά, το Lab καταναλώνει λίγο περισσότερη μνήμη αρχικά, καθώς φορτώνει τη modular εφαρμογή του. Σε πρακτικούς όρους, σε έναν σύγχρονο υπολογιστή η διαφορά είναι αμελητέα. Για βαριά χρήση (π.χ. 10 notebooks ανοιχτά ταυτόχρονα, μεγάλα dataframes εμφανισμένα), το Lab δείχνει να υπερέχει καθώς οργανώνει σε καρτέλες και δεν κρατά τα πάντα ως μία τεράστια σελίδα DOM στον browser. Με άλλα λόγια, το Notebook μπορεί να γίνει αργό αν προσπαθεί κανείς να τα κάνει όλα σε ένα notebook, εκεί που το Lab θα πρότεινε κατανομή εργασίας σε πολλά πάνελ.

- **Επεκτασιμότητα και δυνατότητες προσαρμογής:** Εδώ το **Jupyter Lab** υπερέχει ξεκάθαρα. Η αρχιτεκτονική του με plugins σημαίνει ότι ο χρήστης/διαχειριστής μπορεί να το **εξατομικεύσει** προσθέτοντας λειτουργίες. Το κλασικό Notebook επίσης υποστήριζε επεκτάσεις (μέσω του *nbextensions* framework), αλλά αυτές ήταν συχνά "hacks" στην JavaScript του front-end και λιγότερο αξιόπιστες ή δύσκολες στην

εγκατάσταση. Το Lab έχει επίσημο σύστημα επέκτασης με pip/conda install ή διανομή μέσω npm, καθιστώντας το οικοσύστημά του πιο υγιές. Για παράδειγμα, αν θέλει κάποιος υποστήριξη **TeX** συντάκτη ή integration με **Draw.io**, στο Lab υπάρχει επίσημη επέκταση. Στο Notebook, θα έπρεπε να αρκεστεί σε εξωτερικά εργαλεία.

- **Διαδραστικότητα και συνεργασία:** Και τα δύο περιβάλλοντα παρέχουν τον ίδιο βαθμό διαδραστικότητας όσον αφορά την εκτέλεση κώδικα και την άμεση απόκριση. Ωστόσο, το **Jupyter Lab** κάνει ένα βήμα παραπέρα με μελλοντοστραφή χαρακτηριστικά συνεργασίας (real-time collaboration) που ήδη δοκιμάζονται. Επίσης, στην τάξη ή σε workshops, το Lab μπορεί να διευκολύνει τους εκπαιδευτές να έχουν πολλές σημειώσεις ανοιχτές. Από την άλλη πλευρά, το κλασικό Notebook είναι συμβατό με υπηρεσίες όπως το **Binder** εδώ και περισσότερο καιρό – αν και πλέον και το Lab υποστηρίζεται.

- **Βαθμός ωριμότητας:** Το Notebook υπάρχει ως περιβάλλον από το 2011+ (ως IPython Notebook) και σταθεροποιήθηκε πλήρως. Το Lab, παρότι πλέον ώριμο, πέρασε φάσεις έντονης ανάπτυξης (beta μέχρι το 2018). Σήμερα, θεωρείται παραγωγικό, αλλά μικρά bugs ή αλλαγές μπορεί να εμφανίζονται πιο συχνά απ’ ό,τι σε κάτι "ακίνητο" όπως το Notebook κλασικό. Οι προγραμματιστές του Jupyter έχουν ανακοινώσει ότι το μέλλον είναι το JupyterLab και ότι το κλασικό notebook (έκδοση 7) ουσιαστικά θα είναι μια συμβατότητα layer που τρέχει μέσα από το Lab backend. Αυτό δείχνει την κατεύθυνση προς την οποία συγκλίνει η ανάπτυξη: ενοποίηση στο Lab.

Συνοψίζοντας, το **Jupyter Notebook** και το **Jupyter Lab** υπηρετούν τον ίδιο στόχο – την υποστήριξη διαδραστικού, αναπαραγώγιμου προγραμματισμού – με λίγο διαφορετική φιλοσοφία. Το Notebook δίνει προτεραιότητα στην απλότητα και την άμεση εστίαση σε *ένα* σύνολο κώδικα/αποτελεσμάτων κάθε φορά. Το Lab δίνει προτεραιότητα στην οργάνωση και επέκταση, αναγνωρίζοντας ότι οι χρήστες συχνά χρειάζονται πολλά πράγματα ταυτόχρονα.

Η επιλογή μεταξύ των δύο σε μεγάλο βαθμό εξαρτάται από το use case:

- Για **γρήγορες αναλύσεις** ή *μονοθεματικές* παρουσιάσεις (π.χ. ένα tutorial, ένα μικρό project), πολλοί προτιμούν το κλασικό **Jupyter Notebook** για την καθαρότητά του.

- Για **μεγάλα projects** με πολλά αρχεία, ή για χρήστες που θέλουν ένα πιο ολοκληρωμένο εργαλείο χωρίς να αλλάζουν συνεχώς περιβάλλον, το **Jupyter Lab** είναι συνήθως προτιμότερο.

- Στο πλαίσιο της **εκπαίδευσης**, τα πράγματα είναι μοιρασμένα: αρχικά ίσως διδάσκεται το Notebook για απλότητα, αλλά όλο και περισσότερα ιδρύματα μεταβαίνουν στο Lab λόγω της ευκολίας διαχείρισης τάξεων (π.χ. μέσω JupyterHub).

Αξίζει να σημειωθεί ότι επειδή το υποκείμενο οικοσύστημα είναι το ίδιο, δεν υπάρχει ουσιαστικός φραγμός: ένας οργανισμός μπορεί να υποστηρίζει και τα δύο. Για παράδειγμα, το **JupyterHub** (multi-user server) μπορεί να δώσει στους χρήστες δυνατότητα επιλογής αν θα ξεκινήσουν session σε Notebook ή Lab. Έτσι, η μετάβαση είναι ομαλή.

Στην πράξη, η τάση δείχνει ότι το **Jupyter Lab** κερδίζει συνεχώς έδαφος και ενδέχεται να αντικαταστήσει πλήρως το κλασικό Notebook στο μέλλον, με την έννοια ότι οι νέες λειτουργίες θα αναπτύσσονται κυρίως για αυτό. Ωστόσο, το απλό Notebook θα παραμένει ένα βασικό και χρήσιμο εργαλείο, ειδικά για σκοπούς στους οποίους σχεδιάστηκε να διαπρέψει – την απλότητα στην αφήγηση ενός μεμονωμένου υπολογιστικού αφήγηματος.

# ΚΕΦΑΛΑΙΟ 4 Εφαρμογές στην Επιστήμη Δεδομένων

Η χρήση των εργαλείων Jupyter Notebook και Jupyter Lab έχει εδραιωθεί ως κεντρικό κομμάτι της επιστήμης δεδομένων, προσφέροντας ένα διαδραστικό περιβάλλον ιδανικό για προεπεξεργασία δεδομένων, ανάλυση και οπτικοποίηση [7], [8]. Η ικανότητά τους να συνδυάζουν εκτελέσιμο κώδικα, αποτελέσματα και τεκμηρίωση σε ένα ενιαίο «σημειωματάριο» διευκολύνει την αναπαραγωγιμότητα και την κατανόηση των αναλύσεων από διαφορετικούς χρήστες [13]. Μάλιστα, το Jupyter Notebook αναγνωρίζεται ευρέως ως κρίσιμο εργαλείο για επαγγελματίες και εκπαιδευόμενους στην επιστήμη δεδομένων, χάρη στον διαδραστικό και αυτοτεκμηριωμένο χαρακτήρα του [19]. Στη συνέχεια εξετάζονται οι κύριες εφαρμογές των Notebook/Lab στην επιστήμη δεδομένων: από την προεπεξεργασία των δεδομένων και την ανάλυση με βιβλιοθήκες όπως NumPy [14] και Pandas [15], έως την οπτικοποίηση με Matplotlib [17] και Seaborn [18], καθώς και τις διαδραστικές αναλύσεις με widgets και dashboards [20], [21].

