Università
Ca'Foscari
Venezia

Corso di Laurea Magistrale

in Economia e Gestione
delle Aziende

Tesi di Laurea

# Jupyter Notebook

**Relatore**
Ch. Prof. Agostino Cortesi

**Laureando**
Nicola Howe
Matricola
867123

**Anno Accademico**
2021 / 2022

# Indice

# Introduzione

*Scientia potentia est.*

[Sapere è potere.]

(Thomas Hobbes, Leviatano, 1668)

La conoscenza nel corso dei secoli ha guadagnato via via un'importanza sempre maggiore, fino a diventare senza ombra di dubbio uno degli asset più importanti di ogni organizzazione. Per questo motivo la corretta raccolta, analisi ed interpretazione delle informazioni rappresenta una fase cruciale nel processo di creazione del valore di ciascuna azienda, senza la quale non è possibile identificare ed attuare alcun tipo di decisione strategica. Questo risulterà veritiero indipendentemente dalle dimensioni dell'ambiente in cui è inserita l'azienda. Sarà, infatti, egualmente applicabile alla piccola impresa che lavora in un contesto territoriale locale, come alla grande multinazionale con sedi e filiali in cento paesi. Quando un'azienda non ha accesso o non è in grado di elaborare le informazioni ed i feedback derivanti dai suoi contatti con l'esterno, questa si troverà impossibilitata a comprendere l'ambiente stesso in cui opera.

Questo bisogno ha subito una significativa accelerazione dal quindicesimo secolo ad oggi. Con la Scoperta dell'America molti stati europei si sono lanciati in progetti esplorativi e di colonizzazione, via via normalizzando i viaggi oltreoceano e connettendo gradualmente tutti i paesi del mondo. La Rivoluzione Industriale ha donato all'umanità, tra le altre cose, il trasporto su rotaia ed il telegrafo, strumenti essenziali per coprire grandi distanze, sorpassati in importanza solamente dal telefono nel diciannovesimo secolo, e dall'aeroplano e da internet nel ventesimo secolo. Grazie a tali invenzioni viaggiare e connettersi con persone distanti è diventato sempre più semplice e meno costoso, fino a diventare quasi alla portata di tutti.

Al di là delle implicazioni politiche e sociali, è interessante analizzare cosa ha comportato questo processo da una prospettiva aziendale. Per fare un esempio, al

giorno d'oggi indipendentemente dal paese in cui ci si trova, salvo rare eccezioni, ed avendo a disposizione una connessione internet, è possibile dare inizio ad un progetto su *Kickstarter* e ricevere il denaro necessario ad aprire un'impresa capace di lanciare il suo prodotto sul mercato globale. Gli "unici" requisiti necessari sono una sufficiente motivazione e determinazione, buone capacità di marketing ed un progetto accattivante. Tutto ciò, come facilmente intuibile, era inimmaginabile fino a cinquant'anni fa e testimonia il grado di interconnessione raggiunto e le potenziali opportunità.

Oltre a poter raggiungere posti lontani in breve tempo e poter parlare/vedere una persona a migliaia di chilometri di distanza, la tecnologia ha aumentato anche il grado di tracciabilità degli individui. Nello specifico è accresciuta la quantità di informazioni ricavabili, grazie a sensori e feedback, riguardo a spostamenti, preferenze e desideri. Le persone stesse condividono molte di queste informazioni su piattaforme pubbliche quali Facebook, Twitter, Instagram e Tik-Tok (Lohr, 2012).

La mole di dati ricavabili è aumentata a tal punto, rispetto al passato, che la loro analisi è divenuta una sfida logistica e temporale contro la loro obsolescenza. Conosciuti anche come *Big Data*, questi enormi volumi di dati grezzi, riguardano innumerevoli aspetti e categorie diverse, e vengono raccolti con velocità crescente (Segal, 2022).

Una volta ricavato o acquistato un insieme di dati, le compagnie usano un processo chiamato *data mining* per ricavare informazioni utilizzabili. Ciò avviene tramite l'impiego di software particolari che vanno ad identificare ripetizioni e schemi, chiamati *patterns*, all'interno dei dati grezzi. La qualità di questo processo dipende dall'efficacia del ricavo dei dati, dalla loro conservazione e dalla capacità di elaborazione disponibile.

Le informazioni ricavate potranno essere successivamente impiegate per la creazione di strategie di marketing più efficaci e modellate sui gusti e sulle preferenze del target (Twin, 2021).

Il programma open-source Jupyter Notebook permette di facilitare questo processo. Il progetto, iniziato nel 2014 come evoluzione di IPython Project, ha come scopo il supporto e l'integrazione di calcolo scientifico e *data science* (PJ). Nello specifico,

permette la creazione e condivisione di documenti contenenti stringe di codice sorgente, risultati computazionali, equazioni, schemi, diagrammi e altre risorse multimediali, insieme alla parte testuale. Si tratta quindi di uno strumento decisamente versatile in grado di fungere sia da ambiente di sviluppo del materiale, dei dati, oggetto di studio, che da piattaforma di presentazione ed esposizione.

Il programma può essere utilizzato sia da cloud che da locale e consiste di due componenti principali; le *celle*, di forma rettangolare, svolgono da interfaccia principale in cui è possibile inserire del testo o del codice, e dove verrà visualizzato il risultato, ed i *kernels* che, invisibili all'utente, si occupano della parte di computazione ed elaborazione. Per quanto riguarda la flessibilità in termini di programmazione, oltre ai maggiori, in termine di diffusione e popolarità, come Python, R, Julia, e Scala, Jypyter Notebook supporta un totale di più di 40 linguaggi.

In conclusione, Jupyter Notebook è un ambiente di sviluppo, in cui è possibile sperimentare con il codice, modificarlo e ripetere, come in un dialogo tra l'utente e i dati su cui sta lavorando (OCDS, 2020).

L'obiettivo di questo scritto è lo studio del programma Jupyter Notebook attraverso l'analisi del suo funzionamento, dal processo d'installazione alle dinamiche su cui si basa il software. Si andranno a studiare le sue caratteristiche, facendo una valutazione dei punti di forza e dei punti di debolezza, soprattutto in ottica applicativa, affrontando le varie tematiche in una prospettiva economico-manageriale; Infine, si affronterà come questo software possa integrarsi nel tessuto aziendale, prendendo in considerazione un caso reale.

La metodologia utilizzata per la ricerca delle informazioni e la stesura di questo documento prevede due step. Inizialmente vi è una fase di raccolta del materiale tecnico e della letteratura scientifica. Successivamente si procede all'analisi e all'elaborazione delle informazioni mantenendo la prospettiva dell'utente finale.

Questo documento vuole rappresentare quanto più vicino possibile ad un manuale, ad una guida per un utente con background diverso dall'informatica. Gli scopi ultimi sono la diffusione della conoscenza di questo software, in particolare delle sue potenzialità e criticità, nonché agevolarne l'utilizzo da parte di un aspirante utente.

La struttura adottata prevede quattro capitoli oltre all'introduzione.

Nel capitolo iniziale, oltre alla procedura di installazione, vengono descritti nel dettaglio sia gli elementi grafici che vanno a comporre la schermata principale di Jupyter nonché quelli che formano lo strumento principe, il Notebook.

Il secondo capitolo affronta i punti di forza e di debolezza di Jupyter, e di come questi ultimi possono essere attenuati o eliminati tramite l'implementazione di estensioni al software originale.

Il terzo capitolo analizza un caso reale di utilizzo di Jupyter Notebook da parte di un'azienda, e come l'impatto di questa scelta sia stato significativo.

Il capitolo conclusivo tira le somme del percorso affrontato, sintetizzando ed evidenziando i passaggi che sono risultati più interessanti e offrendo uno spunto riguardo gli argomenti che sarebbe utile approfondire in ricerche successive.

# Capitolo 1. Funzionamento

## 1.1 Avvio

Come già accennato, è possibile usufruire di Jupyter Notebook sia in cloud sia in locale, a seconda delle preferenze dell'utente e della solidità dell'hardware a disposizione.

Per utilizzare il software tramite cloud Jupyter Project ha ideato JupyterHub, una piattaforma contenitore in cui gli utenti hanno a disposizione sia ambienti computazionali che risorse, senza oneri. Su JupyterHub è possibile riservare ed utilizzare un ambiente di lavoro personale ed accedere a risorse condivise con il resto del team di lavoro/gruppo di studio, gestite dagli amministratori di sistema. Questa soluzione è personalizzabile e scalabile, rendendola adatta sia a gruppi di lavoro, indipendentemente dalle dimensioni, sia a corsi accademici; gli unici requisiti necessari sono una connessione ad Internet ed un sistema operativo Linux/Unix, quest'ultimo punto è aggirabile tramite l'utilizzo, ad esempio, di una Linux VM (PJ, Hub).

Per quanto riguarda l'utilizzo in locale, Jupyter Notebook può èssere installato tramite diversi canali, a seconda delle esigenze/preferenze dell'utente, l'unico requisito è aver preventivamente installato anche Python (Python 3.3 o maggiore, oppure Python 2.7).

Il canale consigliato ai nuovi utenti, e ad utenti inesperti, è tramite Anaconda. Anaconda è una piattaforma per data science che, oltre ad installare Python, include al suo interno una vasta libreria di *pacchetti* contenenti software e strumenti utili, tra cui Jupyter Notebook. Uno dei *pacchetti* è Conda, un sistema open-source di gestione degli *ambienti* e dei *pacchetti*; questo consente all'utente di "muoversi" all'interno di questo universo, creando, cambiando, salvando e caricando tra gli *ambienti* a disposizione. Conda, inoltre, permette di cercare, trovare ed installare nuovi *pacchetti*, anche se questi necessitano di versioni di Python o di linguaggi differenti, creando un *ambiente* apposito con le specifiche richieste.

In alternativa, per gli utenti Python esperti che lo desiderassero, è possibile installare Jupyter Notebook tramite il gestore Python di *pacchetti*, chiamato anche *pip* (Pip Installs Packages). Quest'ultimo è specializzato in *pacchetti* che utilizzano il linguaggio Python, installandoli in qualunque ambiente. A differenza di pip, Conda non fa

distinzioni in base al linguaggio ed inserisce tutti i pacchetti in ambienti Conda (PJ,

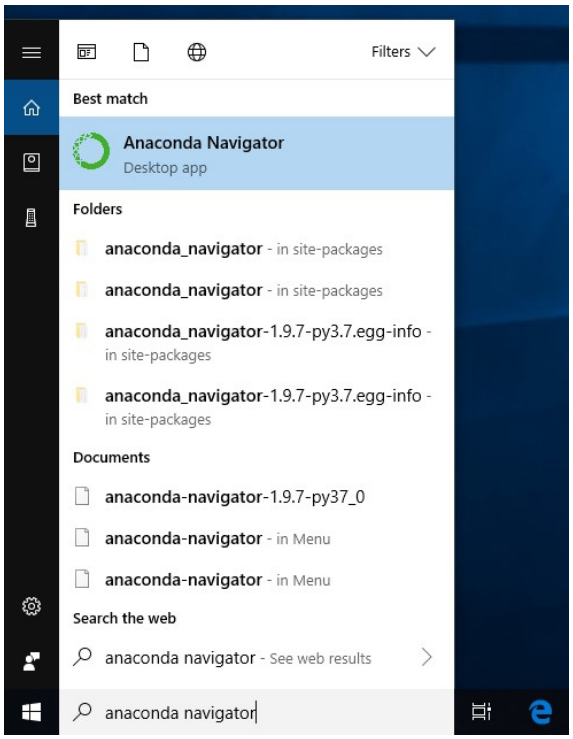



Fig. 1 – Anaconda Navigator, fonte *Anaconda Documents*

Install). Una volta installato Anaconda sarà possibile seguire un piccolo tutorial che avvierà Anaconda Navigator ed illustrerà come accedere a due dei software principali della libreria: Spyder, un ambiente di sviluppo integrato specializzato nel linguaggio Python, e Jupyter Notebook.

Successivamente sarà possibile avviare Anaconda Navigator tramite il percorso Start di Windows, come illustrato nella Figura 1.

Tramite questa piattaforma sarà possibile visionare tutti i programmi inclusi nella libreria.

Qual ora un programma non risulti installato sarà possibile scaricarlo tramite il pulsante *Install*, Anaconda Navigator lo aggiungerà quindi alla lista dei programmi eseguibili.

Se il programma, invece è già presente basterà avviarlo tramite un click sul pulsante *Launch.*

In alternativa, è possibile avviare Jupyter Notebook tramite Anaconda Prompt, come indicato dalla Figura 2.

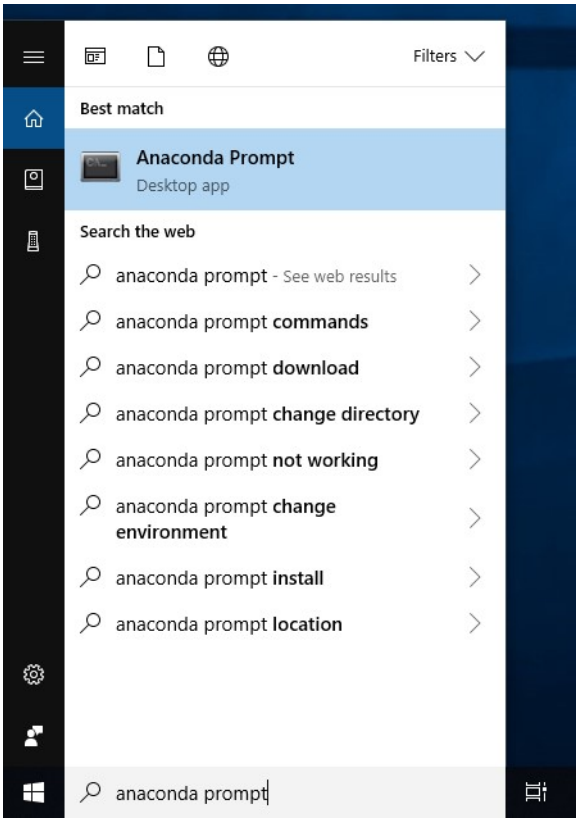



Fig. 2 – Anaconda Prompt, fonte *Anaconda Documents*

Una volta aperto Anaconda Prompt, sarà sufficiente digitare il comando "*jupyter-notebook*" e premere il tasto *Invio* per avviare il programma (Anaconda, Tutorial).

## 1.2 Dashboard

Indipendentemente dal percorso selezionato per avviare Jupyter Notebook, la schermata che si aprirà su browser sarà la medesima.


Fig. 3 – Notebook Dashboard, fonte *Jupyter Notebook*

Come visibile dalla Figura 3, la schermata Notebook Dashboard funge da luogo di controllo e gestione, oltre che da libreria, sia per i kernels, che qui è possibile attivare e disattivare, sia per tutti i "documenti" e le cartelle create dall'utente. Da questa schermata si potranno inoltre creare nuovi "documenti," propriamente detti *Notebook(s)*, tramite il pulsante *New* in alto a sinistra, selezionando il linguaggio di programmazione desiderato dal menù a tendina che si aprirà. Tramite lo stesso pulsante sarà inoltre possibile creare file di testo e cartelle. Tramite *Upload*, invece, si potrà importare nella libreria file di tipo *.ipynb* o *.py* (PJ, Intro).

