Python_Anaconda (autore: Vittorio Albertoni)

Premessa

Sul mio blog, all'indirizzo *www.vittal.it*, nel febbraio 2017 ho pubblicato l'articolo «Python per tutti», archiviato nella categoria Programmazione, con allegato il file PDF «mondo_python» con l'obiettivo di divulgare la conoscenza del linguaggio di programmazione Python e di illustrare la ricchezza di strumenti che la comunità ha reso disponibili per estendere le potenzialità del linguaggio stesso e renderlo utilizzabile per una vastissima serie di esigenze.

Già in quella sede sottolineavo come Python fosse diventato il linguaggio preferito dai data scientists soprattutto per due motivi.

Il primo ricollegabile alla facilità di apprendimento e di uso del linguaggio, che fa risparmiare ai ricercatori un sacco di tempo e fatica nel districarsi tra difficoltà sintattiche come quelle di altri linguaggi, come il C e le sue derivate C++, Java, Ada, ecc., magari più adatti per esigenze diverse da quelle della data science.

Il secondo ricollegabile al fatto che, attorno a Python, grazie al fatto che dalla sua nascita è divenuto di pubblico dominio e a licenza libera, si è sviluppata una comunità che ha prodotto tutta una serie di pacchetti aggiuntivi al linguaggio base tra i quali fanno spicco alcuni particolarmente adatti ad elaborazioni matematiche e statistiche su grandi quantità di dati e che sono diventati strumenti imprescindibili per chi voglia affrontare l'analisi delle maree di dati che il mondo digitalizzato produce ogni minuto (big data).

In questo manualetto mi propongo di illustrare gli strumenti che il mondo Python rende disponibili per questo tipo di analisi: dalla presentazione di quali siano quelli di base in una impostazione minimalista, da artigiano, per arrivare al top: la distribuzione Anaconda.

Il contenuto dei Capitoli 2 e 3 indica la strada attraverso cui è possibile dotarsi di ciò che serve per avere una dotazione di tutto rispetto prescindendo da Anaconda. Per questa soluzione basta che disponiamo di poco più di 1 GB libero sul disco fisso.

Nel capitolo 4 parlerò della soluzione chiamata Miniconda per potersi costruire un ambiente su misura con gli strumenti della collezione Anaconda. Soluzione per la quale dobbiamo stanziare attorno ai 2,5 GB del nostro disco fisso.

Nel capitolo 5 parlerò, infine, di come da Miniconda possiamo passare a Anaconda e della più semplice soluzione di installare in un colpo solo Anaconda dotando il nostro computer di tutto il software open source che si trova in circolazione per la Data Science. Soluzione per la quale dobbiamo disporci ad occupare quasi 4 GB del nostro disco fisso.

Dal momento che si tratta di un impiego di risorse non irrilevante è bene che, prima di prendere una decisione sulla strada da imboccare, ci si renda conto di cosa significa prendere una strada piuttosto che l'altra, sapendo comunque che:
. se prescindiamo da Anaconda tutto viene installato sul computer in modo tale che, per tornare indietro e recuperare le risorse di memoria dobbiamo provvedere alla disinstallazione di ciò che avevamo installato utilizzando gli strumenti del nostro sistema operativo per farlo;
. se scegliamo Miniconda o Anaconda tutto ciò che installiamo viene collocato in una directory dedicata e possiamo tornare indietro e recuperare le risorse di memoria semplicemente cancellando questa directory (non pensando che ciò possa avvenire in pochi secondi, data la quantità di file da eliminare);
. con installato Miniconda o Anconda il sistema Python di default per il nostro computer diventa quello che fa parte della distribuzione e tutto il restante Python che abbiamo sul computer diventa pleonastico e, se vogliamo utilizzarlo, dobbiamo andarcelo a cercare e lanciarlo ad hoc.

Indice

1 Di che cosa stiamo parlando

Ci sono tre termini che ricorrono quando parliamo di moderne tecnologie di sfruttamento dei dati: data science (scienza dei dati), machine learning (apprendimento automatico), artificial intelligence (intelligenza artificiale).

Su che cosa sia l'una cosa o l'altra e su come l'una cosa si rapporti all'altra esiste una fioritura di opinioni, a volte tra loro vistosamente contrastanti.