## 3.4 Προεπεξεργασία δεδομένων με **Jupyter Notebook** και **Jupyter Lab**

Η προεπεξεργασία δεδομένων είναι συνήθως το πρώτο βήμα σε ένα έργο επιστήμης δεδομένων, όπου τα δεδομένα φορτώνονται, καθαρίζονται και μετασχηματίζονται σε κατάλληλη μορφή για ανάλυση. Τα **Jupyter Notebook** και **Jupyter Lab** διευκολύνουν ιδιαιτέρως αυτή τη διαδικασία, παρέχοντας ένα περιβάλλον στο οποίο ο αναλυτής μπορεί να εκτελεί βήμα-βήμα εντολές και να βλέπει άμεσα τα αποτελέσματα κάθε ενέργειας [13]. Αυτή η άμεση ανατροφοδότηση επιτρέπει μια **διαδραστική** και επαναληπτική προσέγγιση στον καθαρισμό δεδομένων: για παράδειγμα, μπορεί κανείς να φορτώσει ένα σύνολο δεδομένων, να εμφανίσει μερικές γραμμές με τη συνάρτηση head() της **Pandas**, να εντοπίσει και να αντιμετωπίσει τυχόν ελλιπείς τιμές, και αμέσως να επανεκτελέσει το κελί για να επιβεβαιώσει τις αλλαγές. Η δυνατότητα σύνθεσης κώδικα και περιγραφικού κειμένου επιτρέπει επίσης στον αναλυτή να τεκμηριώνει κάθε βήμα της προεπεξεργασίας δίπλα στα αποτελέσματα, κάνοντας το notebook **αυτοτεκμηριωμένο** και ευκολότερο στην επανάχρηση από τρίτους [19].

Ένα βασικό πλεονέκτημα που παρέχει το **Jupyter** περιβάλλον στην προεπεξεργασία είναι η υποστήριξη πολλών δημοφιλών βιβλιοθηκών για ανάγνωση και μετασχηματισμό δεδομένων. Για παράδειγμα, η βιβλιοθήκη **Pandas** προσφέρει υψηλού επιπέδου δομές δεδομένων (**DataFrames**) και συναρτήσεις για φόρτωση αρχείων CSV/Excel, φιλτράρισμα γραμμών, διαχείριση τιμών που λείπουν, ομαδοποίηση δεδομένων κ.ά. [15]. Η ενσωμάτωση της Pandas στα **Jupyter notebooks** είναι άψογη – τα αποτελέσματα (π.χ. ένας πίνακας DataFrame) εμφανίζονται σε μορφοποιημένο πίνακα αμέσως κάτω από το κελί κώδικα, διευκολύνοντας τον έλεγχο της ορθότητας των μετασχηματισμών. Επιπλέον, στο **Jupyter Lab** ο χρήστης μπορεί να εκμεταλλευτεί το περιβάλλον πολλαπλών καρτελών: π.χ. να έχει ανοιχτό ένα CSV αρχείο σε προβολή πίνακα σε ένα πάνελ και το notebook σε διπλανό πάνελ, συγκρίνοντας τα αρχικά δεδομένα με τα μετασχηματισμένα [21].

Η διαδραστικότητα αυτή συμβάλλει και στον εντοπισμό προβλημάτων στα δεδομένα σε πρώιμο στάδιο. Ο αναλυτής μπορεί να εκτελεί σταδιακά τον κώδικα σε μπλοκ (κελιά) και να

εξετάζει τα ενδιάμεσα αποτελέσματα – για παράδειγμα, εάν σε κάποιο στάδιο της προεπεξεργασίας μια στήλη παρουσιάζει απρόσμενες τιμές, αυτό θα φανεί αμέσως στην έξοδο του κελιού, επιτρέποντας διορθωτικές κινήσεις πριν προχωρήσει η ανάλυση. Συνολικά, το **Jupyter Notebook/Lab** λειτουργεί ως ένα διαδραστικό εργαστήριο για την προεπεξεργασία: ο συνδυασμός κώδικα, αποτελεσμάτων και τεκμηρίωσης σε πραγματικό χρόνο **ενθαρρύνει την πειραματική διερεύνηση** των δεδομένων και διασφαλίζει ότι κάθε βήμα μετασχηματισμού καταγράφεται επακριβώς, διευκολύνοντας τόσο τον έλεγχο όσο και την αναπαραγωγή της διαδικασίας από άλλους.

## 3.5 Ανάλυση δεδομένων με **NumPy** και **Pandas**

Μετά την αρχική προετοιμασία των δεδομένων, το στάδιο της ανάλυσης απαιτεί αποδοτικά εργαλεία για αριθμητικούς υπολογισμούς και διαχείριση δεδομένων. Στο οικοσύστημα της **Python**, κεντρικό ρόλο παίζουν οι βιβλιοθήκες **NumPy** και **Pandas**, οι οποίες αξιοποιούνται άριστα μέσα στα **Jupyter notebooks** [14], [15].

Το **NumPy** παρέχει μια δομή για πολυδιάστατους πίνακες (**ndarrays**) και ένα πλήθος συναρτήσεων για ταχύτατες αριθμητικές πράξεις πάνω σε αυτά τα δομημένα δεδομένα. Αποτελεί τη θεμέλια βιβλιοθήκη για επιστημονικό υπολογισμό σε **Python** και τον πυρήνα πάνω στον οποίο έχει δομηθεί ολόκληρο το οικοσύστημα scientific Python [14]. Χρησιμοποιώντας **NumPy** μέσα σε ένα notebook, ο αναλυτής μπορεί να εκτελεί αποδοτικούς υπολογισμούς σε μεγάλους όγκους αριθμητικών δεδομένων – για παράδειγμα, πράξεις σε διανύσματα και μήτρες, υπολογισμό στατιστικών δεικτών, γραμμική άλγεβρα, μετασχηματισμούς Fourier κ.λπ. – με συντακτικό τρόπο παρόμοιο με μαθηματική γραφή. Η ταχύτητα της **NumPy** (χάρη σε υλοποίηση σε γλώσσα C στο παρασκήνιο) σε συνδυασμό με τη διαδραστικότητα του **Jupyter** δημιουργεί ένα ισχυρό εργαλείο για αριθμοκεντρικές αναλύσεις. Μάλιστα, η σημασία της **NumPy** είναι τέτοια που υπήρξε κεντρικό στοιχείο σε μερικά από τα μεγαλύτερα επιστημονικά επιτεύγματα της πρόσφατης εποχής, όπως το λογισμικό για την ανακάλυψη των βαρυτικών κυμάτων και την πρώτη απεικόνιση μελανής οπής [14]. Στο πλαίσιο της επιστήμης δεδομένων, η **NumPy** χρησιμοποιείται συχνά για υπολογισμό χαρακτηριστικών, κανονικοποιήσεις, ή ως βάση πάνω στην οποία κτίζονται άλλες βιβλιοθήκες (η **Pandas**, για παράδειγμα, υλοποιεί τις δομές της πάνω σε **NumPy arrays**) [15].

Η βιβλιοθήκη **Pandas** συμπληρώνει τη **NumPy** αναλαμβάνοντας τη διαχείριση δεδομένων σε μορφή πινάκων (πίνακες δύο διαστάσεων με ετικέτες – μοιάζουν με φύλλα Excel ή πίνακες SQL). Σχεδιάστηκε ώστε να είναι γρήγορη, ισχυρή, ευέλικτη και εύχρηστη για ανάλυση και διαχείριση δεδομένων [15]. Με την **Pandas**, ο χρήστης μπορεί εύκολα να φιλτράρει δεδομένα βάσει συνθηκών, να υπολογίσει νέες στήλες από υπάρχουσες, να ομαδοποιήσει και να συγκεντρώσει (aggregate) δεδομένα κατά κατηγορίες, να συγχωνεύσει σύνολα δεδομένων κ.ά., με λίγες μόνο γραμμές κώδικα.