Qualora di desideri disattivare un kernel, sarà sufficiente passare dalla schermata *Files* alla schermata *Running*, cliccando sul corrispettivo in alto a sinistra. Fatto ciò sarà sufficiente cliccare sul pulsante *Shutdown* del Notebook desiderato per disattivarne il kernel. Questo rimarrà inattivo fino al successivo utilizzo del Notebook, quando si attiverà automaticamente non appena si effettuerà l'accesso al documento.

Notebook Dashboard consente di ordinare le cartelle ed i file a disposizione in ordine alfabetico, di ultima modifica, e per dimensioni, tramite gli appositi pulsanti in alto a

destra, appena al di sotto di *New* e *Upload*.

### 1.3 Notebook

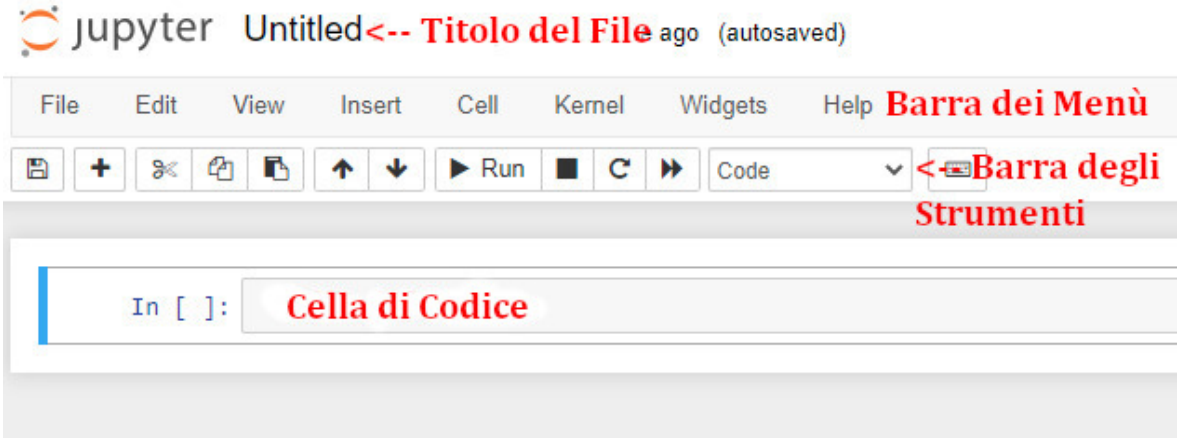



Fig. 4 – Notebook User Interface, fonte *The Jupyter Notebook - Read the Docs*

Per quanto riguarda le celle, invece, è importante fare una breve premessa. Ogni Notebook è composto da una sequenza di celle. Queste ultime, definite come campi di imput, possono essere di tre tipologie: celle di codice, celle di annotazione e celle crude; una volta eseguite si comporteranno in modo differente a seconda della loro tipizzazione. Ogni cella, alla sua creazione, viene tipizzata come cella di codice ("Code") ma può essere convertita in un altro tipo tramite un menù a tendina nella Barra degli Strumenti.

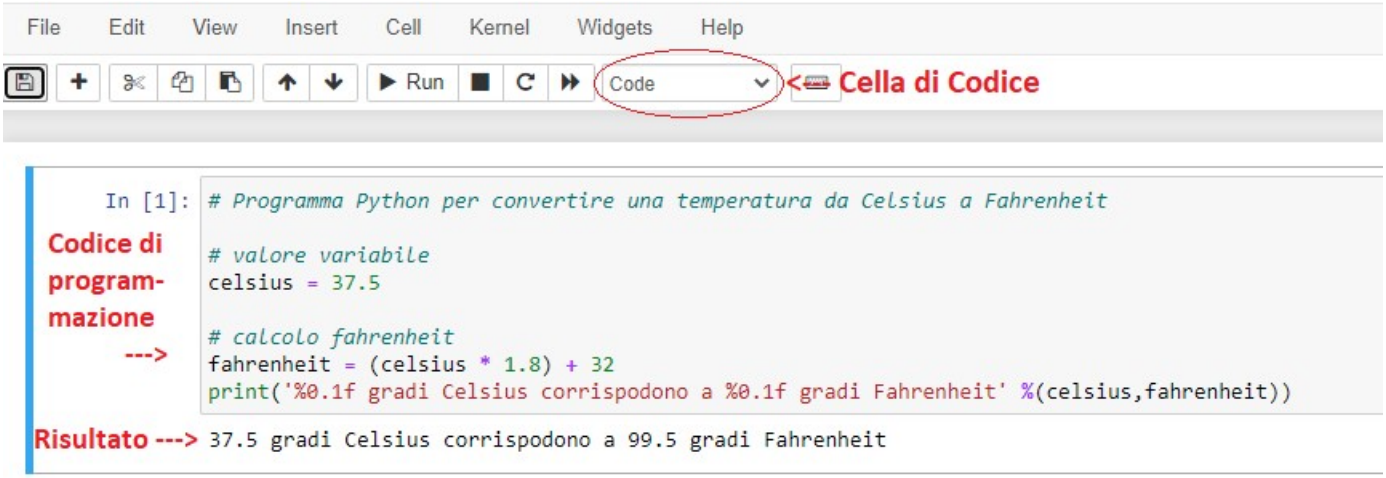



Fig. 5 – User Interface Cella di Codice, fonte *Jupyter Notebook*

Nelle celle di codice, Figura 5, è consentito inserire del codice sorgente da modificare o scriverne del nuovo da zero, utilizzando il linguaggio di programmazione consentito dal

kernel utilizzato. Quando una cella così tipizzata viene eseguita, il suo contenuto viene spedito al kernel ad essa associato dove viene elaborato. Il codice, una volta elaborato, viene mostrato nel Notebook come il risultato della cella.

Le celle di annotazione, Figura 6, consentono di documentare e commentare il processo descritto nel Notebook, alternando la parte di codice computativa con quella di testo descrittiva. Selezionando il kernel di IPython, ad esempio, è possibile utilizzare il linguaggio Markdown per aggiungere complessità e personalizzare il testo. Sarà possibile, infatti, enfatizzare tramite il corsivo, sottolineare, evidenziare con il grassetto,


Fig. 6 – User Interface Cella di Annotazione, fonte *Jupyter Notebook*

formare liste, e così via.

Sempre tramite le celle di annotazione è possibile includere formule e calcoli matematici utilizzando la notazione di LaTex standard. Questo grazie alle funzionalità aggiunte da MathJax, un motore JavaScript per l'elaborazione e l'esposizione/esibizione di formule matematiche, sia in modalità *Inline,* che *Display,* Figure 7 e 8.

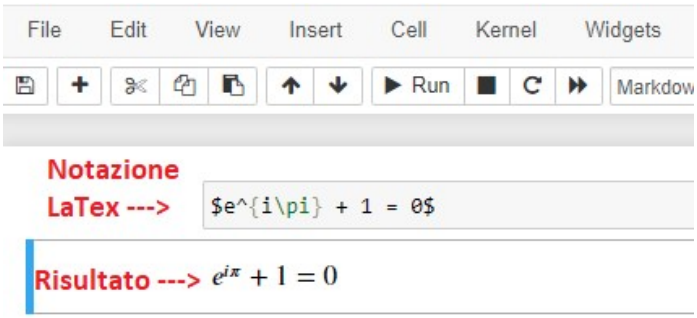



Fig. 7 – Equazione Inline, fonte *Jupyter Notebook*

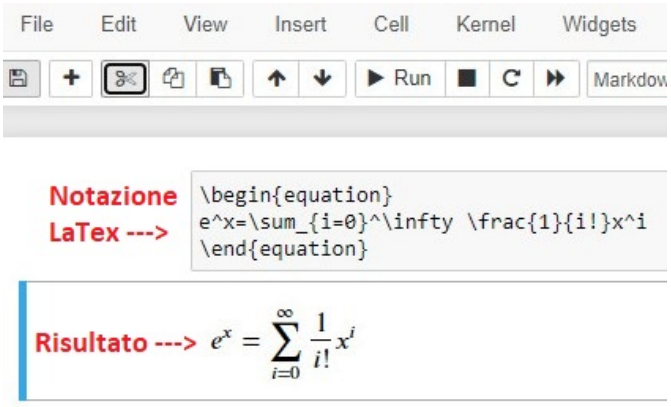



Fig. 8 – Equazione Display, fonte *Jupyter Notebook*


Fig. 9 – Intestazione, fonte *Jupyter Notebook*

La suddivisione in paragrafi consente di incrementare la complessità strutturale del documento utilizzando le intestazioni, Figura 9. I titoli nelle intestazioni sono sempre preceduti dal simbolo # seguito da spazio e dal titolo desiderato; il numero di # varia da 1 a 6 per determinarne le dimensioni, dove un # andrà a generare un'intestazione delle dimensioni massime.

Infine le celle crude, Figura 10, offrono uno spazio di scrittura che non sarà considerato ed elaborato dal Notebook, dove quindi è possibile indicare direttamente il risultato della cella (PJ, Intro).

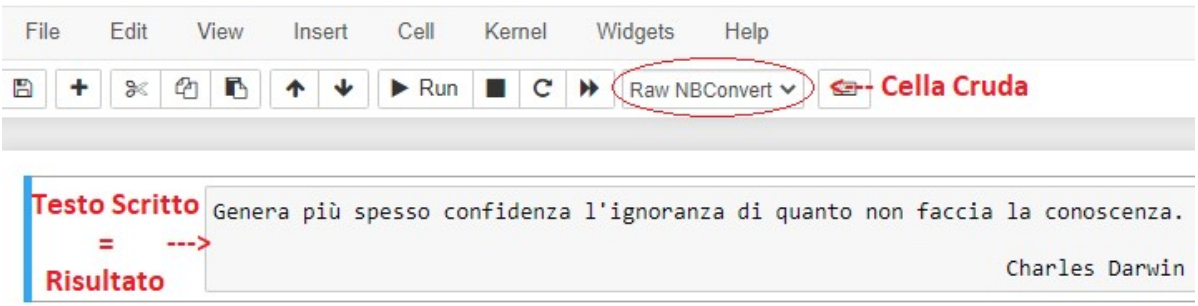



Fig. 10 – User Interface Cella Cruda, fonte *Jupyter Notebook*

## 1.4 Kernels

Come accennato in precedenza, i kernels sono processi indipendenti eseguiti in background, ognuno specifico/specializzato per un linguaggio di programmazione, che interagiscono con la User Interface, elaborando, computando e restituendo l'output finale alle celle, Figura 11.

Di default all'installazione di Jupyter Notebook viene anche installato *IPython*, il kernel specifico per il linguaggio Python. Detto ciò, sono disponibili centinaia dei kernels da scaricare, ciascuno specializzato in un differente linguaggio di programmazione.

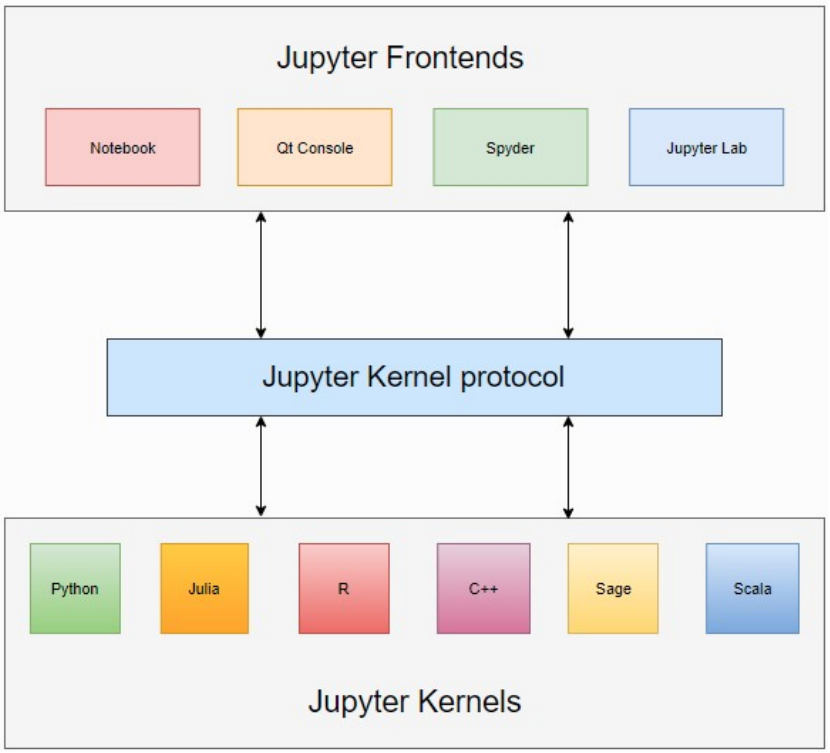



Fig. 11 – Kernels, fonte *Xeus* (Xeus, Kernels)

Per installare nuovi kernels è necessario prima creare nuovi *ambienti*, chiamati environments, per evitare possibili conflitti. In questi sarà possibile immagazzinare le librerie e le dipendenze collegate al kernel di riferimento e necessarie al suo funzionamento. Per creare un nuovo environment è sufficiente avviare Anaconda Prompt e digitare:

a) conda create -n *nomeenvironment*

una volta fato premere invio. Allo step successivo si dovrà attivare l'environment tramite:

b) conda activate *nomeenvironment*

anche qui seguito da invio. Una volta completato si potrà pocedere con l'installazione della libreria di riferimento:

c) conda install *nomelibreria*

premere nuovamente invio e seguire la procedura a schermo. Gli ultimi due step riguardano l'installazione del kernel e la sua aggiunta a Jupyter Notebook, rispettivamente:

d) conda install *nomekernel*

e

e) *linguaggio nomekernel* install --user --name= *nomeenvironment*

questo farà si che il nuovo kernel sia aggiunto alla lista di quelli disponibili quando si desidererà creare un nuovo Notebook. Ad ogni modo, esistono innumerevoli guide online che illustrano nel dettaglio i comandi richiesti per ogni kernel specifico, a seconda delle necessità.

Jupyter Notebook prevede anche che un utente desideri creare un proprio kernel customizzato. Per questo motivo mette a disposizione Xeus, una libreria ideata per facilitare l'implementazione di kernel custom; occupandosi della parte riguardante il protocollo di comunicazione tra Jupyter ed il kernel, Xeus facilita gli utenti sviluppatori permettendogli di concentrarsi esclusivamente sull'implementazione della parte interpretativa del kernel (PJ, Kernels).

# Capitolo 2. Prospettiva Implementativa

## 2.1 Potenzialità

Jupyter Notebook presenta diversi vantaggi, alcuni dei quali sono già stati accennati in precedenza, che ora andremo ad analizzare più nel dettaglio.

La *flessibilità* rappresenta uno dei maggiori punti di forza di questo software, non solamente in termini di linguaggi di programmazione utilizzabili, ma anche grazie alla disponibilità della scelta tra locale e cloud, a seconda delle necessità/preferenze dell'azienda. Qual ora una società lo prediliga è possibile crearsi un'infrastruttura locale dove allocare i server contenenti Jupyter e database, avendo così un maggior controllo per quanto riguarda le informazioni sensibili. In alternativa si può scegliere di puntare su una maggiore connettività ed adottare un sistema cloud, scaricando così molti dei rischi e costi collegati al mantenimento dei servers (Sjursen, 2020).