Mi pare che la più diffusa opinione sia quella che relega la così detta scienza dei dati alla rappresentazione dei dati stessi, alla loro interpretazione, alla ricerca di correlazioni esistenti tra loro, ecc.: in questi casi il computer è solo un computer, fa tabelle, grafici e calcoli.

Secondo questa opinione l'apprendimento automatico si ha quando il computer arriva ad aggiungere qualche cosa alla sola rappresentazione dei dati, arriva, per esempio, sulla base di una serie di situazioni che gli sono state prospettate, a prevedere cosa potrebbe succedere in presenza di una situazione diversa da quelle prospettate.

L'intelligenza artificiale si ha quando addirittura il computer arriva a compiere azioni che tengono conto di tutta una serie di dati che gli sono noti, come succede quando un'automobile che si guida da sola riconosce un ostacolo e frena, per evitarlo, in tempi e modi diversi a seconda se sta piovendo o se c'è il sole.

Sembrerebbe tutto chiaro, ma le difficoltà di classificazione esistono.

Un computer che prevede quali potranno essere le vendite del prossimo anno sulla base della serie temporale delle vendite e di altri fattori correlati, come andamento dei salari e quant'altro, degli anni precedenti, tutte cose che si fanno con semplici modelli regressivi, sta facendo data science o machine learning?

Un computer che gioca a scacchi basandosi su quello che è successo nel milione di partite che abbiamo messo nella sua memoria sta facendo machine learning o intelligenza artificiale?

Per quanto mi riguarda, visto che alla base di tutto ci sono sempre dei dati ed è sempre sulla base di dati che opera il computer, il termine di data science copre tutto quanto e machine learning e artificial intelligence ne sono delle specializzazioni.

Tra l'altro è solo così che si può giustificare il fatto che esistano, come esistono, testi che parlano esattamente delle stesse cose, a volte intitolati alla Data Science, a volte intitolati al Machine Learning.

Scontato il fatto che, se parliamo di intelligenza artificiale sviluppata da o su macchine diverse da un computer, come avviene nella robotica, al data scientist si deve affiancare l'ingegnere - o viceversa - e non basta più ciò che troviamo in Anaconda.

Così come è sempre un ingegnere (data engineer) che fa trovare al data scientist i dati raccolti e immagazzinati in modo che siano elaborabili.

2 La dotazione minimale

Gira e rigira, soprattutto se siamo dei neofiti che vogliono semplicemente muovere i primi passi nelle materie di cui al precedente paragrafo, basta che sul nostro computer sia installata una versione di Python arricchita dai seguenti sei moduli:

. numpy,
. scipy,
. matplotlib,
. sklearn (per esteso scikit-learn),
. pandas,
. nltk.

La versione di Python direi che ormai (siamo nel 2019) non può essere che la 3. Fino a qualche anno fa poteva esserci qualche problema in quanto non di tutti i moduli sopra elencati e di altri eventualmente da aggiungere esisteva una versione adatta per Python3.

Insieme a **Python** assicuriamoci di avere installato anche la **IDLE**, in modo da disporre di un ambiente di lavoro più funzionale rispetto alla spartana shell di Python[1].

Il modulo **numpy** serve per il calcolo matriciale e per l'algebra lineare e il modulo **scipy** serve per calcoli scientifici, tra cui quelli statistici.

Il modulo **matplotlib** serve per creare grafici di ogni tipo.

Il modulo **scikit-learn** è un'estensione di scipy che serve per il data mining e l'analisi dei dati (regressione, clustering, ecc.): nel linguaggio Python è identificato come sklearn.

Il modulo **pandas** serve per maneggiare dati di forma diversa, manipolarli e riaggregarli.

Il modulo **nltk** (acronimo di Natural Language ToolKit) serve per lavorare su testi.

All'indirizzo *https://www.scipy.org/* troviamo una completa documentazione sui primi cinque moduli e istruzioni per come installarli: il tutto in lingua inglese[2].

All'indirizzo *https://www.nltk.org/* troviamo la documentazione per nltk.

Ad evitare di installare cose che già sono installate consiglio di verificare scrivendo nella shell di Python il comando `import <nome_pacchetto>`: se il pacchetto viene importato senza messaggio di errore vuol dire che è già installato[3].

Tutto questo software, che spesso lavora in maniera interdipendente, è soggetto a continui aggiornamenti e arricchimenti non sempre sincronizzati tra loro. Può pertanto accadere, per esempio, che se installiamo la versione corrente di scikit-learn su un sistema sul quale è installata una versione di scipy di tre anni fa qualche cosa non funzioni.