Μέσα στα **Jupyter notebooks**, αυτές οι δυνατότητες της **Pandas** ενισχύονται από την άμεση οπτικοποίηση των αποτελεσμάτων: όταν τυπώνεται ένα **DataFrame**, το notebook το εμφανίζει σε μορφοποιημένο πίνακα με διάταξη στηλών και γραμμών, καθιστώντας ευκολότερη την ερμηνεία του σε σχέση με μια απλή εκτύπωση κειμένου. Επιπλέον, το Notebook επιτρέπει στον αναλυτή να γράψει περιγραφές ή συμπεράσματα ακριβώς κάτω από

κάθε έξοδο – π.χ. “βλέπουμε ότι η στήλη *Age* έχει μέση τιμή 35 έτη και τυπική απόκλιση 5.2” – συνδυάζοντας έτσι το αποτέλεσμα με την ανάλυσή του. Αυτός ο συνδυασμός «δεδομένων και αφήγησης» εντός του notebook είναι που μετατρέπει το **Jupyter** σε εργαλείο υποστήριξης συλλογισμού πάνω στα δεδομένα, επιτρέποντας στον αναλυτή όχι μόνο να εκτελεί τον κώδικα αλλά και να σκέφτεται και να “αφηγείται” ευρήματα μέσα στο ίδιο περιβάλλον [19].

Η **Pandas** έχει καθιερωθεί *de facto* ως το βασικό εργαλείο για ανάλυση πίνακων δεδομένων στην **Python**, τόσο στην ακαδημαϊκή όσο και στη βιομηχανική κοινότητα. Εισήχθη αρχικά από τον Wes McKinney το 2010 για να καλύψει την έλλειψη κατάλληλων δομών στην **Python** για στατιστική ανάλυση δεδομένων [16]. Έκτοτε έχει εξελιχθεί από κοινοπραξία ανοιχτού κώδικα (**Pandas Development Team**) και σήμερα διαθέτει πάνω από 100 εκατομμύρια λήψεις το μήνα, αποτελώντας το πρότυπο εργαλείο για διαχείριση επιχειρησιακών και επιστημονικών δεδομένων σε **Python** [15].

Συνολικά, η συνδυασμένη χρήση **NumPy** και **Pandas** μέσα στα **Jupyter notebooks** δίνει τη δυνατότητα στον αναλυτή να εκτελεί σύνθετους υπολογισμούς και μετασχηματισμούς δεδομένων σε ένα ενιαίο, διαδραστικό περιβάλλον. Η **NumPy** χειρίζεται τις αριθμητικές πράξεις μεγάλης κλίμακας με βέλτιστη απόδοση, ενώ η **Pandas** προσφέρει τα εργαλεία για εξερεύνηση και αναδιάρθρωση των δεδομένων σε ανώτερο επίπεδο [14]–[16]. Η διαδραστικότητα του Notebook επιτρέπει την ομαλή εναλλαγή μεταξύ των δύο: για παράδειγμα, μπορεί κανείς να χρησιμοποιήσει **NumPy** για να υπολογίσει έναν πίνακα αποτελεσμάτων (π.χ. μια μήτρα ομοιότητας) και αμέσως να ενσωματώσει αυτόν τον πίνακα σε ένα **Pandas DataFrame** για περαιτέρω ανάλυση ή συσχέτιση με άλλα δεδομένα, βλέποντας όλα τα ενδιάμεσα αποτελέσματα [19].

## 3.6 Οπτικοποίηση δεδομένων με **Matplotlib** και **Seaborn**

Η οπτικοποίηση αποτελεί ζωτικό μέρος της επιστήμης δεδομένων, καθώς μέσω αυτής ο αναλυτής αντιλαμβάνεται πρότυπα και τάσεις στα δεδομένα και επικοινωνεί τα αποτελέσματα. Τα **Jupyter Notebook/Lab** διευκολύνουν την ενσωμάτωση γραφημάτων και διαγραμμάτων στην ανάλυση, αξιοποιώντας βιβλιοθήκες όπως η **Matplotlib** και η **Seaborn** [17], [18].

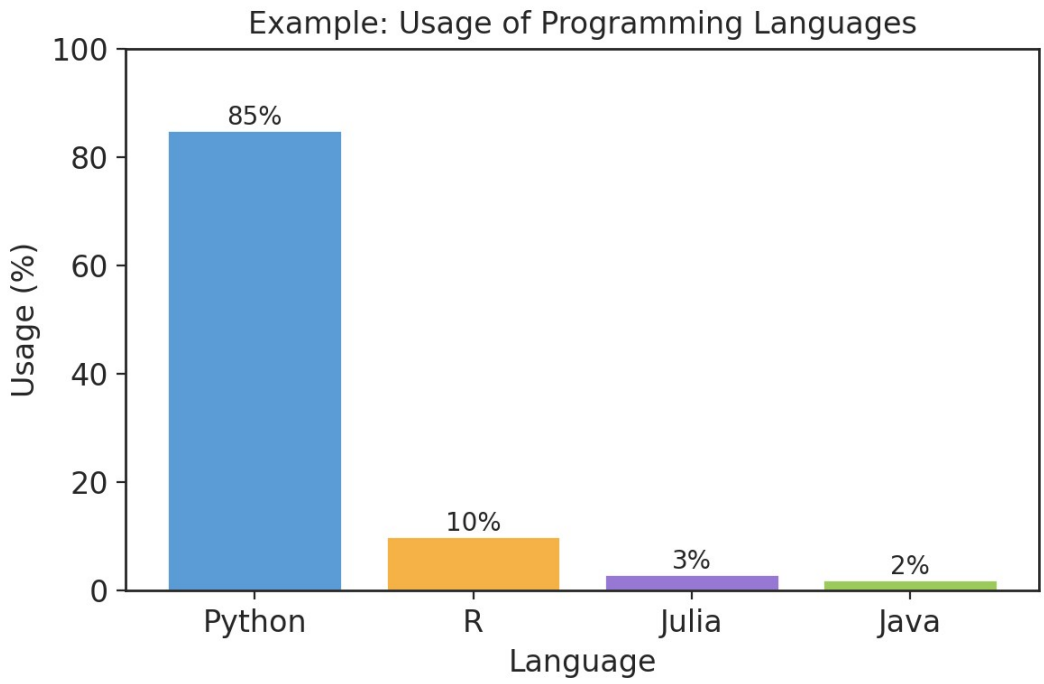
Η **Matplotlib** είναι η κλασική βιβλιοθήκη γραφικών της **Python**, προσφέροντας πληθώρα δυνατοτήτων για δημιουργία δισδιάστατων διαγραμμάτων – από απλά γραφήματα γραμμών και διασποράς, μέχρι σύνθετες προσαρμοσμένες visualizations. Έχει αναπτυχθεί με στόχο να παρέχει ένα περιβάλλον γραφικών ανάλογο με της **MATLAB**, υποστηρίζοντας τόσο διαδραστική χρήση σε σενάρια scripting όσο και παραγωγή γραφημάτων υψηλής ποιότητας για δημοσιεύσεις [17]. Στο πλαίσιο των **Jupyter notebooks**, η **Matplotlib** συνήθως χρησιμοποιείται με την εντολή %matplotlib inline (η οποία ενεργοποιεί την εμφάνιση των γραφημάτων εντός του notebook). Έτσι, κάθε φορά που ο χρήστης δημιουργεί ένα γράφημα με κώδικα **Matplotlib** (π.χ. plt.plot() ή plt.bar()), το αποτέλεσμα εμφανίζεται ακριβώς κάτω από το κελί κώδικα. Αυτό επιτρέπει έναν **γρήγορο κύκλο ανάδρασης**: ο αναλυτής μπορεί να δοκιμάζει διαφορετικές παραμετροποιήσεις (χρώματα, ετικέτες αξόνων, τύπους γραφήματος) και να βλέπει αμέσως πώς αλλάζει το διάγραμμα.