Un secondo aspetto importante della *flessibilità* è la *capacità d'integrazione* con altri software. Tutti i file prodotti con Jupyter Notebook sono in formato JSON, uno dei formati più diffusi e facilmente convertibili in altri formati qualora necessario. Oltre a ciò, il software prevede uno strumento chiamato *nbconvert* che permette l'utente di convertire i suoi documenti in svariati formati. Per fare qualche esempio, se l'utente avesse la necessità di condividere dei contenuti via web, sarebbe possibile convertire il/i documento/i in HTML; se fosse richiesto presentare il lavoro in formati più familiari, *nbconverter* consentirebbe la conversione in PDF, oppure in LaTeX se si volesse pubblicare, ed altri ancora (PJ, nbconverter).

In tema di *sicurezza*, per sua natura Jupyter prevede possibilità di implementare diversi modi e layers per proteggere il lavoro svolto ed i dati sensibili; da una semplice password ad una hashing password, fino ad un sistema di criptaggio TLS (Transport Layer Security) (PJ, Server).

Un altro vantaggio offerto da Jupyter è la grande capacità di *divulgazione* del materiale contenuto in un Notebook, ovvero nell'approccio che questo strumento permette di adottare per esibire il materiale oggetto di studio. Come già menzionato è possibile vedere al tempo stesso il codice ed il risultato, attivando tutte le celle insieme per una visione d'insieme oppure una per volta per un'analisi più dettagliata. Questa

funzionalità, come intuibile, costituisce un potenziale significativo, specialmente in ottica di *knowledge sharing*, consentendo sia di sperimentare con il codice e prendere immediatamente nota degli effetti, sia di accedere alla sorgente da cui si raccolgono i dati direttamente dal documento con cui si presenta uno studio/un'idea (BM, 2021). Grazie alla relativa *semplicità di utilizzo* e *customizzazione*, qualora si dovesse rielaborare il lavoro svolto da altri, la struttura a celle indipendenti facilita di molto la comprensione, velocizzando le tempistiche ed ottimizzando lo sforzo richiesto. Questa semplicità di utilizzo risulta molto importante qualora l'utente, pur non avendo compiuto studi di programmazione, necessiti di compiere delle elaborazioni su dei dati per poi presentarli al team.

Grazie a queste caratteristiche Jupyter Notebook è uno strumento eccellente per la comunicazione, sia che venga utilizzato per creare tutorials, che presentazioni o spiegazioni. Il grado di personalizzazione consente all'utente di nascondere la parte di codice, illustrare i concetti tramite grafici, includerci del materiale video e rendere interattiva presentazione (Grootendorst, 2021).

## 2.2 Limitazioni

La capacità divulgativa e la semplicità di utilizzo, pur apportando un valore importante al prodotto finale, rappresentano anche un potenziale punto debole di Jupyter Notebook.

La struttura a celle indipendenti, nei grandi volumi di materiale, può causare non poco caos e conseguente difficoltà nel tenere traccia del codice. Come insegnano gli ingegneri del software a proposito della scrittura del codice, per evitare queste problematiche è consigliata la strutturazione in funzioni, classe, ed oggetti. Questo approccio previene anche le varie problematiche collegate alla duplicazione accidentale del codice, evento discretamente più probabile, o se non altro più arduo da prevenire, in una struttura a celle indipendenti (BM, 2021).

Una seconda criticità è data dalla mancanza di un sistema standard di correzione del codice, o dello stile del codice. La spiegazione nella natura di Jupyter, nato principalmente come strumento per il *data science*, non per la pura programmazione. Nondimeno, può rappresentare un ostacolo quando, a seguito di sperimentazioni

effettuate sul codice in un Notebook, si prosegue alla fase di test su una piattaforma di programmazione più specifica. Qui il codice scritto su Notebook potrebbe essere recepito come caotico e/o disorganizzato oppure direttamente incorretto.

Infine, è importante affrontare il tema della *diffusione di dati sensibili/strategici* impiegati, chiamati anche *underlying data*. Il problema, naturalmente non si pone se il Notebook contenente dati sensibili viene condiviso solo con il team, ma qualora questo venisse impiegato per la divulgazione ad un pubblico più ampio, si pensi ad una riunione interdivisionale, con il cliente, o ad una conferenza, ecco che verrebbe a crearsi un problema per l'azienda, poiché consentirebbe l'accesso diretto ai dati ad un numero di utenti rischioso. La diffusione di dati sensibili rappresenta un problema della massima serietà e potenzialmente molto rischioso per l'azienda. Ecco che in osservanza del Regolamento Generale sulla Protezione dei Dati, in sigla RGPD, questa deve fare tutto il possibile per evitarne la diffusione, mettendo in pratica i comportamenti necessari (Sjursen, 2020).

## 2.3 Estensioni

Come già spiegato, Jupyter Notebook è altamente *customizzabile* e permette l'implementazione di *estensioni* per arricchire l'esperienza dell'utente. Queste estensioni sono semplici add-on che consentono di estendere le funzionalità a disposizione, in base alle necessità specifiche (Koehrsen, 2018). Un esempio è il già precedentemente menzionato *nbconvert*, il quale consente la conversione del Notebook in innumerevoli formati, tra cui JSON, HTML, PDF e LaTeX.

Il primo step da seguire per poter beneficiare delle estensioni consiste nell'installazione di *nbextensions*, una libreria che permette la configurazione delle estensioni desiderate. Una volta avviato Anaconda Prompt, digitare il comando:

*conda install jupyter_contrib_nbextensions && jupyter contrib nbextension install*

questo avvierà l'installazione automatica. Se tutto andrà a buon fine *nbextensions* sarà visibile come tab nella Notebook Dashboard, Figura 12.

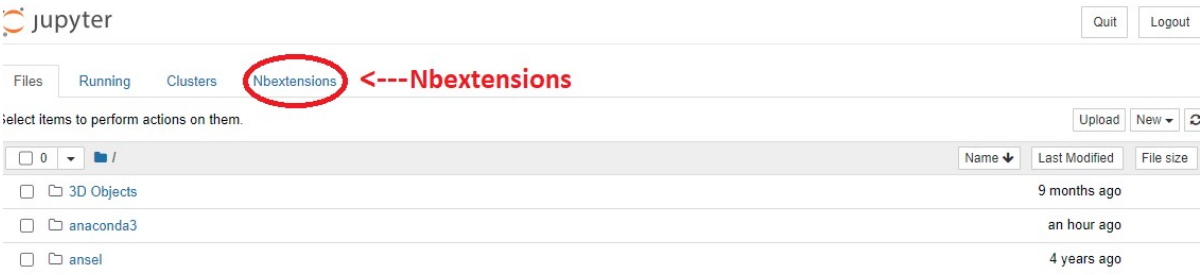



Fig. 12 – Nbextensions, fonte *Jupyter Notebook*

Una volta cliccatoci, verrà visualizzata la lista delle estensioni disponibili. Sarà sufficiente flaggare il corrispettivo checkbox per implementare una applicazione, Figura 13. Selezionando una estensione, invece, verrà mostrata in basso una breve guida completa di esempi, oltre alla possibilità di customizzare l'implementazione, Figura 14.

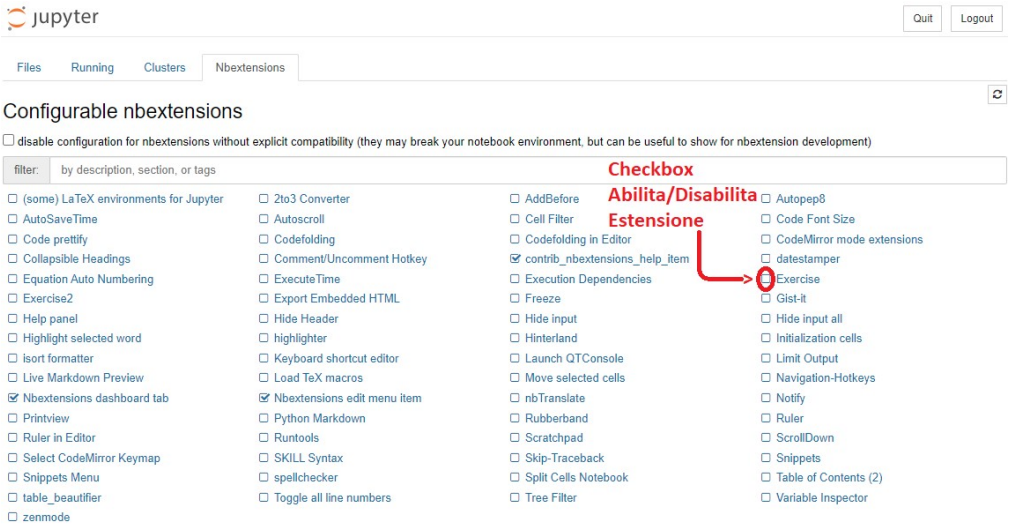



Fig. 13 – Extensions Tab, fonte *Jupyter Notebook*


Fig. 14 – Extensions Tab Description, fonte *Jupyter Notebook*

Analizzando più nel dettaglio le varie estensioni a disposizione, sempre tenendo a mente le limitazioni di Jupyter Notebook, è possibile trovare alcune soluzioni che ne migliorano l'esperienza e mitigano alcuni dei punti di debolezza del software.

Un esempio calzante è *Table of Contents*, l'estensione che aggiunge un indice interattivo, posizionabile a piacere nella schermata di lavoro, Figura 15.

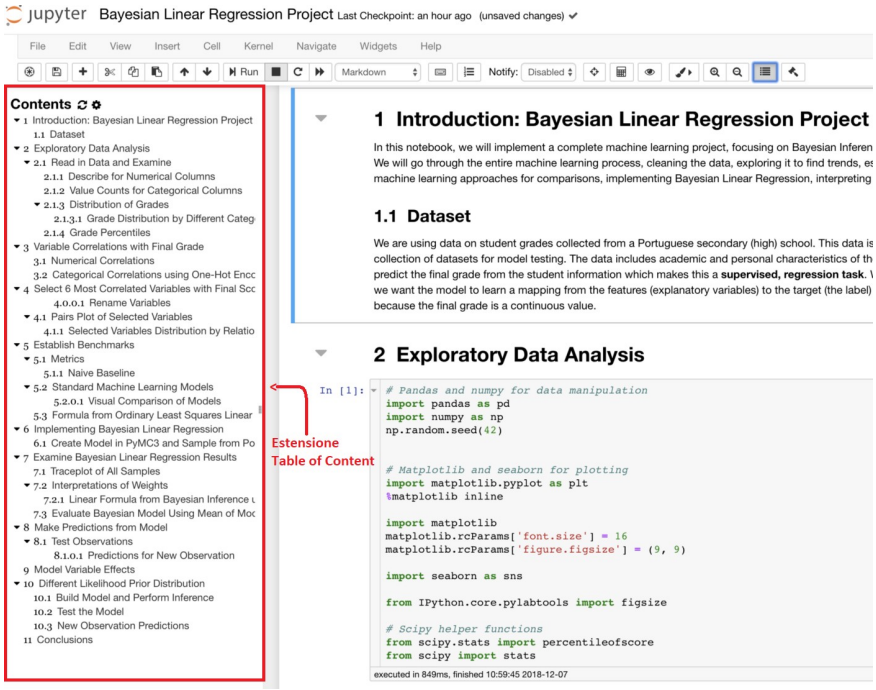



Fig. 15 – Estensione Table of Content, fonte *Koerhsen 2018*

L'estensione, inoltre, rende possibile inserire un indice interno al documento che oltre ad essere cliccabile e rendere il documento discretamente più navigabile, indicherà anche le celle selezionate e quelle in esecuzione. *Table of Contents* consente di mitigare quelli svantaggi di Jupyter che vengono evidenziati nelle situazioni in cui è necessario gestire grandi volumi di materiali. Se la struttura a celle indipendenti, in una situazione simile, può risultare eccessivamente dispersiva, grazie a questa estensione è possibile organizzare il materiale ed accedervi con più semplicità.

Per quanto riguarda la mancanza di un sistema standard di correzione del codice, o dello stile del codice, invece, esistono diverse estensioni la cui implementazione può venire in aiuto. Per quanto riguarda l'estensione Autopep8, è linguaggio-specifica per Python e consente sia di mantenere uno stile di scrittura consistente nel tempo, che di aderire allo standard di scrittura PEP 8. L'estensione *Code Prettify*, invece, è un formattatore di codice non-specifico per un solo linguaggio, ma customizzabile ed adattabile alle preferenze di programmazione dell'utente. Sarà necessario, dunque,

scaricare le librerie richieste per il linguaggio utilizzato, seguendo le indicazioni che appariranno come in Figura 14, una volta selezionato *Code Prettify*. Come visibile dal confronto delle Figure 16-17, l'estensione *Code Prettify* rende il codice più leggibile, facilitandone quindi la comprensione, utile sia per l'individuazione di eventuali errori, sia per una successiva modifica temporalmente distante dalla data di prima creazione, sia per soddisfare una necessità di tipo educativo o esplicativo.

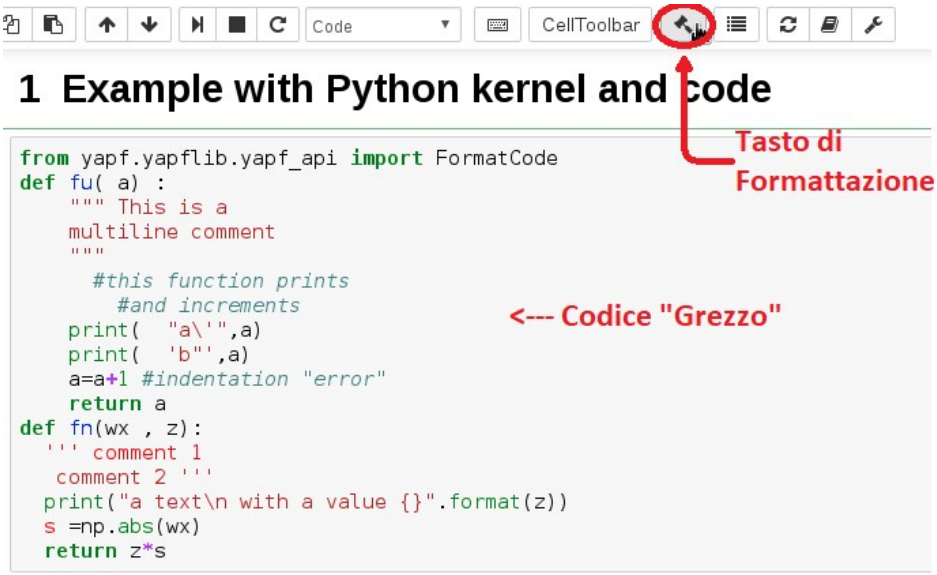
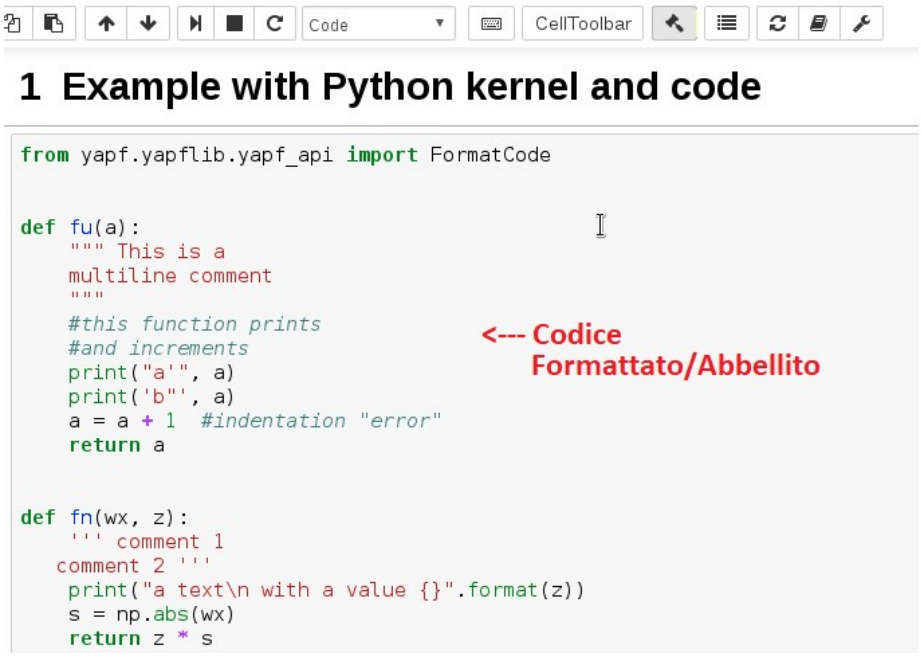



Fig. 16 – Estensione Code Prettify - prima della formattazione, fonte *Jupyter Notebook*

Fig. 17 – Estensione Code Prettify - dopo la formattazione, fonte *Jupyter Notebook*




Sempre allo scopo di rendere la parte di scrittura del codice più pratica e funzionale, l'estensione *Variable Inspector* consente di avere sempre a portata di click, tramite un pulsante nella barra degli strumenti, una lista aggiornata delle variabili create all'interno dell'intero documento, Figura 18. Questo evita di dover controllare ogni

cella manualmente alla ricerca della variabile di cui si ha bisogno, oppure per verificare di non nominarne una nuova con lo stesso nome dato ad un'altra e così via. Basti pensare che la convenienza di questa estensione e l'efficientamento del tempo che da essa deriva aumenta in maniera direttamente proporzionale al volume di dati e codice che vengono inseriti in un Notebook.