E' bene pertanto installare i moduli tutti insieme e in presenza di una versione di Python che non sia stata installata oltre due anni prima.

Chi ha la fortuna di usare Linux può ovviare a questi inconvenienti installando tutto ciò che manca attraverso il gestore di pacchetti di sistema, meglio ancora il software Synaptic o il comando a terminale `apt install`. In questo modo, se la versione Linux è un po' vecchia, andremo ad installare software non aggiornatissimo ma che funziona.

Per quanto possa servire rammento che, usando il comando `pip install <nome_pacchetto>` (meglio `pip3 install <nome_pacchetto>` se lavoriamo con Python3) installiamo l'ultima versione del pacchetto presente nel repository di Python.
Per installare una versione diversa del pacchetto, in modo che sia allineata a quanto già installato, è possibile usare il comando `pip install <nome_pacchetto> == <versione>`.
La difficoltà sta nel sapere quale sia la versione adatta.

Ma qui stiamo parlando di come procurarci la dotazione minimale in maniera ruspante per cui qualche difficoltà bisogna superarla.

Il vantaggio è che con questa dotazione minimale abbiamo a disposizione tutto ciò che serve per eseguire le varie esercitazioni proposte dai testi in circolazione per l'apprendimento delle basi su data science e machine learning.

Il tutto avendo occupato non più di 800 MB del disco del nostro computer.

3 Arricchimento con iPython

La IDLE di Python, con la sua finestra di shell interattiva e il suo editor mi pare siano già ottimi strumenti, almeno per muovere i primi passi.

Molti preferiscono tuttavia utilizzare un'altra shell interattiva, che si chiama **iPython**, che, rispetto alla normale IDLE, presenta almeno questi due vantaggi:
. riconosce ed esegue anche i comandi della shell Unix, per cui, dalla shell di iPython, oltre che poter interagire con il linguaggio Python possiamo navigare nel filesystem e intervenire sui file con i comandi che sono ben noti a chi sa utilizzare i terminali dei sistemi operativi Linux e Mac OS X, potendo accedere a questa utilità anche utilizzando il sistema operativo Windows;

[1] A chi abbia problemi a seguire ciò che sto dicendo consiglio la lettura dei capitoli 1 e 2 del manualetto «mondo_python» richiamato in premessa.

[2] Per quanto riguarda l'installazione suggerisco la lettura dei capitoli 3 e 4 del manualetto «mondo_python» richiamato in premessa.

[3] Rammento che il nome pacchetto da utilizzare per importare scikit-learn è sklearn

. ci offre una code completion molto più efficiente (nella IDLE dopo che abbiamo digitato il punto che separa il nome dell'oggetto dal nome del metodo che vogliamo richiamare, automaticamente compare la lista dei metodi tra cui possiamo scegliere; in iPython possiamo far comparire ciò che manca per completare un'istruzione semplicemente premendo il tasto di tabulazione dopo che ne abbiamo scritto una parte).

A mio avviso, comunque, la vera utilità di iPython è quella di essere funzionale ad un altro arricchimento, che si chiama **iPython notebook**.

Con iPython notebook possiamo produrre documenti in formato html contenenti testo, codice, visualizzazione dei risultati dell'esecuzione del codice, oltre che file multimediali audio e video.

Anche prescindendo dalla multimedialità, questo strumento può essere molto utile per illustrare una ricerca e per produrre documentazione a scopo didattico.

La seguente figura 1 mostra un esempio di pagina in cui si illustra uno script di utilità matematica.