Η **Matplotlib** φημίζεται για την ευελιξία της – παρέχει χαμηλού επιπέδου έλεγχο σε κάθε στοιχείο του γραφήματος (τίτλοι, άξονες, σημεία δεδομένων, πλέγμα, υπομνήματα κ.λπ.), κάτι που την καθιστά ικανή να παράξει δημοσιεύσιμης ποιότητας γραφήματα [17]. Ωστόσο, αυτή η χαμηλού επιπέδου προσέγγιση σημαίνει ότι ο χρήστης πολλές φορές χρειάζεται να γράψει περισσότερο κώδικα για κοινές εργασίες οπτικοποίησης – για παράδειγμα, η δημιουργία ενός scatter plot με διαφορετικό σχήμα σημείων ανά κατηγορία απαιτεί ρητό βρόχο ή χρήση ξεχωριστών κλήσεων της plt.scatter για κάθε κατηγορία. Σε ανταπόκριση σε αυτή τη σχετική πολυπλοκότητα, αναπτύχθηκε η βιβλιοθήκη **Seaborn**, η οποία επεκτείνει τη **Matplotlib** παρέχοντας έναν υψηλότερου επιπέδου, δηλωτικό τρόπο παραγωγής στατιστικών γραφημάτων [18].

Η **Seaborn** ενσωματώνεται στενά με την **Pandas** και τη δομή **DataFrame**, διευκολύνοντας τη δημιουργία σύνθετων γραφημάτων που απαντούν σε ερωτήματα για τα δεδομένα με ελάχιστη προσπάθεια από τον χρήστη [18]. Προσφέρει μια **δηλωτική API προσανατολισμένη στα σύνολα δεδομένων**: ο χρήστης δηλώνει ποια δεδομένα θέλει να οπτικοποιήσει (π.χ. στήλες ενός πίνακα) και τον τύπο του γραφήματος, και η **Seaborn** αναλαμβάνει τα υπόλοιπα – π.χ. χαρτογραφεί αυτόματα τις μεταβλητές σε ιδιότητες του γραφήματος όπως το χρώμα ή το μέγεθος των σημείων, υπολογίζει εσωτερικά στατιστικές συναρτήσεις (όπως κατανομές ή παρεμβολές) και διαμορφώνει το γράφημα με κατατοπιστικούς άξονες και υπόμνημα [18]. Για παράδειγμα, με μία μόνο εντολή sns.pairplot(df, hue="Species") μπορεί κανείς να δημιουργήσει ένα πλέγμα διαγραμμάτων διασποράς για κάθε ζεύγος χαρακτηριστικών ενός συνόλου δεδομένων (όπως το γνωστό **Iris dataset**), με τα σημεία χρωματισμένα ανά κατηγορία είδους – κάτι που στη **Matplotlib** θα απαιτούσε πολλαπλές εντολές και ρυθμίσεις.

Η **Seaborn** είναι σχεδιασμένη για να καλύπτει όλον τον κύκλο ζωής ενός έργου: αφενός διευκολύνει τον **ταχύ σχεδιασμό** και την εξερευνητική ανάλυση (με προεπιλογές που παράγουν ολοκληρωμένα γραφήματα από μια μόνο κλήση συνάρτησης), αφετέρου προσφέρει δυνατότητες λεπτομερειακής **προσαρμογής** και πρόσβαση στα υποκείμενα αντικείμενα **Matplotlib** όταν απαιτείται τελική επιμέλεια για δημοσίευση [18]. Έτσι, ο αναλυτής μπορεί να ξεκινήσει γρήγορα με προκαθορισμένα γραφήματα για να διερευνήσει τα δεδομένα του, και στη συνέχεια να τροποποιήσει τις λεπτομέρειες (π.χ. χρωματικές παλέτες, στυλ γραφημάτων) ώστε να παραγάγει μια τελική εικόνα υψηλής ποιότητας.

**Στο ακόλουθο σχήμα** παρουσιάζεται ένα παράδειγμα γραφήματος που δημιουργήθηκε εντός ενός Jupyter notebook, συγκεκριμένα ένα απλό διάγραμμα στήλης το οποίο θα μπορούσε να προκύψει ως αποτέλεσμα διερευνητικής ανάλυσης (π.χ. κατανομή χρήσης γλωσσών προγραμματισμού σε ένα dataset προγραμματιστών). Το διάγραμμα αυτό εμφανίζεται άμεσα κάτω από τον κώδικα δημιουργίας του, χάρη στη λειτουργία inline των notebooks, επιτρέποντας στον χρήστη να αξιολογήσει επιτόπου την πληροφορία που μεταφέρει το γράφημα και να προβεί σε περαιτέρω ενέργειες (π.χ. τροποποίηση του κώδικα για διαφορετική οπτικοποίηση).




**Εικόνα 4.3 2 : Παράδειγμα γραφήματος εξόδου σε Jupyter Notebook: ένα απλό διάγραμμα στήλης που απεικονίζει την υποθετική κατανομή χρήσης διαφόρων γλωσσών προγραμματισμού. Το γράφημα δημιουργήθηκε με κώδικα Matplotlib και εμφανίζεται εντός του notebook αμέσως μετά την εκτέλεση του αντίστοιχου κελιού.**

Η ολοκλήρωση των εργαλείων οπτικοποίησης με το Jupyter μεγιστοποιεί την αποτελεσματικότητα της ανάλυσης. Ο αναλυτής μπορεί να δημιουργήσει ένα γράφημα, να το δει, και να το ενημερώσει άμεσα. Αν το γράφημα αποκαλύψει κάποιο μοτίβο στα δεδομένα, αυτό μπορεί να περιγραφεί αμέσως σε κείμενο κάτω από το γράφημα. Αν χρειάζεται διαφορετική απεικόνιση, ο κώδικας μπορεί να τροποποιηθεί και να εκτελεστεί εκ νέου, με το νέο αποτέλεσμα να αντικαθιστά το παλιό. Επιπλέον, τόσο η Matplotlib όσο και η Seaborn υποστηρίζονται από ένα οικοσύστημα επεκτάσεων που συνεργάζονται με τα notebooks: για παράδειγμα, η χρήση της βιβλιοθήκης plotly ή Bokeh μπορεί να δημιουργήσει διαδραστικά γραφήματα (με δυνατότητα zoom, hover tooltips κλπ.) τα οποία επίσης εμφανίζονται μέσα στα notebooks. Το JupyterLab μάλιστα επιτρέπει και ακόμα πιο σύνθετες διατάξεις οπτικοποίησης, όπως παράθεση πολλαπλών γραφημάτων σε διπλανά πάνελ, δημιουργία απλών «ταμπλό» (dashboards) εντός του workspace κ.ο.κ., προσφέροντας έτσι εργαλεία business intelligence σε κλίμακα εργαστηρίου για τον data scientist.

Συνολικά, η Matplotlib και η Seaborn παρέχουν στον επιστήμονα δεδομένων τα μέσα για να μετατρέψει τους αριθμούς σε εικόνες, και το Jupyter Notebook/Lab τούς δίνει τον καμβά για να το κάνει αυτό διαδραστικά και με συνεχή τεκμηρίωση. Η δυνατότητα εξαγωγής των γραφημάτων (σε μορφές PNG, SVG, PDF) από το notebook σημαίνει ότι τα αποτελέσματα μπορούν εύκολα να ενσωματωθούν σε αναφορές ή παρουσιάσεις. Επιπλέον, το notebook ως αρχείο μπορεί να διαμοιραστεί – και όποιος το ανοίξει (ακόμη και μέσω πλατφορμών όπως το GitHub ή NBViewer) θα δει τόσο τον κώδικα όσο και τα γραφήματα, εξασφαλίζοντας διαφάνεια και επαληθευσιμότητα στην επικοινωνία των ευρημάτων.

## 3.7 Διαδραστική ανάλυση με widgets και dashboards

Μία από τις πλέον καινοτόμες δυνατότητες που προσφέρει το οικοσύστημα Jupyter είναι η δημιουργία διαδραστικών στοιχείων ελέγχου (widgets) μέσα στα notebooks, καθώς και η μετατροπή ενός notebook σε αυτόνομο διαδραστικό dashboard. Τα widgets του Jupyter είναι στοιχεία διεπαφής χρήστη, όπως π.χ. ρυθμιστικά (sliders), μενού επιλογής (dropdown menus), κουμπιά, πλαίσια ελέγχου κ.ά., που μπορούν να ενσωματωθούν σε ένα κελί του notebook και να συνδεθούν με τον κώδικα Python. Μέσω του μηχανισμού αυτού, ο χρήστης δύναται να τροποποιεί δυναμικά παραμέτρους της ανάλυσης και να βλέπει αμέσως πώς μεταβάλλονται τα αποτελέσματα, χωρίς να απαιτείται επεξεργασία του κώδικα και επανεκτέλεση με μη αυτόματο τρόπο [20].