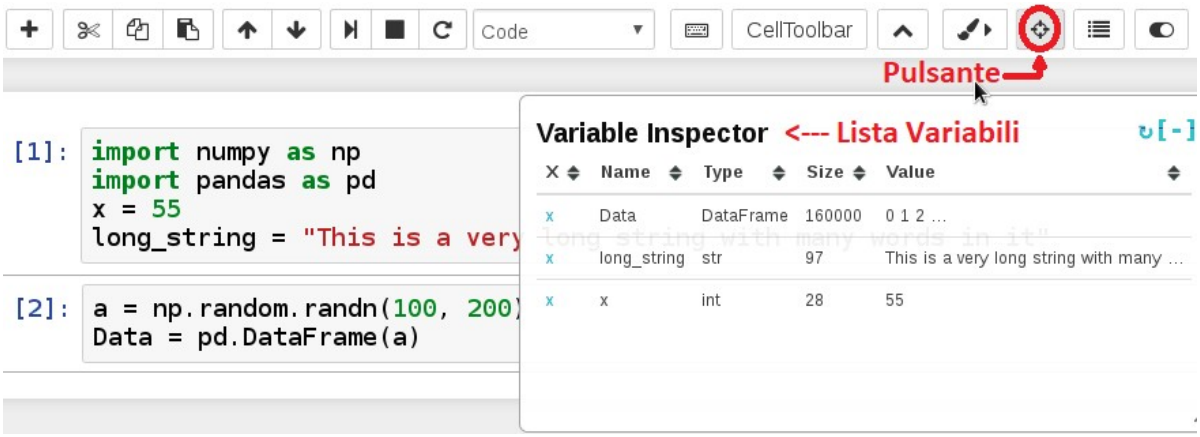



Fig. 18 – Estensione Variable Inspector, fonte *Jupyter Notebook*

Un fattore spesso sottovalutato finché non diviene un problema è la velocità di esecuzione. Mantenere il controllo sulle performance e sull'efficienza del codice che viene sviluppato è una questione vitale nel lungo periodo. Basti immaginare ad una soluzione per estrarre dieci dati da una fonte, la quale impiega tre secondi per completare l'operazione. Ora si pensi alla stessa soluzione, ma questa volta i dati da estrarre sono tre miliardi e provengono da fonti distinte. Quanto tempo verrà impiegato per il completamento? Che costo comporta questa attesa per l'azienda, e ogni quanto è necessario sostenerla? Una questione simile è facilmente gestibile tramite l'estensione *Execute Time*, la quale registra e mostra all'utente quando ogni cella è stata eseguita e quanto tempo ha impiegato, Figura 19.

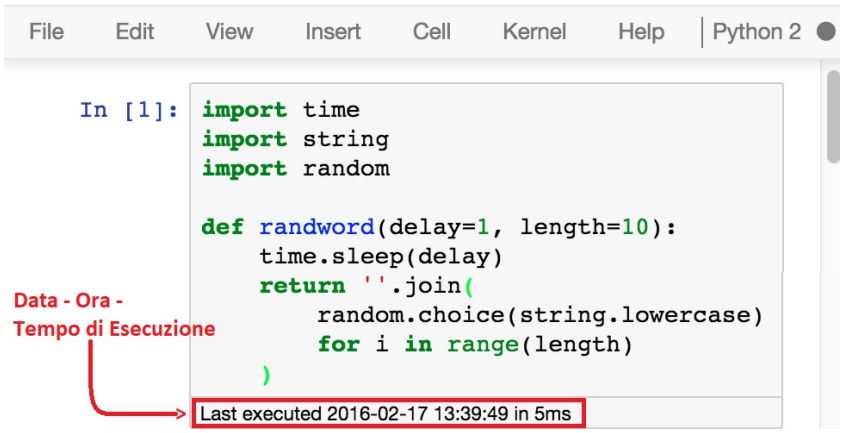



Fig. 19 – Estensione Execute Time, fonte *Jupyter Notebook*

La quinta estensione in esame riguarda l'argomento della diffusione di dati sensibili e rappresenta una possibile soluzione per mitigarne il rischio. *Hide input all* consente, con un solo click di "*nascondere*" tutta la parte di codice delle celle, ovvero tutti gli inputs, lasciando visibile solamente il risultato finale, gli outputs, Figure 20-21.

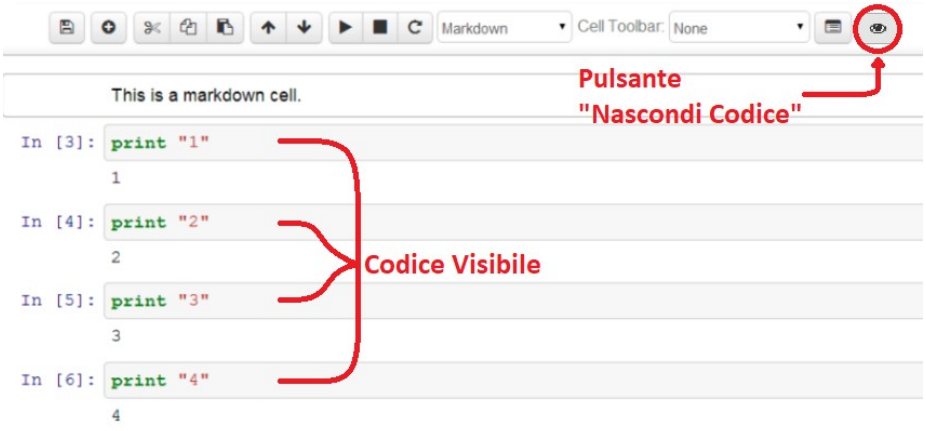
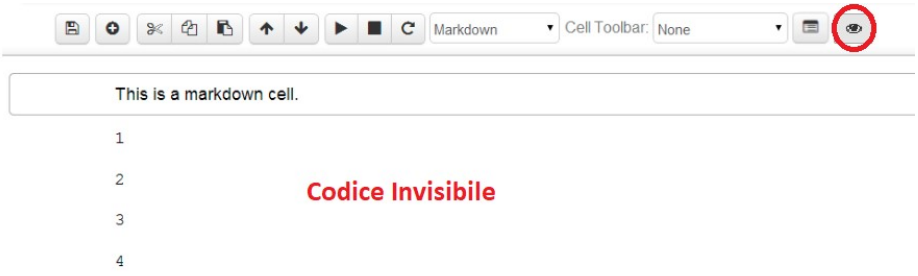



Fig. 20 – Estensione Hide input all - prima, fonte *Jupyter Notebook*

Fig. 21 – Estensione Hide input all - dopo, fonte *Jupyter Notebook*




Per questa estensione esiste anche un'altra versione, rinominata *Hide input*, che permette di scegliere quali celle nascondere e quali, invece, mantenere visibili. La scelta sarà determinata dal livello di flessibilità e personalizzazione richiesti.

Come anticipato, *Hide input* e *Hide input all*, aiutano nella gestione del rischio di diffusione di dati sensibili e/o strategici. Con una sola operazione consentono di rendere un Notebook idoneo ad essere utilizzato per una presentazione interna all'azienda, una classe o una conferenza. Sarà comunque necessario vigilare sul documento, in quanto la soluzione offerta da queste due estensioni, non ha una natura permanente. Il codice può essere rivelato nello stesso modo in cui è stato occultato.

Finora abbiamo analizzato cinque estensioni grazie alle quali, se non è possibile eliminare del tutto, perlomeno riducono l'impatto che hanno le limitazioni di Jupyter sulla valutazione dei suoi pro e dei suoi contro. A seguire, una breve analisi introduttiva di quelle estensioni che permettono di semplificare e/o migliorare l'esperienza, ovvero che vanno ad aggiungersi ai lati positivi del software.

*Codefolding* è un'estensione che similarmente a *Hide input*, va a nascondere alcune parti del documento, Figure 22-23.

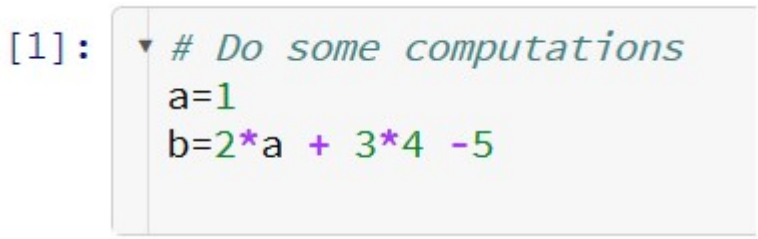



Fig. 22 – Estensione Codefolding - dopo, fonte *Jupyter Notebook*

Fig. 23 – Estensione Codefolding - dopo, fonte *Jupyter Notebook*


Come per *Hide input*, questa funzionalità esprime al meglio la sua utilità nei casi in cui è preferibile, se non addirittura imperativo, l'occultamento di informazioni sensibili e/o strategiche, oppure semplicemente a fini accademici. Un documento espandibile punto per punto permette un maggiore livello di comprensibilità dello stesso. Questa compressione visiva risulta molto utile anche in fase di scrittura del codice, mantenendo lo schermo sempre pulito ed ordinato ad un grado inimmaginabile senza *Codefolding*.

L' estensione *Hinterland*, fa si che Jupyter mostri un menu di suggerimenti ed auto compilazioni durante la scrittura nelle celle, Figura 24.

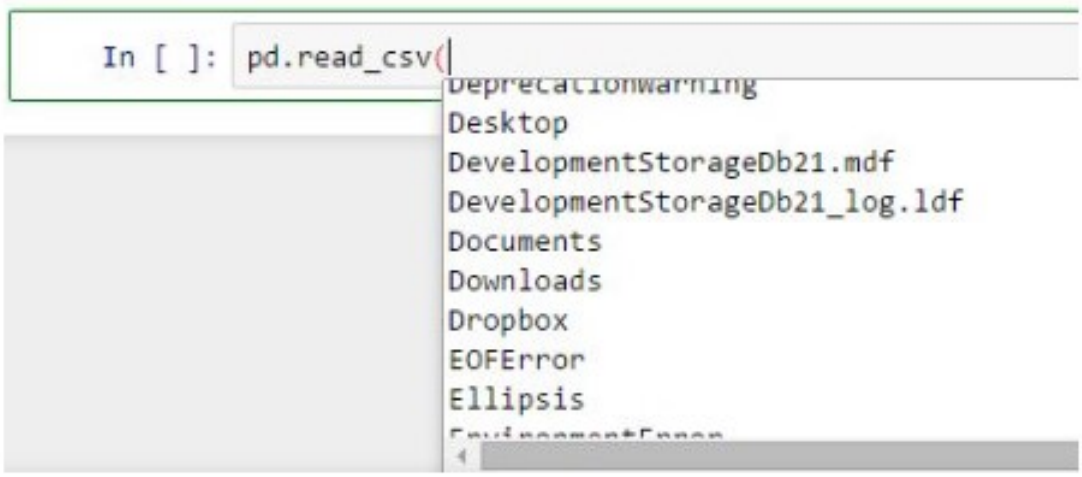



Fig. 24 – Estensione Hinterland, fonte endtoend.ai (Lee, 2018)

Anche se può sembrare poca cosa, *Hinterland*, è uno strumento che fa la differenza, alleggerendo il carico e facilitando la scrittura, rappresentando un piccolo, ma significativo, passo nella direzione di ambiente lavorativo migliore.

Le estensioni *Snippets* e *Snippets Menu* consentono di salvare un template o dei pezzi di codice in un documento JSON e di averli a portata di click ogni volta che lo si desidera, rispettivamente tramite un menu a tendina nella barra degli strumenti, Figure 25-26,

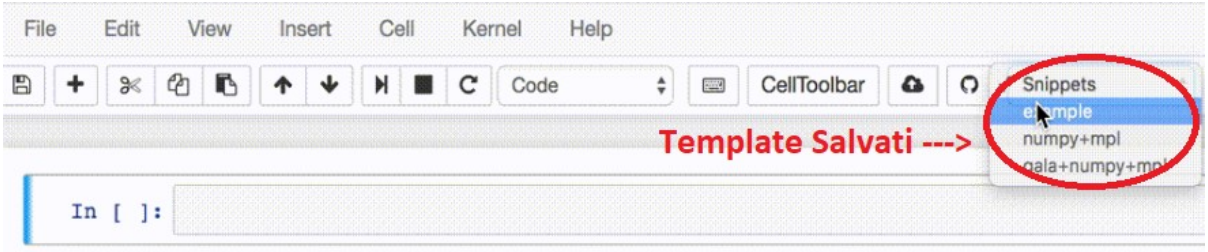



Fig. 25 – Estensione Snippets - prima, fonte *Jupyter Notebook*

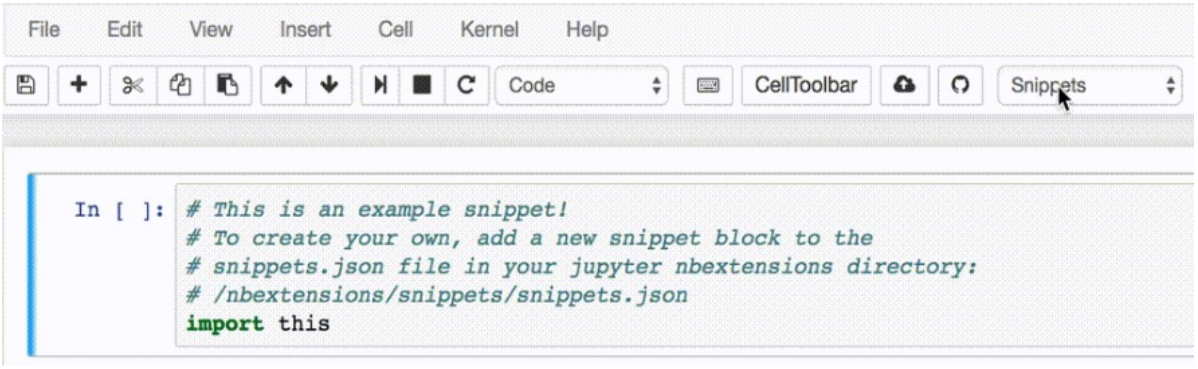



Fig. 26 – Estensione Snippets - dopo, fonte *Jupyter Notebook*

oppure tramite un menu personalizzabile nella barra dei menu, Figura 27.