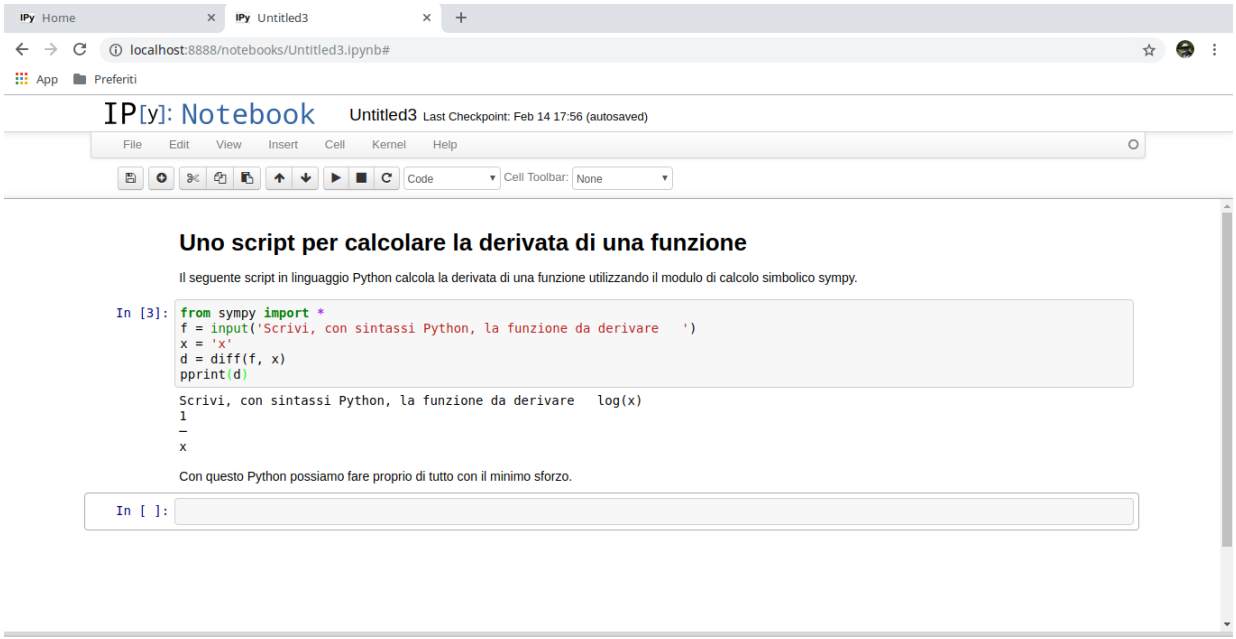

Figura 1: Finestra di lavoro di Notebook

In questa pagina ho inserito del testo, uno script e la descrizione della sua esecuzione con il risultato.

Oltre che memorizzare la pagina come pagina di lavoro (formato .ipynb) è possibile esportarla in formato .html e/o in formato .py.

Il file .html è visibile su un browser e può essere agevolmente convertito in formato .pdf (basta aprirlo con LibreOffice Write ed esportarlo come PDF).

Il file .py altro non è che lo script eseguibile che, nel caso, possiamo utilizzare per calcolare quante derivate vogliamo.

Attualmente iPython e iPython Notebook sono unificati nel progetto nominato **Jupyter** e si può installare il tutto ricorrendo al comando `pip3 install jupyter`.

Fatto questo, la shell iPython si apre con il comando a terminale `ipython3` e la finestra di lavoro per il notebook si apre con il comando a terminale `ipython3 notebook` o `jupyter notebook`.

Per vedere come funziona jupyter notebook consiglio la visione del filmato all'indirizzo *https://www.youtube.com/watch?v=KDA6MKh03bw*.

Tutto su Jupyter si trova sul sito *https://jupyter.org/*.

4 Miniconda

Nei due capitoli precedenti ho illustrato come sia possibile dotarsi praticamente di tutto ciò che serve per sviluppare progetti di data science anche importanti ed impegnativi limitando al massimo l'impiego di risorse di memoria grazie a procedure un po' ruspanti per l'installazione di ciò che serve, procedure che possono comportare qualche difficoltà nell'allineare i vari pacchetti con le relative dipendenze e, pertanto, creare installazioni che poi presentano qualche difficoltà di funzionamento.
Diciamo che se il nostro obiettivo è quello di fare qualche prova per imparare può andare bene così.

Se vogliamo affrontare le cose con più professionalità è bene che ci rivolgiamo altrove.

Per i pacchetti Python dedicati al calcolo e alla scienza esiste un repository alternativo a PyPI (quello che contiene i pacchetti installabili con pip) e, per chi usa il sistema operativo Linux, al repository della distro utilizzata.
Si tratta del repository collegato al gestore di pacchetti e ambienti di sviluppo **Conda**.

Di Conda possiamo sapere tutto visitando il sito *https://conda.io/en/latest/* e la prima cosa che vi troviamo è l'affermazione che «Conda è un sistema di gestione di pacchetti open source e un sistema di gestione di ambienti di sviluppo che funziona su Windows, macOS e Linux. Conda installa, esegue e aggiorna rapidamente i pacchetti e le loro dipendenze. Conda crea, salva, carica e passa facilmente da un ambiente di sviluppo all'altro sul tuo computer locale. È stato creato per i programmi Python, ma può creare pacchetti e distribuire software per qualsiasi linguaggio».