Το Jupyter περιβάλλον υλοποιεί τα widgets με μια αρχιτεκτονική δύο μερών: ένα κομμάτι “front-end” (στον browser) που υλοποιεί την παρουσίαση και τη διάδραση (π.χ. το slider που κινεί ο χρήστης) και ένα κομμάτι “back-end” (στο kernel Python) που αντιπροσωπεύει το μοντέλο των δεδομένων που ελέγχει το widget [20]. Όταν ο χρήστης κινεί, για παράδειγμα, το ρυθμιστικό, ένα μήνυμα αποστέλλεται μέσω του μηχανισμού του Jupyter στο kernel, όπου ενημερώνεται η αντίστοιχη τιμή (π.χ. μια μεταβλητή Python), και ακολούθως μπορεί να εκτελεστεί κώδικας (π.χ. επανασχεδιασμός ενός γραφήματος με βάση τη νέα τιμή). Αυτή η αμφίδρομη επικοινωνία επιτρέπει πλούσια διαδραστικότητα: οι ενέργειες του χρήστη στο UI μεταφράζονται άμεσα σε υπολογισμούς στο kernel και τα αποτελέσματα επιστρέφουν και εμφανίζονται στο notebook [20]. Στην πράξη, τα widgets λειτουργούν σαν “ζωντανές” παράμετροι του κώδικα.

Η υλοποίηση των widgets έχει καθιερωθεί μέσω της βιβλιοθήκης ipywidgets (Jupyter Widgets), η οποία παρέχει μια βασική συλλογή από κοινά στοιχεία διεπαφής (π.χ. IntSlider, Dropdown, Button, Checkbox κ.λπ.) και λειτουργίες για τη σύνδεσή τους με συναρτήσεις Python. Χάρη στην αρχιτεκτονική επεκτασιμότητας του Jupyter, έχουν δημιουργηθεί επίσης πολλά πρόσθετα πακέτα που παρέχουν εξειδικευμένα widgets – π.χ. για εμφάνιση γεωγραφικών χαρτών (ipyleaflet), για τρισδιάστατη απεικόνιση μορίων (ipyvolume) κ.ά. [20]. Σε κάθε περίπτωση, ο βασικός μηχανισμός παραμένει ο ίδιος: τα widgets επεκτείνουν τις δυνατότητες εξόδου του notebook προσθέτοντας διαδραστικές όψεις και ελέγχους που επικοινωνούν άμεσα με τον κώδικα και τα δεδομένα του χρήστη [20].

Για έναν επιστήμονα δεδομένων, αυτό σημαίνει ότι μπορεί εύκολα να μετατρέψει μια στατική ανάλυση σε ένα διαδραστικό εργαλείο διερεύνησης. Για παράδειγμα, ας υποθέσουμε ότι έχει αναπτυχθεί ένα μοντέλο μηχανικής μάθησης και ο αναλυτής θέλει να μελετήσει την επίδραση μιας παραμέτρου (π.χ. του αριθμού δέντρων σε ένα τυχαίο δάσος) στην απόδοση του μοντέλου. Αντί να τροποποιεί την τιμή στον κώδικα και να επανατρέχει το κελί χειροκίνητα πολλές φορές, μπορεί να ενσωματώσει ένα slider widget για τον αριθμό των δέντρων και να συνδέσει τη μεταβολή του με τον κώδικα εκπαίδευσης/αξιολόγησης του μοντέλου. Έτσι, σύροντας το slider, το μοντέλο θα επανεκπαιδεύεται αυτόματα και η απόδοση (π.χ. accuracy) θα εμφανίζεται άμεσα, επιτρέποντας διαδραστική πειραματική αξιολόγηση του υπερ-παραμέτρου.

Η δυνατότητα δημιουργίας dashboards προκύπτει ως φυσικό επακόλουθο των widgets. Ένα dashboard στην ουσία είναι μια συλλογή από visualizations και ελεγχόμενα φίλτρα/παραμέτρους που επιτρέπουν σε μη-τεχνικούς χρήστες να εξερευνήσουν τα δεδομένα ή τα αποτελέσματα ενός μοντέλου. Η κοινότητα του Jupyter έχει αναπτύξει το πλαίσιο Voilà,

το οποίο επιτρέπει την μετατροπή ενός Jupyter notebook σε αυτόνομη web εφαρμογή/dashboard. Μέσω του Voilà, όλα τα στοιχεία κώδικα και οι έξοδοι ενός notebook μπορούν να εμφανιστούν ως διαδραστική σελίδα – τα widgets μετατρέπονται σε πραγματικά στοιχεία φόρμας στο web, και τα γραφήματα/πίνακες παρουσιάζονται χωρίς να φαίνεται ο κώδικας πίσω τους [19].

Εκτός του Voilà, αξίζει να σημειωθεί ότι και το ίδιο το JupyterLab έχει δυνατότητες δημιουργίας απλών dashboards. Για παράδειγμα, οποιοδήποτε output (π.χ. ένα γράφημα ή ένας πίνακας) μπορεί να αποσπαστεί σε ξεχωριστή καρτέλα και να τοποθετηθεί δίπλα σε widgets εντός του χώρου εργασίας [21]. Αυτό σημαίνει ότι κάποιος μπορεί χειροκίνητα να οργανώσει ένα dashboard μέσα στο JupyterLab, χωρίς εξαγωγή – αν και για πιο επίσημη και καθαρή παρουσίαση το Voilà παραμένει η ενδεδειγμένη λύση [21].

Συμπερασματικά, τα Jupyter widgets και τα dashboards μετατρέπουν τα notebooks από στατικές αναφορές σε διαδραστικές εφαρμογές ανάλυσης. Η τεχνολογία αυτή αντικατοπτρίζει μια γενικότερη τάση προς πιο επαναλήψιμη, έμπρακτη ανάλυση δεδομένων – ο αναλυτής δεν δίνει απλώς αποτελέσματα, αλλά παρέχει ένα “περιβάλλον” όπου τα αποτελέσματα μπορούν να μεταβληθούν και να εξερευνηθούν περαιτέρω από τον ίδιο ή συνεργάτες/αποφασίζοντες [19], [20], [21].

# ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα και Μελλοντικές Προοπτικές

Σε αυτό το κεφάλαιο συνοψίζονται τα βασικά ευρήματα της μελέτης και διατυπώνονται συμπεράσματα σχετικά με τη χρήση του **Jupyter Notebook** και του **Jupyter Lab** στην επιστήμη δεδομένων. Παρουσιάζονται επίσης προτάσεις για την καταλληλότερη χρήση του κάθε εργαλείου ανάλογα με τις ανάγκες του χρήστη, καθώς και μια επισκόπηση μελλοντικών εξελίξεων στο οικοσύστημα **Jupyter**.

## 3.8 Συνοπτικά συμπεράσματα και βασικά ευρήματα

Η παρούσα εργασία ανέδειξε τον κεντρικό ρόλο των εργαλείων **Jupyter** στην επιστήμη δεδομένων και την ανάλυση γενικότερα. Τα **Jupyter Notebook** και **Jupyter Lab** παρέχουν ένα μοναδικό περιβάλλον που συνδυάζει κώδικα, δεδομένα, αποτελέσματα και τεκμηρίωση, διευκολύνοντας την υλοποίηση της ιδέας των **υπολογιστικών αφηγήσεων (computational narratives)** [19]. Με άλλα λόγια, επιτρέπουν στον αναλυτή όχι μόνο να εκτελεί πολύπλοκες τεχνικές εργασίες, αλλά και να τις παρουσιάζει σε ανθρώπινα κατανοητή μορφή, βοηθώντας τον να "σκεφτεί και να αφηγηθεί ιστορίες με κώδικα και δεδομένα" [19].