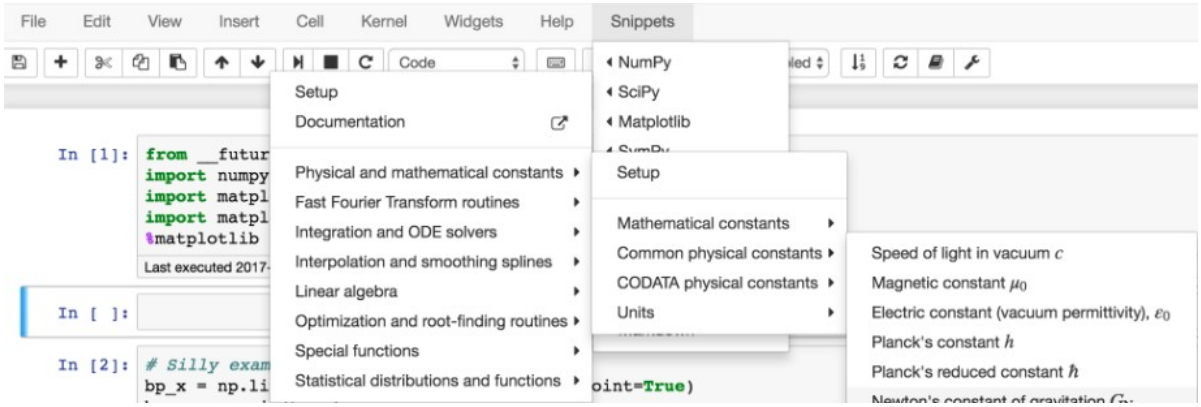



Fig. 27 – Estensione Snippets Menu, fonte *Jupyter Notebook*

Entrambe queste estensioni offrono un metodo veloce per inserire una grande quantità di codice con un click.

In un mondo globalizzato non poteva mancare *nbTranslate*, l'estensione per tradurre il contenuto delle celle da/in un'altra lingua. *nbTranslate* offre la possibilità di utilizzare Google Translate come motore di traduzione, grazie al quale ha accesso a più di cento lingue. L'estensione opera copiando la cella originale e creando una nuova cella con il contenuto nella lingua di destinazione selezionata, Figure 28-29.

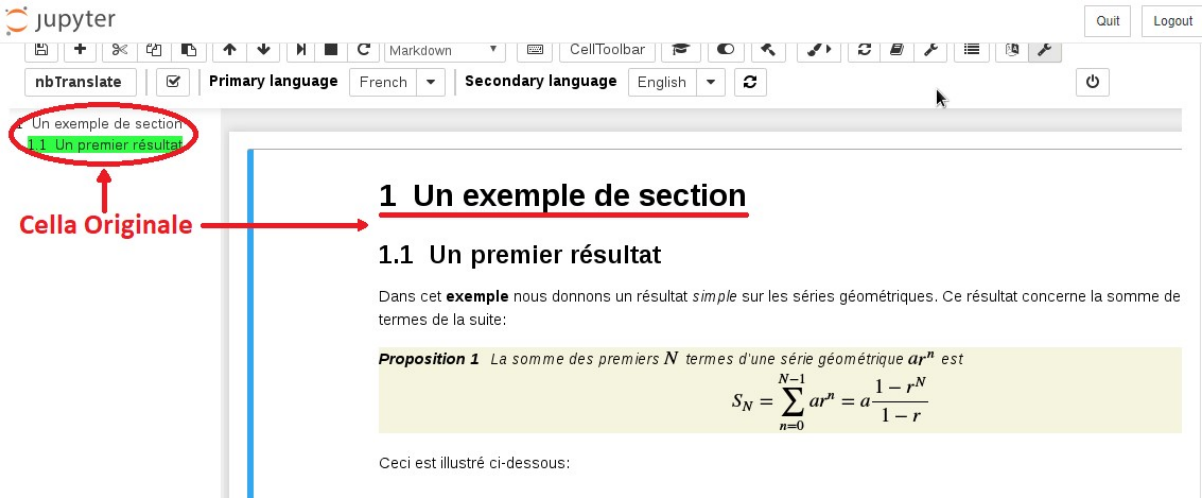



Fig. 28 – Estensione nbTranslate - prima, fonte *Jupyter Notebook*

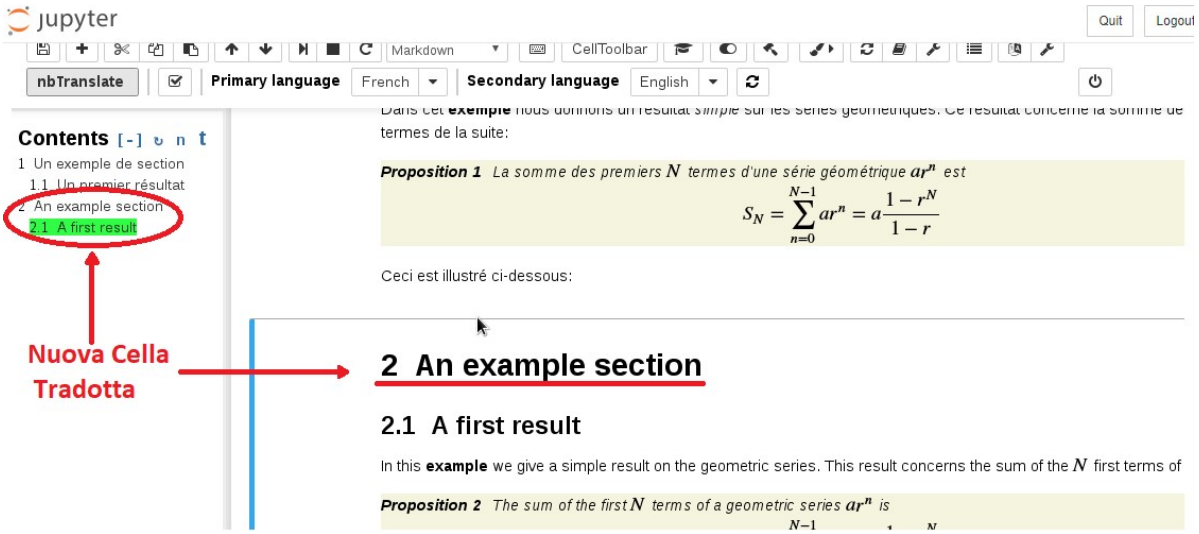



Fig. 29 – Estensione nbTranslate - dopo, fonte *Jupyter Notebook*

*Notify* aggiunge la possibilità di impostare e ricevere notifiche quando un kernel diventa *idle*, ovvero quando questo ha concluso il lavoro assegnatogli. Può essere utile qualora l'elaborazione delle celle richieda più di qualche secondo; permette così di concentrarsi su altri task durante l'attesa, sapendo di venire avvertiti da una notifica, Figura 30, così da poter riprendere il lavoro, non appena il kernel avrà finito il compito assegnatogli.

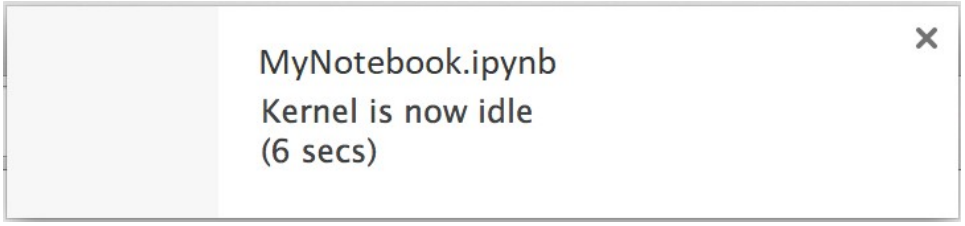



Fig. 30 – Estensione Notify, fonte *Jupyter Notebook*

L'estensione Scrachpad può sembrare banale ma nasconde un grande potenziale; essa offre la possibilità di usufruire di un blocco per appunti virtuale. Questa cella "fantasma", attivabile sia tramite icona, posizionata in basso a destra nella schermata, che tramite lo shortcut *ctrl+B*, mette a disposizione un terreno di prova che, sfruttando il medesimo kernel del notebook utilizzato, permetterà all'utente di sperimentare con il codice senza realmente modificare le celle, Figura 31. Una volta soddisfatto del risultato si procederà quindi alla scrittura nelle celle "ufficiali".

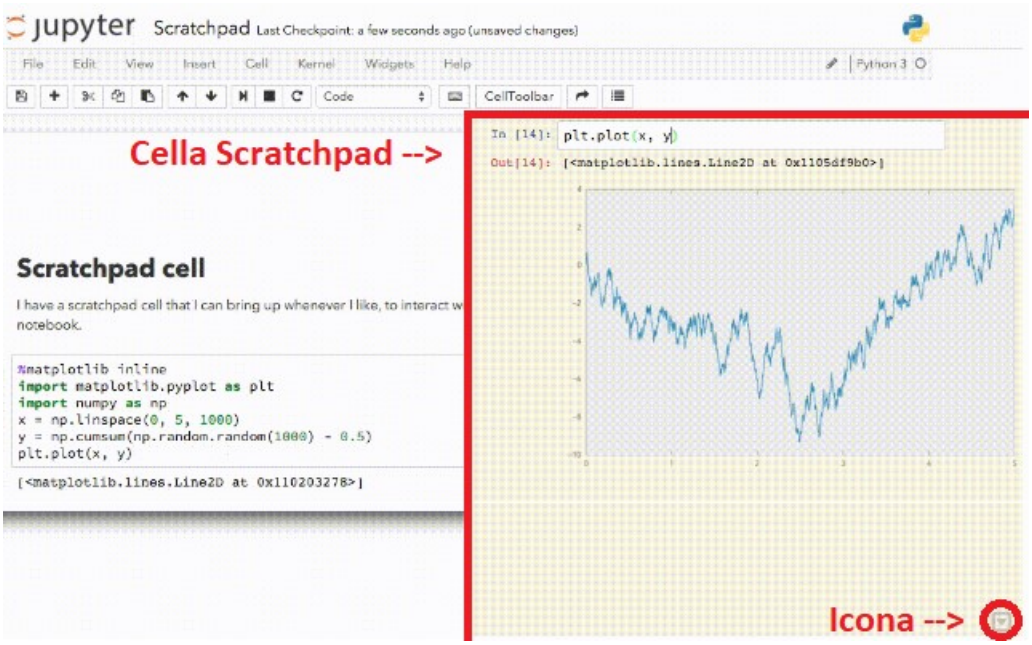



Fig. 31 – Estensione Scratchpad, fonte *Jupyter Notebook*

*Move selected cell*, è un estensione molto semplice ma che può tornare altrettanto utile, specialmente quando si necessita di riorganizzare parzialmente o per intero il lavoro svolto. Una volta installata essa consentirà, tramite gli shortcuts Alt+up e *Alt+down* di spostare la cella selezionata, rispettivamente verso l'alto e verso il basso, cambiandone quindi l'ordine di esecuzione nel Notebook. In alternativa è possibile anche trascinare la cella con il mouse, a seconda delle inclinazioni e delle preferenze dell'utente.

Infine, *Highlighter* offre una serie di pulsanti aggiuntivi nella toolbar tramite cui è possibile evidenziare del testo contenuto in celle di annotazione, Figura 32.

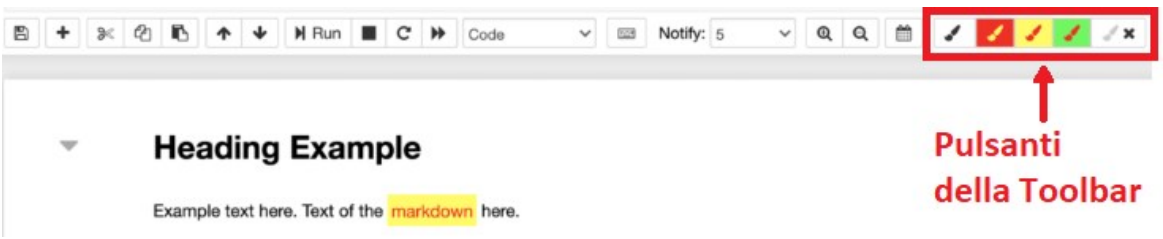



Fig. 32 – Estensione Highlighter, fonte *Jupyter Notebook*

Di default viene fornita una scelta di tre colori, ma è prevista la possibilità di customizzare lo strumento tramite il file *highlighter.css*. L'estensione permette di evidenziare il testo in qualsiasi momento, indipendentemente dal fatto che la cella sia in modalità di modifica o che questa sia già stata eseguita, e se nessun testo è selezionato l'intera cella verrà evidenziata. Le modifiche apportate verranno mantenute anche qualora il documento venga esportato in html o LaTex.

In conclusione, Jupyter Notebook presenta alcune debolezze significative in grado di evolvere in veri e propri problemi data la giusta situazione e mole di dati; nondimeno, grazie alle innumerevoli estensioni a disposizione è possibile non solo superare queste difficoltà ma addirittura aumentare il potenziale dello strumento.

# Capitolo 3. Il Caso Netflix

## 3.1 Motivazioni

Netlifx, la piattaforma di streaming e tv on-demand con la maggior quota di mercato, ha intrapreso nel 2017 un radicale processo di adozione di Jupyter Notebook (Ufford et al., 2018). La compagnia trae il suo vantaggio dalla gestione tempestiva di una mole elevata di informazioni riguardanti la sua clientela reale e potenziale. Tramite l'analisi di questi dati è possibile adattare in tempo reale l'offerta proposta agli utenti e rimanere così competitivi, offrendo un'esperienza personalizzata per ciascun individuo. Questo permette alla compagnia di continuare ad investire per innovare e sperimentare.

La gestione di tanti dati richiede un'infrastruttura non indifferente. Per dare qualche numero, nel 2018, il sistema gestiva più di 1 trilione di eventi, processati e registrati in un data warehouse in cloud, avente una capacità di 100PB, o Petabytes (Ufford et al., 2018). Per rendersi conto della vastità di queste dimensioni, 1 petabyte è composto da 1,024 terabytes, o 1 milione di gigabytes. Ogni giorno, più di 150,000 impiegati interpellano questo data warehouse, chiedendo le elaborazioni più disparate, dalle analisi alle raccomandazioni algoritmiche basate sul machine learning (Ufford et al., 2018).

Allo scopo di supportare tutto ciò, la compagnia ha progettato e implementato il Netflix Data Platform ed un ecosistema connesso, contenente una serie di strumenti quali: Genie, un servizio federato di esecuzione/elaborazione delle richieste, e Metacat, un metastore federato; questi permettono di semplificare la complessità del Data Platform, permettendo l'accesso e la gestione di un numero maggiore di utenti.