Μέσα από τα παραδείγματα των προηγούμενων κεφαλαίων, διαπιστώσαμε ότι:
Το **Jupyter Notebook** (κλασικό) προσφέρει μια ελαφριά, απλή διεπαφή ιδανική για γρήγορη έναρξη ανάλυσης και εκπαιδευτικούς σκοπούς. Η έμφαση δίνεται στην αμεσότητα: ο χρήστης επικεντρώνεται σε ένα σύνολο κελιών κώδικα/αποτελεσμάτων κάθε φορά, γεγονός που απλοποιεί τη ροή εργασίας όταν δεν απαιτείται παράλληλος συντονισμός πολλών αρχείων ή παραθύρων. Αυτή η απλότητα καθιστά τα notebooks εξαιρετικά για διδακτική χρήση, tutorials, αλλά και για ατομικές αναλύσεις όπου ο αναλυτής θέλει να καταγράψει τη σκέψη του μαζί με τα αποτελέσματα [7], [8].

Το **Jupyter Lab**, από την άλλη πλευρά, αποτελεί τη φυσική εξέλιξη του Notebook, διατηρώντας όλα τα βασικά του στοιχεία αλλά παρέχοντας μια πιο ισχυρή και επεκτάσιμη πλατφόρμα. Όπως αναλύθηκε, το Lab εισάγει ένα ολοκληρωμένο περιβάλλον τύπου **IDE** μέσα στον browser, με δυνατότητα ταυτόχρονης προβολής πολλαπλών εγγράφων, ενσωματωμένο τερματικό, διαχείριση αρχείων, και γενικότερα καλύτερη οργάνωση της εργασίας [12], [21]. Αυτό το καθιστά ιδιαίτερα κατάλληλο για σύνθετες εργασίες επιστήμης δεδομένων, όπου ο χρήστης συχνά χρειάζεται να εργάζεται με πολλά πράγματα ταυτόχρονα (κώδικα, δεδομένα, σημειώσεις, γραφήματα).

Η σύγκλιση των δύο εργαλείων ως προς τον τελικό σκοπό τους (υποστήριξη διαδραστικού και αναπαραγώγιμου προγραμματισμού) σημαίνει ότι, ανεξαρτήτως επιλογής, ο χρήστης επωφελείται από τα κοινά πλεονεκτήματά τους. Η διαδραστικότητα, η αμεσότητα αποτελεσμάτων, και η ενσωματωμένη τεκμηρίωση χαρακτηρίζουν τόσο το Notebook όσο και το Lab, δίνοντας στον επιστήμονα δεδομένων τη δυνατότητα να αναπτύξει μια ανάλυση από το μηδέν έως το τέλος μέσα στο ίδιο περιβάλλον [19].

Η ενσωμάτωση βασικών βιβλιοθηκών επιστήμης δεδομένων (όπως **NumPy**, **Pandas**, **Matplotlib**, **Seaborn**) είναι άριστη και στα δύο περιβάλλοντα. Αυτό σημαίνει ότι το **Jupyter** λειτουργεί ως κόμβος όπου συναντώνται όλα τα απαραίτητα εργαλεία: από την αποθήκευση

και επεξεργασία δεδομένων μέχρι τον στατιστικό υπολογισμό και την απεικόνισή τους, όλα μπορούν να γίνουν εντός του ίδιου εγγράφου [14]–[18]. Το πλεονέκτημα αυτό έχει τεράστια επίπτωση στην παραγωγικότητα – ο χρόνος μεταξύ σκέψης και πειραματισμού μειώνεται στο ελάχιστο, καθώς δεν απαιτείται εναλλαγή σε διαφορετικά εργαλεία ή γλώσσες περιγραφής (π.χ. η μετάβαση από κώδικα σε μια ξεχωριστή εφαρμογή γραφημάτων).

Συνολικά, η μελέτη επιβεβαίωσε ότι τα εργαλεία του **Project Jupyter** έχουν επαναστατικοποιήσει τον τρόπο με τον οποίο διεξάγεται η ανάλυση δεδομένων. Έχουν μετατρέψει την παραδοσιακή στατική προσέγγιση (κώδικας σε ένα αρχείο, αποτελέσματα αλλού, τεκμηρίωση σε αναφορές) σε μια δυναμική, ενοποιημένη διαδικασία όπου όλα συμβαίνουν μαζί [7], [19]. Αυτό όχι μόνο βελτίωσε τη ροή εργασίας των **data scientists**, αλλά συνέβαλε και στη διάδοση των πρακτικών **ανοιχτής επιστήμης** και **αναπαραγώγιμης έρευνας**, αφού η ανάλυση μπορεί να κοινοποιηθεί ολόκληρη (με κώδικα, δεδομένα και περιγραφές) ως ένα φορητό αρχείο [22], [27].

## 3.9 Προτάσεις χρήσης των εργαλείων Jupyter

Αν και το **Jupyter Notebook** και το **Jupyter Lab** μοιράζονται κοινούς στόχους και μεγάλο μέρος της λειτουργικότητάς τους, υπάρχουν διαφορές που καθιστούν το καθένα πιο κατάλληλο σε συγκεκριμένα σενάρια χρήσης. Με βάση τα όσα παρουσιάστηκαν, μπορούν να δοθούν οι ακόλουθες κατευθυντήριες προτάσεις:

**Χρήση του Jupyter Notebook (κλασικό):** Συνιστάται όταν η απλότητα και η εστίαση είναι πρωταρχικής σημασίας. Για νέους χρήστες ή εκπαιδευτικούς σκοπούς, το κλασικό περιβάλλον Notebook είναι ιδανικό επειδή έχει λιγότερα στοιχεία στην οθόνη και επιτρέπει στον χρήστη να επικεντρώνεται σε μία ακολουθία κελιών κώδικα-αποτελεσμάτων κάθε φορά. Εάν εργάζεστε σε ένα αυτόνομο **analysis** (π.χ. ένα σύντομο **exploratory data analysis**) ή γράφετε ένα tutorial/βιβλίο με κώδικα, το Notebook παρέχει ακριβώς αυτά που χρειάζεστε χωρίς πρόσθετη πολυπλοκότητα [7], [8]. Επίσης, σε υπολογιστικά περιβάλλοντα με περιορισμένους πόρους (π.χ. ένα μικρό **VM** στο **cloud**) το Notebook είναι ελαφρώς πιο λιτό σε απαιτήσεις από το Lab [21]. Με λίγα λόγια, χρησιμοποιήστε το Notebook όταν θέλετε "να μπείτε και να ξεκινήσετε γρήγορα" και δεν απαιτείται η διαχείριση πολλαπλών αρχείων/παραθύρων.

**Χρήση του Jupyter Lab:** Ενδείκνυται για πιο σύνθετα ή μεγάλης κλίμακας έργα, όπου η οργάνωση και η επεκτασιμότητα έχουν σημασία. Αν βρίσκεστε σε μια κατάσταση όπου χρειάζεται να επεξεργάζεστε ταυτόχρονα πολλαπλά **notebooks** ή/και **scripts**, να βλέπετε αρχεία δεδομένων παράλληλα με τον κώδικα, ή να έχετε ανοιχτό έναν τερματικό για εκτέλεση εντολών (π.χ. git, pip install), το **JupyterLab** είναι η καλύτερη επιλογή. Παρέχει ένα ενοποιημένο περιβάλλον εργασίας, παρόμοιο με ένα **desktop IDE**, όπου όλα τα σχετικά με το έργο σας μπορούν να συσσωρευτούν [12]. Για **data science** ομάδες ή έμπειρους χρήστες που αναπτύσσουν πολύπλοκα **pipelines**, το **JupyterLab** προσφέρει επίσης πληθώρα επεκτάσεων – π.χ. εργαλεία για διαχείριση περιβαλλόντων, **linting** κώδικα, ενσωμάτωση με **Git**, υποστήριξη **live collaboration** κ.ά. – καθιστώντας το ευέλικτο και προσαρμόσιμο στις ανάγκες του προχωρημένου χρήστη [21]. Σε πολλές σύγχρονες πλατφόρμες **cloud** (π.χ. **AWS SageMaker**, **Google Colab Pro**) ή on-premises εγκαταστάσεις (π.χ. **JupyterHub** πανεπιστημίων), το **JupyterLab** έχει γίνει η προεπιλεγμένη διεπαφή, γεγονός που

αντικατοπτρίζει την τάση οι χρήστες να μεταβαίνουν σε αυτό για την καθημερινή τους εργασία [24].

**Συνδυασμός και συνύπαρξη:** Αξίζει να σημειωθεί ότι τα δύο εργαλεία δεν είναι ασύμβατα – αντιθέτως, μπορούν να συνυπάρχουν και να χρησιμοποιούνται ανάλογα με την περίσταση. Μπορείτε να έχετε και τις δύο διεπαφές εγκατεστημένες και να εναλλάσσεστε: για γρήγορα πρόχειρα πειράματα να ανοίγετε ένα κλασικό Notebook, ενώ για την ενσωμάτωση του πειράματος σε ένα μεγαλύτερο έργο να μεταβαίνετε στο Lab. Η συμβατότητα των αρχείων notebook (.ipynb) είναι απόλυτη μεταξύ τους – ένα αρχείο notebook μπορεί να ανοιχτεί και να τρέξει εξίσου καλά και στο Notebook και στο Lab [21]. Έτσι, η επιλογή του εργαλείου γίνεται περισσότερο θέμα προσωπικής προτίμησης και workflow παρά τεχνικού περιορισμού. Στο μέλλον μάλιστα, όπως θα συζητηθεί, τα όρια μεταξύ των δύο πιθανότατα θα εξαλειφθούν περαιτέρω [19].

**JupyterHub και κοινοχρησία:** Όταν ο στόχος είναι να μοιραστείτε το **Jupyter** περιβάλλον με πολλούς χρήστες (π.χ. τάξη φοιτητών, ομάδα σε μια εταιρεία) σε έναν κεντρικό server, τότε το **JupyterHub** είναι το κατάλληλο εργαλείο. Παρέχει έναν τρόπο να σερβίρονται multi-user περιβάλλοντα **Notebook/Lab** μέσω web, με διαχείριση αυθεντικοποίησης, πόρων κλπ. [24]. Οι συστάσεις σε αυτήν την περίπτωση αφορούν περισσότερο την υποδομή, αλλά αξίζει να αναφερθεί ότι το **JupyterHub** μπορεί να σερβίρει είτε το κλασικό Notebook interface είτε το **JupyterLab** (ή και να δίνει επιλογή). Σε εκπαιδευτικά ιδρύματα, το **JupyterHub** με κλασικό Notebook συχνά προτιμάται για απλότητα προς τους φοιτητές, ενώ σε επαγγελματικά περιβάλλοντα ίσως ενεργοποιείται το **Lab** για μέγιστη αποδοτικότητα των χρηστών [22], [27].

**Ενσωμάτωση με άλλα εργαλεία:** Σε περίπτωση που κάποιος χρήστης προτιμά ένα παραδοσιακό **IDE** (όπως το **VS Code**) αλλά θέλει τα πλεονεκτήματα των **notebooks**, υπάρχει και η επιλογή ενσωμάτωσης – π.χ. το **VS Code** υποστηρίζει άνοιγμα και εκτέλεση **Jupyter notebooks** εγγενώς [21]. Έτσι, ένας προγραμματιστής μπορεί να παραμείνει στο IDE του και ταυτόχρονα να χρησιμοποιεί **notebooks** όταν χρειάζεται. Αυτή η ευελιξία δείχνει και τη διάχυση της τεχνολογίας **Jupyter** στο ευρύτερο οικοσύστημα εργαλείων ανάπτυξης.

Συνοψίζοντας, η επιλογή μεταξύ **Notebook** και **Lab** θα πρέπει να καθοδηγείται από το είδος του έργου και τις προσωπικές προτιμήσεις/ανάγκες του χρήστη. Δεν πρόκειται τόσο για ανταγωνισμό, όσο για δύο όψεις του ίδιου νομίσματος. Όπως εύστοχα έχει επισημανθεί, το **Notebook** δίνει προτεραιότητα στην απλότητα, ενώ το **Lab** δίνει προτεραιότητα στην οργάνωση και επέκταση [19]. Και τα δύο υπηρετούν τον κοινό σκοπό της **διαδραστικής υπολογιστικής** και μπορούν να οδηγήσουν σε εξίσου ισχυρά αποτελέσματα, όταν χρησιμοποιηθούν σωστά.

## 3.10 Μελλοντικές εξελίξεις στο οικοσύστημα Jupyter

Κοιτώντας προς το μέλλον, το οικοσύστημα του Project Jupyter βρίσκεται σε συνεχή εξέλιξη, με αρκετές σημαντικές τάσεις και προοπτικές να διαμορφώνουν την πορεία του. Μία από τις πιο αξιοσημείωτες εξελίξεις είναι η ενοποίηση του κλασικού Notebook με το JupyterLab. Οι προγραμματιστές του Jupyter έχουν καταστήσει σαφές ότι το JupyterLab αποτελεί τη βάση για το μέλλον των notebooks – ήδη η επερχόμενη έκδοση Jupyter Notebook 7 είναι ουσιαστικά υλοποιημένη πάνω στην τεχνολογία του JupyterLab, λειτουργώντας ως μια στιβάδα συμβατότητας που προσφέρει την κλασική εμφάνιση αλλά χρησιμοποιεί το backend και τις επεκτάσεις του Lab [21]. Αυτό σημαίνει ότι μακροπρόθεσμα, η διάκριση μεταξύ Notebook και Lab ενδέχεται να γίνει δυσδιάκριτη ή και να εξαλειφθεί, με ένα ενοποιημένο περιβάλλον που θα συνδυάζει την απλότητα της κλασικής διεπαφής με τη δύναμη και επεκτασιμότητα του Lab. Ήδη από το 2023, με την κυκλοφορία του JupyterLab 4 και τη σταδιακή μετάβαση στο Notebook 7, έχει ανακοινωθεί ότι το Notebook 7 θα αντικαταστήσει πλήρως το κλασικό Notebook, διατηρώντας πλήρη συμβατότητα με τον υπάρχοντα φορμά των αρχείων και τις επεκτάσεις [12], [21].

Μια άλλη σημαντική εξέλιξη είναι η προσθήκη συνεργατικών δυνατοτήτων σε πραγματικό χρόνο (real-time collaboration). Σε μια εποχή όπου η απομακρυσμένη συνεργασία είναι συχνή, το Project Jupyter εργάζεται πάνω στη δυνατότητα πολλαπλοί χρήστες να επεξεργάζονται ταυτόχρονα το ίδιο notebook, βλέποντας ο ένας τις αλλαγές του άλλου όπως σε ένα Google Doc. Ήδη, το JupyterLab υποστηρίζει πειραματικά real-time collaboration μέσω ειδικής επέκτασης (με χρήση CRDT algorithms για συγχρονισμό). Από το 2023, νέες εκδόσεις του Notebook/Lab άρχισαν να ενσωματώνουν αυτή τη δυνατότητα [12]. Στο μέλλον, αναμένεται τα notebooks να γίνουν πλήρως συνεργατικά, διευκολύνοντας ομάδες επιστημόνων δεδομένων να εργάζονται ταυτόχρονα στον ίδιο κώδικα/ανάλυση χωρίς συγκρούσεις – μετατρέποντάς τα έτσι και σε εργαλείο συνεργατικής μάθησης και έρευνας [19].