Il workflow di questo processo informativo può essere sintetizzato in fase esplorativa, fase preparativa, fase di sviluppo, fasi pre- e post-implementativa. Durante questo processo si susseguono diverse figure professionali, ognuna delle quali impiega strumenti e linguaggi diversi. Mentre un ingegnere dei dati impiegherà *Scala* in un ambiente *IntelliJ* per creare un aggregato da un dataset di trilioni di eventi di streaming, un ingegnere analitico utilizzerà tale aggregato per la stesura di un nuovo report sullo streaming globale tramite *SQL* e *Tableau*. Questo report servirà, infine, ad un data scientist per l'elaborazione di un modello di compressione sullo streaming,

realizzato impiegando *R* e *RStudio* (Ufford et al., 2018).

Tutto ciò, come accennato in precedenza, genera una gran mole di eventi che il sistema Netflix Data Platform dovrà gestire. Nelle fasi iniziali gli eventi possono essere la richiesta di visualizzazione di campioni di dati, queries di profilazione statistica e/o analisi esplorativa. Successivamente saranno necessarie azioni di pulizia del dato, di standardizzazione, trasformazione, de-normalizzazione ed aggregazione; questi eventi tendono ad essere frequenti e ripetitivi, rappresentando una delle parti più onerose sul budget temporale di un progetto. Infine, nelle fasi finali, sarà necessario implementare il codice nei server produttivi, generare modelli formativi/di addestramento, validare i dati, e programmare i workflows (Ufford et al., 2018).

Anche se inizialmente Jupyter Notebook era stato adottato esclusivamente come strumento di supporto ai data scientists nella creazione dei workflows, la compagnia si rese conto delle potenzialità e iniziò a diffonderne l'utilizzo. Jupyter Notebook racchiudeva gli strumenti necessari a svolgere e gestire la maggior parte degli eventi finora descritti, essendo stato progettato per girare codice, esplorare dati e presentare i risultati ottenuti.

Entrando più nello specifico, sono state tre motivazioni ad attirare l'attenzione del colosso dello streaming.

Jupyter Notebook utilizza un protocollo di comunicazione per l'esecuzione del codice che è agnostico dal punto di vista del linguaggio di programmazione utilizzato. Il protocollo impiegato per la comunicazione con i kernels è uno standard API. Tale protocollo permette un'architettura componibile in grado di separare dove il codice viene scritto, il Notebook, e dove invece viene eseguito, il kernel. Isolando l'interfaccia dal runtime, il software può sia adottare un numero elevato di linguaggi di programmazione, che mantenere una certa flessibilità nel modo in cui l'ambiente di esecuzione è configurato. L'unico requisito per l'impiego di un linguaggio di programmazione è l'esistenza di un kernel per quel linguaggio in grado di comunicare tramite il protocollo di Jupyter. Una volta questo requisito è stato soddisfatto, i notebooks possono eseguire il codice scritto nel linguaggio desiderato scambiandosi messaggi con quel kernel.

In secondo luogo, Jupyter Notebook salva i file creati nel formato editabile .ipynb in cui

vengono mostrati sia il codice sia l'output, consentendo di accedere a questi ultimi in un secondo momento senza la necessità di eseguire nuovamente il codice. Grazie alle celle di annotazione, inoltre, la lettura dei file è resa ancore meno complessa, poiché queste oltre a fornire il contesto permettono anche di tenere traccia di quello che accade all'interno del file.

Infine, il terzo motivo è dato dalla comodità dell'interfaccia del Notebook, grazie alla quale è possibile scrivere righe di codice ed avere immediatamente accesso ai risultati. Le potenzialità comunicative, sia in ottica informativa che commerciale sono evidenti.

## 3.2 Casi d'Uso

In Netflix, Jupyter Notebook ha principalmente tre casistiche d'impiego: l'accesso ai dati, l'utilizzo di templates, e la gestione dei workflows (Ufford et al., 2018).

Il primo caso, l'accesso ai dati, è stato il primo che ha visto l'introduzione e l'impiego di Notebook nella compagnia; lo scopo iniziale per l'impiego di questa tipologia di software era la gestione dei workflows. Una volta scoperto il potenziale, Netflix ha deciso di incrementare l'uso di Jupyter Notebook incorporandolo nella Netflix Data Platform come strumento di grande importanza e rilievo (Ufford et al., 2018). La compagnia, adottando la prospettiva dell'utente, ha riconosciuto facilmente le potenzialità offerte da Jupyter, come l'esecuzione iterativa del codice, la possibilità di effettuare un controllo praticamente istantaneo del risultato nonché poterlo visualizzare, il tutto tramite un singolo ambiente di sviluppo in cloud. L'integrazione di Jupyter nella Netflix Data Platform ha permesso alla compagnia di mantenere la propria libreria Python con cui consolidare l'accesso agli APIs della piattaforma; gli Application Program Interface(s) ovvero un software intermediari che permettono la comunicazione tra due applicazioni. Questa configurazione ha virtualmente reso possibile, ai dipendenti della compagnia, l'accesso programmatico all'intera piattaforma tramite l'utilizzo di un Notebook, rendendo Jupyter lo strumento più utilizzato per i lavori data-related in Netflix (Ufford et al., 2018).

L'adozione di Jupyter Notebook e la sua integrazione con Netflix Data Platform hanno stimolato la ricerca di nuovi ambiti di utilizzo, come nel caso dell'utilizzo di templates. Nello specifico la compagnia ha customizzato il software introducendo i Notebooks

parametrizzati, i quali consentono sia di specificare i parametri nel codice, che accettare valori di input a runtime, rendendoli così dei template riutilizzabili a disposizione degli utenti. Per semplificare la comprensione di questo ultimo passaggio basti pensare ai seguenti esempi pratici di utenti e del loro utilizzo dei template notebook. I Data Scientists all'interno delle compagnie utilizzano i template per effettuare dei test utilizzando coefficienti diversi e per la somma dei risultati ottenuti; i Data Engineers, utilizzano i template per effettuare campionamenti e controlli sulla qualità dei dati durante i diversi stadi dei processi; i Data Analysts preparano, mediante i template, differenti analisi ed elaborazioni che verranno successivamente utilizzate dagli stakeholders; i Software Engineers, infine, li utilizzano ogni qualvolta si verifichi un malfunzionamento, per migliorare la qualità dei messaggi e della reportistica in merito al problema (Ufford et al., 2018).

La terza casistica d'impiego di Jupyter Notebook in Netflix riguarda la gestione dei workflows. La compagnia ha sfruttato la struttura dei Notebook come un layer unificatore in cui i workflows vengono schedulati, creati e gestiti. Grazie alla flessibilità che definisce la relazione tra notebook e kernel, e grazie alla caratteristica indipendenza e linearità d'esecuzione delle celle, la mappatura delle anomalie e delle problematiche è semplice e precisa. Questo permette agli utenti la programmazione di un workflow completo in un singolo notebook, creando così un elemento flessibile, scalabile e semplice da replicare. Talvolta un notebook viene utilizzato in combinazione ad altre applicazioni, con la funzione di test ed archivio. All'interno del notebook vengono riversati il codice, i parametri, la configurazione, i logs ed i messaggi d'errore. Questo permette di avere accesso ad un ambiente di debug interattivo specifico per quel job (Ufford et al., 2018).

### 3.3 Customizzazione

Netflix ha ideato e sviluppato diverse soluzioni per quanto riguarda la customizzazione di Jupyter Notebook, per far sì che quest'ultimo potesse integrarsi con Netflix Data Cloud e soddisfi i bisogni dall'azienda; in particolare, è possibile identificare quattro progetti significativi, ovvero *nteract*, *Papermill*, *Commuter* e *Titus*, e come questi sono inseriti nell'infrastruttura della compagnia, Figura 33.

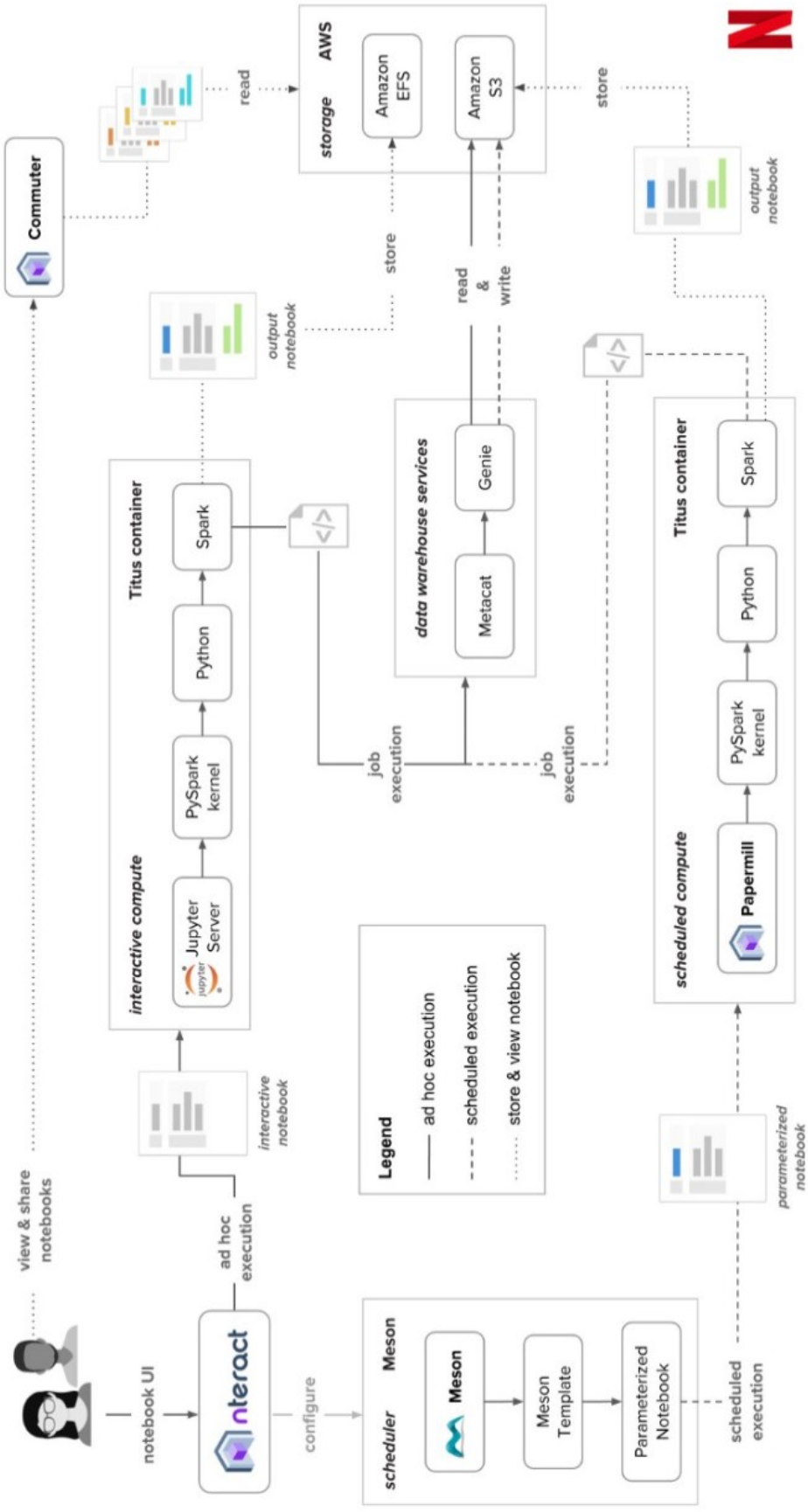



Fig. 33 – Infrastruttura Notebook in Netflix, fonte *Ufford et al., 2018*

Il progetto *nteract* si focalizza sull'aspetto, sull'interfaccia del software, aggiungendo, tra le altre varie migliorie/customizzazioni, la possibilità di muovere e spostare liberamente le celle, *drag-and-drop*, ed incorporando un sistema di data-explorer interno al notebook.

*Papermill* costituisce una libreria del progetto *nteract* e permette l'analisi di uno o più notebooks, dopo che questi sono stati parametrizzati ed eseguiti simultaneamente, consentendo così la comparazione. I risultati ottenuti vengono poi sommati ed aggregati per favorire una comprensione ed un'ulteriore analisi approfondita.

Il servizio *Commuter*, anch'esso parte di *nteract*, permette la consultazione e la condivisione di notebooks; è leggero e scalabile, in grado di offrire una versione dei contenuti APIs in un formato compatibile con Jupyter, facilitandone così la lettura; inoltre, *Commuter* mette a disposizione una directory per l'esplorazione dei notebooks disponibili, siano essi archiviati in locale o su *Amazon S3*, un servizio cloud di archiviazione/memorizzazione.

Infine, il progetto *Titus*, è una piattaforma per la creazione e la gestione di *container*, pacchetti che racchiudono al loro interno un'applicazione e le sue dipendenze per favorirne, semplificarne e velocizzarne lo sviluppo e produzione. *Titus* rappresenta uno strumento affidabile, scalabile ed integrabile in cloud con *Amazon Web Services (AWS)*; è largamente utilizzato in produzione per il funzionamento dei servizi di streaming e per la gestione dei sistemi di contenuti e consigliati/raccomandazioni.

### 3.4 Infrastruttura a Supporto

Per quanto riguarda l'infrastruttura in cui vanno ad inserirsi i progetti affrontati nella sezione precedente, è possibile effettuare una distinzione in tre componenti: Storage, Calcolo ed Interfaccia.

Come già precedentemente menzionato, Netflix Data Platform si appoggia ad Amazon S3 ed Amazon Elastic File System (EFS) per la parte di Storage, l'immagazzinamento e l'archiviazione dei dati; questo workflow è previsto ed integrato con Jupyter. Ogni utente avrà accesso alla propria directory su EFS dove sarà presente uno spazio personale contenente tutti i notebooks creati o caricati da quell'utente (Ufford et al., 2018).

Al fine di semplificare ed efficientare la pratica di storicizzazione e archiviazione dei notebooks viene utilizzata la combinazione *utente-workspace-nomedelfile*; un esempio del risultato di questa pratica è visibile in Figura 34.


Fig. 34 – EFS Filesystem

Questo permette una veloce e precisa identificazione sia dell'utente autore del notebook che della posizione di quest'ultimo nell'enorme mole di dati contenuti in EFS. La soluzione consente anche una facile condivisione dei notebooks, tramite l'uso del path/URL, generando però nuove problematiche legate a sovrascritture dei file dovute all'accesso simultaneo di più utenti. Iniziarono da qui lo sviluppo e l'implementazione del servizio *Commuter*, per far sì che gli utenti possano condividere file notebook in modalità di sola visualizzazione, senza mettere a rischio lavori di produzione o notebooks in esecuzione.

Il sistema facilita, inoltre, la schedulazione dei notebooks, tramite l'utilizzo di una directory comune su Amazon S3; quando l'utente pianifica l'esecuzione un notebook in EFS, il sistema *scheduler* ne fa una copia nella directory comune in S3. Questa copia verrà considerata come backup, fonte di confronto e rispristino in caso di anomalie, il *source notebook*. Ogni qualvolta lo *scheduler* dovrà eseguire un notebook, istanzierà una copia del *source notebook* da utilizzare. Questa copia verrà effettivamente eseguita, e conterrà al suo interno sia il codice, che il risultato e i logs di ogni cella; fungerà da documentazione dell'esecuzione, chiamata l'*output notebook* (Ufford et al., 2018).

La parte di Calcolo è gestita sempre attraverso AWS, tramite una soluzione altamente scalabile basata su containers, dei pacchetti che possono contenere dalle query ai notebooks; un nuovo container viene fornito all'utente ogni volta questi avvia un notebook. Le dimensioni di questi pacchetti possono essere ampliate quando richiesto dall'utente mediante un'interfaccia.