Παράλληλα, το οικοσύστημα Jupyter διευρύνεται με νέα έργα που στοχεύουν σε ελαφρύτερη και ευρύτερη πρόσβαση στην τεχνολογία. Ένα χαρακτηριστικό παράδειγμα είναι το JupyterLite, μια διανομή του JupyterLab που τρέχει εξ ολοκλήρου μέσα στον browser, αξιοποιώντας WebAssembly και τη μηχανή Pyodide για την εκτέλεση του kernel χωρίς ανάγκη backend server [25]. Με το JupyterLite, μπορεί κανείς να ανοίξει ένα notebook σε μια απλή σελίδα web (π.χ. GitHub Pages) και να εκτελέσει Python κώδικα τοπικά στον browser. Αν και βρίσκεται ακόμη σε πρώιμο στάδιο, αυτή η προσέγγιση ανοίγει το δρόμο για εύκολη διάθεση notebooks ως στατικών ιστοσελίδων – π.χ. εκπαιδευτικό υλικό που ο αναγνώστης μπορεί να τρέξει άμεσα χωρίς εγκατάσταση λογισμικού. Στο μέλλον, με τη βελτίωση του WebAssembly και έργα όπως το Pyodide, το JupyterLite πιθανώς θα γίνει ακόμη πιο πρακτικό, διευρύνοντας την απήχηση του Jupyter σε σενάρια όπου η εγκατάσταση server δεν είναι εφικτή [25].

Επιπλέον, το Jupyter οικοσύστημα επεκτείνεται με θεματικές προσθήκες όπως το Jupyter AI, η εμβάθυνση στη διαχείριση μεγάλων δεδομένων (σύνδεση notebooks με clusters Apache Spark, Dask κ.ά.), αλλά και η βελτίωση της ποιότητας ζωής του προγραμματιστή. Αναπτύσσονται εργαλεία όπως το Pynblint, που καθοδηγούν σε καλύτερες πρακτικές, ενώ η υποστήριξη Git και οι μορφές diff για notebooks εξελίσσονται [13]. Η νέα μορφή Notebook

7 (βασισμένη στο JupyterLab) ενσωματώνει πολλές από τις επεκτάσεις του Lab και υπόσχεται καλύτερη διαχείριση πακέτων και περιβαλλόντων [21].

Τέλος, από κοινοτικής πλευράς, το Project Jupyter συνεχίζει να κερδίζει αναγνώριση και υποστήριξη. Έχει βραβευθεί, μεταξύ άλλων, με το ACM Software System Award για την επίδρασή του στην επιστήμη των υπολογιστών [22], [27], ενώ μεγάλες εταιρείες τεχνολογίας και οργανισμοί ανοιχτού λογισμικού συνεισφέρουν ενεργά στην ανάπτυξή του [19]. Η φιλοσοφία του παραμένει προσανατολισμένη στην ανοιχτή επιστήμη και την προσβασιμότητα, με στόχο να επιταχύνει τον κύκλο της ανακάλυψης μέσω εργαλείων που μπορούν να μοιραστούν και να επαναχρησιμοποιηθούν ανοιχτά [19], [22].

Συμπερασματικά, το μέλλον του Jupyter προδιαγράφεται δυναμικό: ενοποιημένες και πιο συνεργατικές πλατφόρμες, ευρύτερη πρόσβαση μέσω του web, βαθύτερη ενσωμάτωση σε όλα τα στάδια της επιστήμης δεδομένων. Για τον επιστήμονα δεδομένων, αυτό σημαίνει ακόμα πιο ομαλή εμπειρία – τα όρια μεταξύ σκέψης, προγραμματισμού, ανάλυσης και παρουσίασης θα μειώνονται συνεχώς. Το Jupyter έχει ήδη μεταμορφώσει τον τρόπο που εργαζόμαστε με δεδομένα, και με τις εξελίξεις αυτές αναμένεται να παραμείνει στον πυρήνα της επιστήμης δεδομένων, προσαρμοζόμενο στις νέες ανάγκες και τεχνολογίες, αλλά παραμένοντας πιστό στο όραμα της ανοιχτής, συνεργατικής και αναπαραγώγιμης έρευνας.

# Βιβλιογραφία


[1] Project Jupyter, “blog.jupyter.org,” Jul. 2015. [Online]. Available: https://blog.jupyter.org/project-jupyter-computational-narratives-as-the-engine-of-collaborative-data-science-2b5fb94c3c58

[2] S. Lau, “Instructor-Centered Design of Tools to Support Teaching Programming and Data Science At Scale,” 2023.

[3] researchgate, “researchgate.net,” [Online]. Available: https://www.researchgate.net/scientific-contributions/Brian-Granger-2039423532

[4] J. P. S. Moreno, “Software Engineering in the IoT context,” 2020.

[5] J. Tuloup, “github.com,” 2025. [Online]. Available: https://github.com/jupyterlab/jupyterlab

[6] ipython, “ipython.org,” [Online]. Available: https://ipython.org/#:~:text=IPython%20is%20a%20growing%20project%2C,a%20Python%20kernel%20for%20Jupyter

[7] T. Kluyver, B. Ragan-Kelley, and F. Perez, “Jupyter Notebooks - a publishing format for reproducible computational workflows,” in *Proc. 20th Int. Conf. on Electronic Publishing*, 2016.

[8] O. Valery, A. Stevens, and A. I. Tikhonov, “Jupyter Notebook, JupyterLab – Integrated Environment for STEM Education,” 2022.

[9] B. Granger, “researchgate.net,” [Online]. Available: https://www.researchgate.net/scientific-contributions/Brian-Granger-2039423532

[10] L. M. V. B. João Felipe Pimentel and J. F. João Felipe Pimentel, “A Large-Scale Study About Quality and Reproducibility of Jupyter Notebooks,” 2019.

[11] L. M. V. B. J. F. João Felipe Pimentel, “A Large-scale Study about Quality and Reproducibility of Jupyter Notebooks,” 2019.

[12] J. Weill, “blog.jupyter.org,” Jun. 2023. [Online]. Available: https://blog.jupyter.org/jupyterlab-4-0-is-here-388d05e03442

[13] L. Quaranta, F. Calefato, and F. Lanubile, “Pynblint: A quality assurance tool to improve the quality of Python Jupyter notebooks,” *SoftwareX*, vol. 28, art. 101959, 2024.

[14] C. R. Harris et al., “Array programming with NumPy,” *Nature*, vol. 585, pp. 357–362, 2020.

[15] The Pandas Development Team, “pandas-dev/pandas: Pandas,” *Zenodo*, Oct. 2020, doi: 10.5281/zenodo.3509134.

[16] W. McKinney, “Data Structures for Statistical Computing in Python,” in *Proc. 9th Python in Science Conf.* (SciPy 2010), Austin, TX, USA, 2010, pp. 51–56.

[17] J. D. Hunter, “Matplotlib: A 2D Graphics Environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[18] M. L. Waskom, “seaborn: statistical data visualization,” *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021.

[19] B. E. Granger and F. Pérez, “Jupyter: Thinking and Storytelling With Code and Data,” *Computing in Science & Engineering*, vol. 23, no. 2, pp. 7–14, 2021.

[20] D. Du et al., “Jupyter widgets and extensions for education and research in computational physics and chemistry,” *arXiv preprint*, arXiv:2401.06113, 2024.

[21] Project Jupyter, “JupyterLab Documentation — Get Started (JupyterLab 4),” Online documentation, 2023.

[22] M. Rule, A. Tabard, and J. D. Hollan, “Exploration and Explanation in Computational Notebooks,” *Proc. ACM Human-Computer Interaction (CSCW)*, vol. 2, article 162, 2018.

[23] Project Jupyter, “Jupyter Notebook 7 — Documentation,” Online documentation, 2023–2024.

[24] The Jupyter Development Team, “JupyterHub — Documentation,” Release 2.3.0, 2022.

[25] Project Jupyter, “NBViewer and JupyterLite: Open access to notebooks anywhere,” *Jupyter Blog*, Mar. 2022.

[26] L. N. Lamy et al., “Toward Notebooks that Learn to Tell Stories: An Architecture for Narrative Generation in Computational Notebooks,” in *Proc. Workshop on Human-Computer Interaction and Visualization (HCIV)*, IEEE VIS, 2021.

[27] N. C. Rule et al., “Ten Simple Rules for Reproducible Research in Jupyter Notebooks,” *PLOS Computational Biology*, vol. 15, no. 7, e1007007, 2019.

[28] O. K. A. Maayan, “Why Jupyter is Data Scientists’ Computational Notebook of Choice,” *Communications of the ACM*, vol. 63, no. 11, pp. 40–47, 2020.