Insieme a questi container l'utente ha a disposizione un ambiente di sviluppo unificato

contenente librerie comuni già preinstallate oltre ai kernel più utilizzati a disposizione. Questo sistema consente di ridurre i tempi di configurazione ed evita il proliferare di ambienti di sviluppo diversi.

Il progetto *Titus*, oltre a permettere la gestione ed il controllo di questa soluzione a containers, consente un ulteriore step tramite l'assegnazione degli utenti a gruppi di sicurezza, introducendo i ruoli, e gestendo delle specifiche variabili legate alla sicurezza negli ambienti comuni di sviluppo (Ufford et al., 2018). Questo si traduce, per l'utente, in una guida, un reindirizzamento automatico alle risorse disponibili a cui è autorizzato ad accedere, evitando lungaggini e perdite inutili di tempo legate alla navigazione nell'infrastruttura, prediligendo invece il tempo speso nell'analisi dei dati e la user experience.

L'Interfaccia rappresenta uno dei nodi più complessi da affrontare, poiché deve soddisfare un alto numero di utenti con esigenze diverse tra loro. L'approccio adottato da Netflix è caratterizzato dalla semplicità, minimalismo e modularità. Al cuore della soluzione si trova il progetto *nteract*, un frontend reattivo in grado di rispondere alle esigenze dei diversi tipi di utente permettendo la customizzazione dell'interfaccia in base alle risorse richieste.

### 3.5 Problemi e Soluzioni

La compagnia, fin dalla fase esplorativa in cui stava raccogliendo informazioni riguardo un numero di software che potevano rappresentare dei potenziali candidati per l'adozione, tra cui Jupyter Notebook, aveva notato delle caratteristiche che potevano creare delle difficoltà. I notebooks, per la loro natura frammentata in cellule, possono essere ardui da testare e non prevedono la configurazione dinamica dell'esecuzione. Oltre a ciò, è richiesto un server notebook per l'esecuzione, creando una dipendenza architetturale. Netflix però trovò il modo per superare queste criticità grazie ai progetti custom affrontati in precedenza, *Papermill* in particolare.

Come già anticipato nel paragrafo 3.3, *Papermill* non fa altro che risolvere le problematiche appena menzionate in modo semplice, configurabile ed affidabile. Questa library, parte di *nteract*, non fa altro che prendere un notebook dal sistema EFS grazie al suo path univoco, aggiungerci dei parametri di configurazione ed infine

eseguire il notebook. La sua particolarità sta nello scattare un'*istantanea* della situazione ad ogni cella eseguita, salvandosi il risultato in un *output notebook* nel sistema S3, Figura 35 (Seal et al., 2018).


Fig. 35 – Papermill semplificato, fonte (Seal et al., 2018)

Il risultato, oltre a contenere tutti gli elementi e le informazioni di esecuzione, costituisce anche un template per ripeterne l'analisi in un qualsiasi momento; questo verrà archiviato in un contenitore apposito in S3 gestito da *Commuter*, che ne permetterà la visualizzazione. Da qui l'utente potrà averne accesso ed eseguirne analisi e debug per risolverne anomalie/problemi, controllarne i risultati e creare nuovi templates senza alterare o impattare il notebook, e quindi il workflow, originale (Seal et al., 2018)..

Questa soluzione risolve anche il problema della dipendenza architetturale, dovuta le server notebook, in quanto *Papermill* controlla i propri processi runtime, evitando di utilizzare i kernels, Figura 36.

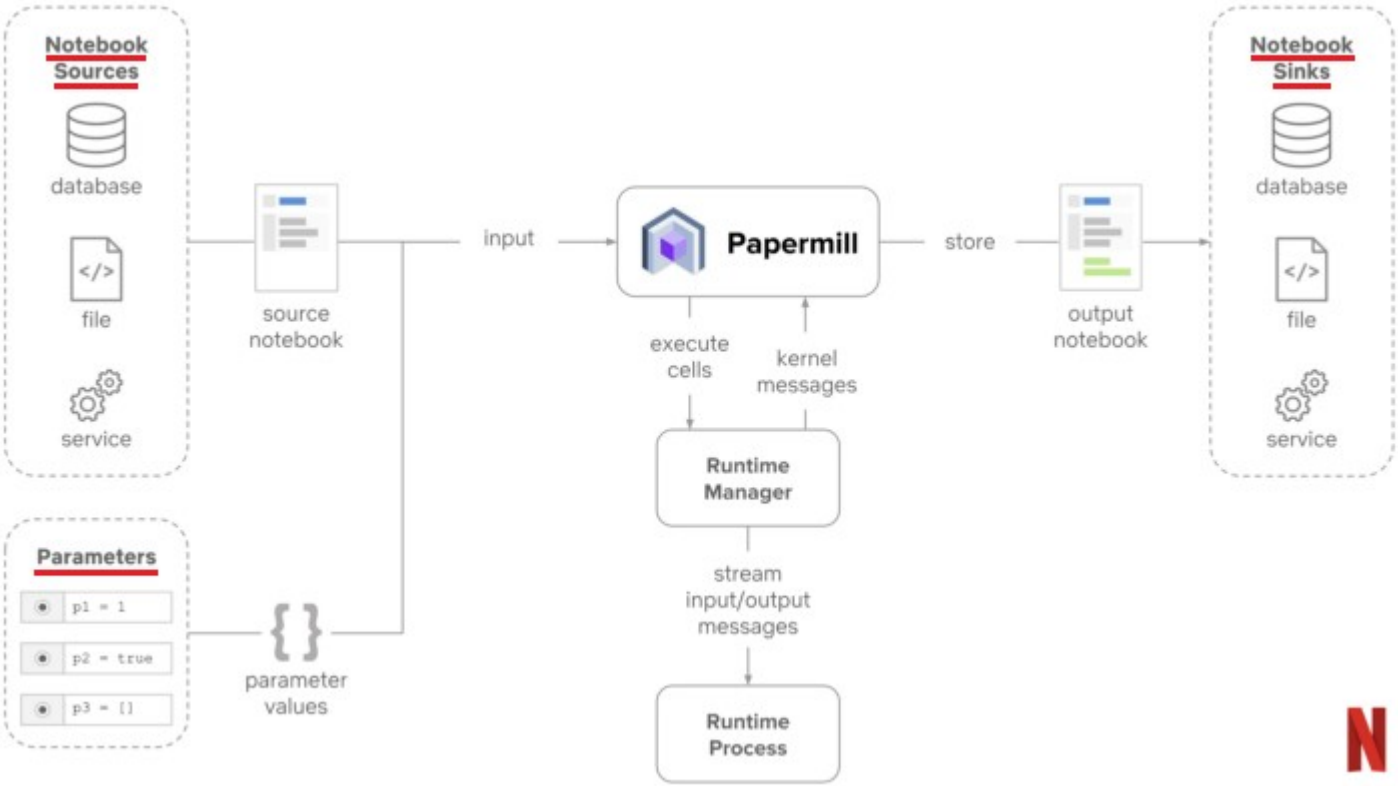



Fig. 36 – Papermill in-depth, fonte (Seal et al., 2018)

Questo ha permesso sia di migliorare il grado di affidabilità dei notebooks che di implementare un processo in grado di utilizzarli come input e produrli in output, completamente eseguibili e facili da condividere mediante la stessa interfaccia.

Nonostante le migliorie apportate a Jupyter Notebook grazie all'introduzione di Papermill, il software mancava ancora di un layer per la schedulazione delle attività, per l'esecuzione programmata dei jobs, in grado di appoggiarsi ed integrarsi con *Papermill* e creare una sinergia. Oltre a ciò, il componente doveva rispettare sei criteri per soddisfare a pieno i requisiti e rappresentare il match migliore.

Innanzitutto, doveva essere predisposto alla ricezione di eventi esterni, ed attivarsi o attendere secondo le indicazioni ricevute; doveva poter eseguire il job programmato anche all'interno di un ambiente di esecuzione controllato, come ad esempio *Docker* prima dell'introduzione di *Titus*, Figura 37. Doveva essere in grado di tener traccia delle varie esecuzioni e dei fallimenti, ed eventualmente esporne i dati a riguardo su richiesta. Il software, inoltre, doveva prevedere dei controlli per evitare anomalie nel caso due jobs venissero eseguiti allo stesso momento; doveva predisporre la possibilità di configurare in modo dinamico le casistiche di riavvio/rilancio dell'esecuzione e

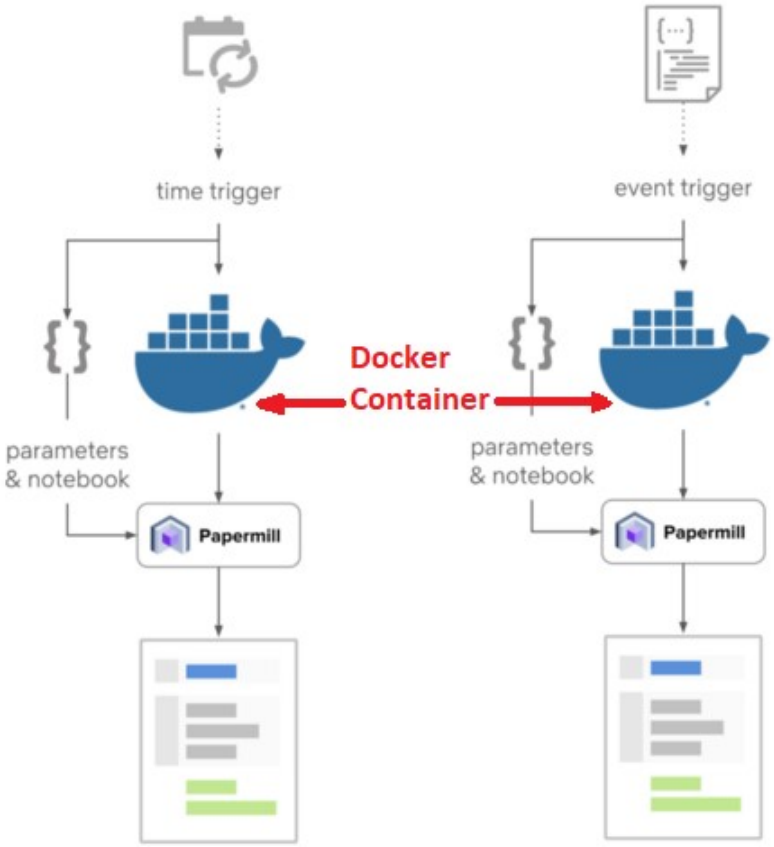



Fig. 37 – Flusso di Schedulazione con Docker, fonte (Seal et al., 2018)

consentire la suddivisione in ruoli, e dei privilegi associati, degli utenti (Seal et al., 2018).

Tra le molte soluzioni prese in considerazione, la scelta è ricaduta su *Meson*, un software di schedulazione di proprietario Netflix. Oltre a soddisfare i requisiti sopra menzionati, *Meson* si distingue per le elevate capacità integrative con il sistema e l'infrastruttura della compagnia, essendo stato sviluppato all'interno di questi.

Grazie agli aggiustamenti descritti finora, Jupyter Notebook è stato selezionato ed integrato nell'infrastruttura Netflix diventando il mezzo di Estrazione, Trasformazione e Caricamento (ETL) più diffuso, nonché la maggior fonte di reportistica interna.

### 3.6 Polynote

La compagnia di streaming non si è accontentata dell'adozione di Jupyter Notebook, anzi ha continuato a sperimentare alternative che si adattassero ancora di più alle proprie esigenze, in particolare uno strumento con una maggiore integrazione e supporto per Scala, linguaggio predominante della piattaforma di machine learning proprietaria. Da questa ricerca è nato *Polynote*, un nuovo ambiente di sviluppo basato su notebook.

Annunciato e rilasciato a fine 2019, il nuovo software open source presenta miglioramenti significativi, molti dei quali operano sui punti deboli di Jupyter

Notebook. Come prima cosa, le difficoltà incontrate da Jupyter nel tener traccia del codice a causa della struttura, soprattutto quando è richiesta l'esecuzione di un numero significativo di celle, vengono aggirate grazie and una nuova tipologia di notebook. Questo essendo in grado di tenere traccia della posizione di ciascuna cella all'interno del documento, al momento di eseguirle incontra molte meno difficoltà del sistema precedente. In secondo luogo, Polynote migliora la user experience, integrando nativamente un sistema di auto compilazione interattiva del codice, in grado di consigliare anche i parametri da utilizzare. È, inoltre, in grado di evidenziare eventuali errori di sintassi nel codice ed offrire supporto durante la formattazione del testo, compatibile con LaTex (Smith et al., 2019).

Altre caratteristiche di Polynote includono un'interfaccia (UI) modificata rispetto a Jupyter, che consente la distinzione tra gli elementi attivi/in esecuzione da quelli inattivi con maggiore facilità, Figura 38, il salvataggio delle impostazioni di configurazione del notebook all'interno dello stesso ed un sistema nativo per la visualizzazione dei dati contenuti in un dataset, disponibile senza l'istallazione di altri componenti o la scrittura di nessun codice (Rodriguez, 2020).

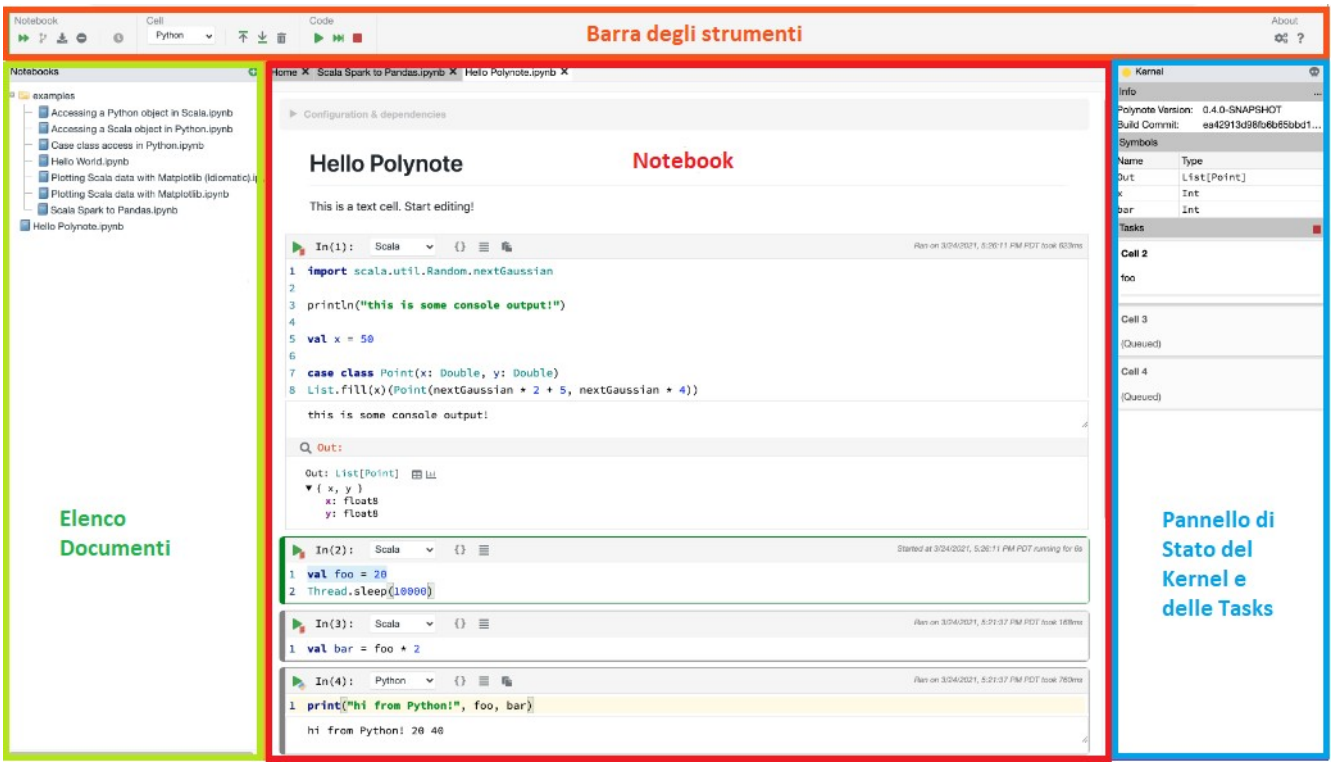



Fig. 38 – Polynote UI, fonte (Polynote, Tour)

Una differenziazione importante rispetto a Jupyter Notebook è l'interoperabilità

linguistica, ovvero la possibilità di scrivere ogni cella utilizzando un linguaggio di programmazione diverso senza perdere le funzionalità tecniche; ad esempio, il software è in grado di riconoscere una variabile definita in una cella ne permetterne l'utilizzo in un'altra anche se il codice utilizzato nella seconda è differente. Naturalmente questa possibilità è soggetta a limitazioni dovute alle differenze tecniche di alcuni linguaggi, ma rappresenta comunque uno strumento molto potente a disposizione. Va chiarito che, almeno per il momento, il numero di linguaggi non è paragonabile a Jupyter Notebook, infatti, sono supportati solamente Scala, Python, SQL e Vega (Titcombe, 2019).

Oltre al limitato numero di linguaggi di programmazione compatibili, Polynote ha anche un altro svantaggio significativo. Il software supporta solamente due sistemi operativi, ovvero, OSX e Linux, e non prevede, almeno per ora, la compatibilità con Windows (Titcombe, 2019). Com'è possibile vedere dalla Figura 39, i sistemi operativi appena citati rappresentano, approssimando per difetto al numero intero, rispettivamente il 14%, il 2% ed il 75% del mercato (GlobalStats, 2022).

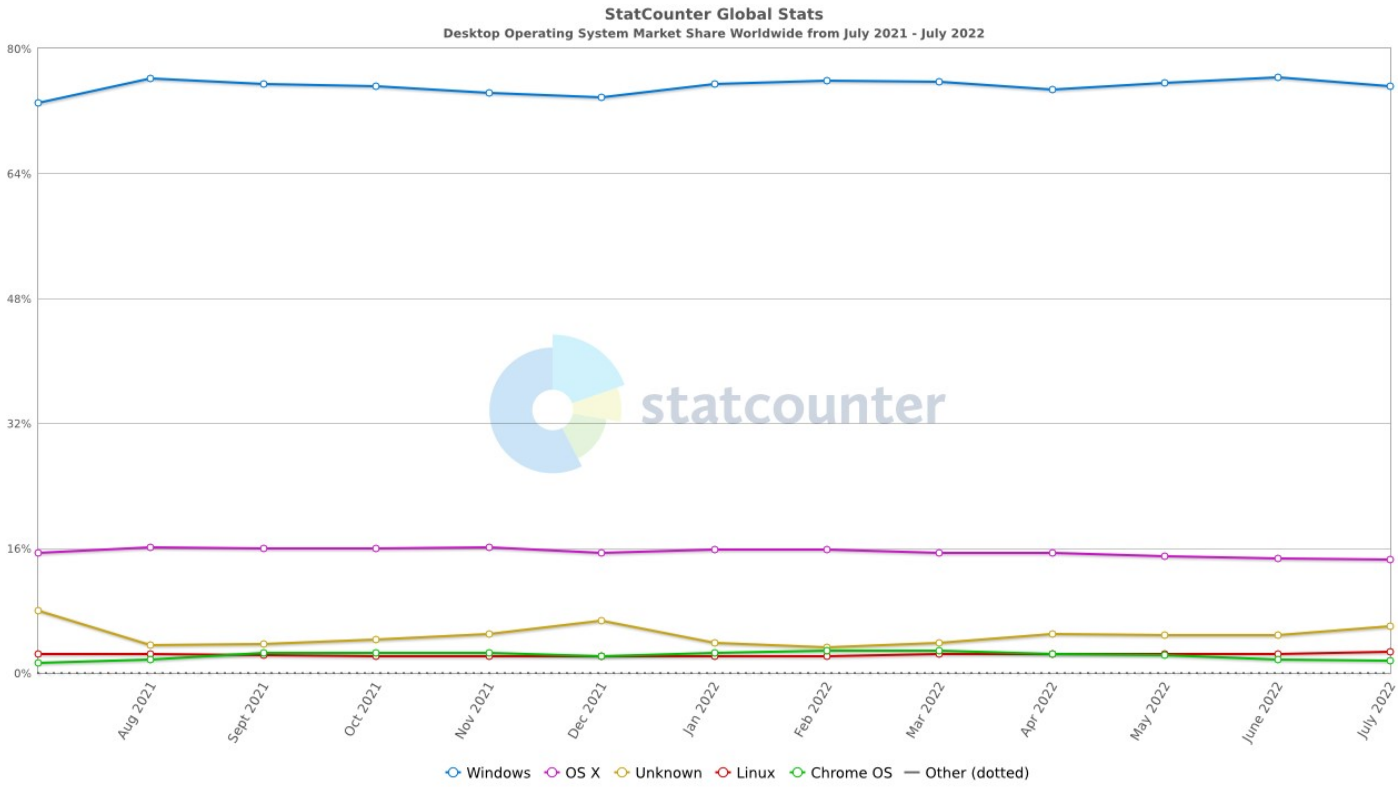



Fig. 39 – OS Market Share, fonte Global Stats: https://gs.statcounter.com/os-market-share

Il mancato supporto del sistema operativo Windows potrebbe rappresentare un fattore in grado di limitare significativamente la diffusione di Polynote, ostacolandone così il naturale percorso di ampliamento e miglioramento, avvenuto invece per Jupyter.

## Conclusione

Da quanto emerso in questo studio, Jupyter Notebook è un software valido e relativamente semplice da utilizzare, in grado di offrire uno strumento con grandi potenzialità alle imprese che decideranno di adottarlo ed implementarlo.

Come mostrato nel primo capitolo, la curva d'apprendimento non risulta per nulla ripida all'utente, il programma è molto intuitivo e permette una rapida familiarità con le varia funzionalità, grazie anche ad un'interfaccia è pulita e minimale. Pur mancando una formazione informatica ho avuto scarsa difficoltà nell'apprendere il funzionamento di Jupyter Notebook, e sono stato stimolato a testarne il maggior numero possibile anche solo per soddisfare il desiderio di toccare con mano i molti strumenti.

Nel secondo capitolo è possibile riscontrare che Jupyter Notebook, pur riscontrando alcune lacune, dispone di molte risorse grazie alle quali è in grado contrastare o addirittura superare le proprie limitazioni. Le estensioni supportate dal software sono numerose ed in espansione grazie alla community; solo una piccola porzione è stata rappresentata in questa analisi, le possibilità sono quindi elevate che una di quelle non trattate risponda alle esigenze ancora non soddisfatte.

Come nota personale aggiungo che il contributo della community di Jupyter si è rivelato significativo al fine dell'apprendimento, della raccolta delle informazioni e della scrittura di questo documento. Combinando le guide originale, disponibili sul sito di Project Jupyter, ai forum della community è possibile non solo apprendere il funzionamento del programma, ma si viene invogliati a proseguire l'esplorazione delle varie funzionalità ed estensioni.

In merito al terzo capitolo, è stato utile osservare come Jupyter sia stato modellato ed adattato alle esigenze del caso specifico, quando le sue caratteristiche abbiano fatto si che venisse adottato in differenti ambiti all'interno dell'azienda. La possibilità di un elevata customizzazione e la flessibilità nel suo impiego hanno permesso a Jupyter Notebook di rispondere a tutte le esigenze del caso.

Tra i vari punti di discussione rimasti senza una risposta decisiva mi permetto di segnalare la questione della gestione della sicurezza dei dati, le eventuali integrazioni

native supportate da Jupyter Notebook, nonché l'esplorazione di altri casi reali d'implementazione di questo software. Un quarto spunto, a mio parere interessante, potrebbe essere lo studio di quanto Jupyter soddisfi le esigenze di mercato correnti e cercare di capirne l'evoluzione futura.

Infine, in merito a Polynote, questo rappresenta un'alternativa interessante, anche se ancora troppo acerba, a Jupyter Notebook. Sarebbe opportuno effettuare uno studio approfondito sull'argomento al fine di comprenderne al meglio le caratteristiche ed i possibili sviluppi futuri, e se questi potrebbero renderlo un concorrente serio di Jupyter Notebook o meno.

## Fonti

(Anaconda, Tutorial) – "Getting started with Anaconda", *Anaconda Documentation*, https://docs.anaconda.com/anaconda/user-guide/getting-started/#open-nav-win [Consultato in data 10 Aprile 2022]

(BM, 2021) – "Jupyter Notebook for Data Science? The pros and cons of this tool", *Business Matters*, https://bmmagazine.co.uk/business/jupyter-notebook-for-data-science-the-pros-and-cons-of-this-tool/ [Consultato in data 30 Aprile 2022]

(GlobalStats, 2022) – "Operating System Market Share Worldwide", *statcounter*, https://gs.statcounter.com/os-market-share [Consultato in data 20 Agosto 2022]

(Grootendorst, 2021) – Grootendorst M., "Why Jupyter Notebooks Aren't All That Bad!", *Towards Data Science*, https://towardsdatascience.com/why-jupyter-notebooks-arent-all-that-bad-d75e90d02c3a [Consultato in data 30 Aprile 2022]

(Koehrsen, 2018) – Koehrsen W., "Jupyter Notebook Extensions", *Towards Data Science*, https://towardsdatascience.com/jupyter-notebook-extensions-517fa69d2231 [Consultato in data 14 Maggio 2022]

(Lee, 2018) – Lee S. R., "Jupyter Notebook extensions to enhance your efficiency", *endtoend.ai*, https://www.endtoend.ai/blog/jupyter-notebook-extensions-to-enhance-your-efficiency/ [Consultato in data 29 Maggio 2022]

(Li, 2019) – Li M., "What You Need to Know About Netflix's 'Jupyter Killer': Polynote", *Towards Data Science*, 25 Ottobre 2019, https://towardsdatascience.com/what-you-need-to-know-about-netflixs-jupyter-killer-polynote-dbe7106145f5 [Consultato in data 14 Agosto 2022]

(Lohr, 2012) – Lohr S., "The Age of Big Data", *The New York Times*, 11 Febbraio 2012, https://www.nytimes.com/2012/02/12/sunday-review/big-datas-impact-in-the-world.html [Consultato in data 6 Marzo 2022]

(OCDS, 2020) – "Why You Should Be Using Jupyter Notebooks", *Open Data Science*, 15 Luglio 2020, https://odsc.medium.com/why-you-should-be-using-jupyter-notebooks-ea2e568c59f2 [Consultato in data 30 Marzo 2022]

(PJ, About) – "About Us", *Project Jupyter*, https://jupyter.org/about [Consultato in data 4 Marzo 2022]

(PJ, Extensions) – "Extensions", *Project Jupyter*, https://jupyter-contrib-nbextensions.readthedocs.io/en/latest/index.html [Consultato in data 14 Maggio 2022]

(PJ, Hub) – "JupyterHub", *Project Jupyter*, https://jupyter.org/hub [Consultato in data 6 Aprile 2022]

(PJ, Install) – "Install and Use", *Project Jupyter*, https://docs.jupyter.org/en/latest/install/notebook-classic.html#prerequisite-python [Consultato in data 10 Aprile 2022)

(PJ, Intro) – "The Jupyter Notebook Introduction", *Project Jupyter*, https://jupyter-notebook.readthedocs.io/en/latest/notebook.html#opening-notebooks [Consultato in data 14 Aprile 2022]

(PJ, Kernels) – "Kernels: Programming Languages", *Project Jupyter*, https://docs.jupyter.org/en/latest/projects/kernels.html [Consultato in data 23 Aprile

2022]

(PJ, nbconvert) – "nbconvert: Convert Notebooks to other formats", *Project Jupyter*, https://nbconvert.readthedocs.io/en/latest/ [Consultato in data 8 Maggio 2022]

(PJ, Server) – "Running a notebook server", *Project Jupyter*, https://jupyter-notebook.readthedocs.io/en/stable/public_server.html [Consultato in data 7 Maggio 2022]

(Polynote, Tour) – "The Polynote UI", *Polynote*, https://polynote.org/latest/docs/tour/ [Consultato in data 14 Agosto 2022]

(Rodriguez, 2020) – Rodriguez J., "Netflix's Polynote is a New Open Source Framework to Build Better Data Science Notebooks", *Data Series*, 3 Agosto 2020, https://medium.com/dataseries/netflixs-polynote-is-a-new-open-source-framework-to-build-better-data-science-notebooks-4bdab6b8d0ae [Consultato in data 14 Agosto 2022]

(Seal et al., 2018) – Seal M., Kelley K., Ufford M., "Part 2: Scheduling Notebooks at Netflix", *Netflix Technology Blog*, 22 Agosto 2018, https://netflixtechblog.com/scheduling-notebooks-348e6c14cfd6 [Consultato in data 7 Agosto 2022]

(Segal, 2022) – Segal T., "Big Data", *Investopedia*, 28 Marzo 2022, https://www.investopedia.com/terms/b/big-data.asp [Consultato in data 30 Marzo 2022]

(Sjursen, 2020) – Sjursen S., "The Pros and Cons of Using Jupyter Notebooks as Your Editor for Data Science Work", *Better Programming*,

https://betterprogramming.pub/pros-and-cons-for-jupyter-notebooks-as-your-editor-for-data-science-work-tip-pycharm-is-probably-40e88f7827cb [Consultato in data 30 Aprile 2022]

(Smith et al., 2019) – Smith J., Indig J., Siddiqi F., "Open-sourcing Polynote: an IDE-inspired polyglot notebook", *Netflix Technology Blog*, 23 Ottoble 2019, https://netflixtechblog.com/open-sourcing-polynote-an-ide-inspired-polyglot-notebook-7f929d3f447 [Consultato in data 14 Agosto 2022]

(Titcombe, 2019) – Titcombe T., "Polynote: the new Jupyter?", *Towards Data Science*, 21 Dicembre 2019, https://towardsdatascience.com/polynote-the-new-jupyter-c7696a321b09 [Consultato in data 14 Agosto 2022]

(Twin, 2021) – Twin A., "Data Mining", *Investopedia*, 17 Settembre 2021, https://www.investopedia.com/terms/d/datamining.asp [Consultato in data 10 Marzo 2022]

(Xeus, Kernels) – "Usage", Xeus, https://xeus.readthedocs.io/en/latest/usage.html [Consultato in data 23 Aprile 2022]

(Ufford et al., 2018) – Ufford M., Pacer M., Seal M., Kelley K., "Beyond Interactive: Notebook Innovation at Netflix", *Netflix Technology Blog*, 16 Agosto 2018, https://netflixtechblog.com/notebook-innovation-591ee3221233 [Consultato in data 7 Agosto 2022]