I vantaggi di Conda sono praticamente due:
. quando installiamo un pacchetto con Conda, se il pacchetto si installa sicuramente funziona; se c'è qualche cosa che non combina quanto a dipendenze e compatibilità con la versione del nostro sistema operativo il pacchetto non si installa e si evita così il rischio di installare pacchetti che si installano ma poi non funzionano (come può avvenire con pip);
. il fatto che Conda sia anche un gestore di ambienti di sviluppo ci consente di creare ambienti diversi con diverse edizioni dei pacchetti utilizzati.

4.1 Installazione

Conda si installa insieme a Anaconda o, in maniera molto più rapida, installando Miniconda.

Potremmo definire Miniconda come il germe di Anaconda: basti pensare che nel momento in cui scrivo (aprile 2019) la dimensione dell'installer di Miniconda sta tra i 45 e i 65 MB (a seconda del sistema operativo) e quella dell'installer di Anaconda sta tra i 600 e i 650 MB.

Miniconda contiene solamente il software Conda e il sistema Python di base, compreso IDLE, senza moduli e pacchetti aggiuntivi.

L'installer, per i sistemi operativi Linux, Windows e Mac OS X, lo troviamo su
https://conda.io/en/latest/
scegliendo la pagina Miniconda e possiamo optare per Python 3.7 o Python 2.7 e per sistemi a 32 o 64 bit.
Penso che ormai la versione attuale sia da considerarsi quella per Python 3.7 e, se il sistema che usiamo è tale, a 64 bit.

Sistema Linux Se abbiamo scaricato l'installer per Python3 a 64 bit abbiamo il file Miniconda3-latest-Linux-x86_64.sh nella directory dei file scaricati.

Con il comando a terminale `bash Miniconda3-latest-Linux-x86_64.sh` eseguiamo l'installazione.

Sistema Mac OS X Il file che scarichiamo si chiama Miniconda3-latest-MacOSX-x86_64.sh e il comando a terminale per l'installazione diventa `bash Miniconda3-latest-MacOSX-x86_64.sh`.

Sistema Windows Doppio click sul file installer con estensione .exe.

In ogni caso, durante l'installazione è bene che accettiamo e confermiamo le impostazioni che ci vengono proposte per default e che accettiamo la proposta di inizializzare Miniconda3 nel percorso della nostra home bash. In tal modo con il terminale potremo impartire i comandi a Conda senza entrare nella directory che ospita Miniconda.

Tutti i file che compongono Miniconda, infatti, vengono situati nella directory miniconda3, nella quale verranno installati quelli relativi ai pacchetti che andremo ad aggiungere.

Per eliminare Miniconda dal nostro computer basta cancellare questa directory.

Terminata l'installazione chiudiamo e rilanciamo il terminale e scrivendo il comando
`conda list`
dovremmo vedere l'elenco dei packages installati: se questo accade vuol dire che l'installazione è avvenuta con successo.

Notiamo che nei nomi degli installer ricorre il termine latest, a significare che si tratta dell'installer più recentemente rilasciato.
Per avere un sistema perfettamente funzionante ci dobbiamo preoccupare che anche il sistema operativo su cui andiamo ad installare sia latest o quasi. Per esempio, se vogliamo installare su Ubuntu l'ultima versione di Miniconda, che è la 4.5.12 del gennaio 2019, è bene che non andiamo su un Ubuntu precedente la versione 16.4. Se lavoriamo ancora su un vecchio Windows XP la versione che fa per noi è la 3.10 dell'aprile 2015.
Per procurarci una vecchia versione di Miniconda dobbiamo andare all'indirizzo
https://repo.continuum.io/miniconda/
e scaricare quella di epoca più relazionata all'anzianità del nostro sistema operativo.

4.2 Installazione e disinstallazione dei pacchetti aggiuntivi

I comandi per installare o disinstallare pacchetti sono, rispettivamente
`conda install <nome_pacchetto>`
`conda uninstall <nome_pacchetto>`.

Per arrivare alla dotazione corrispondente a quella creata nei Capitoli 2 e 3 dobbiamo ora dare i comandi

```
conda install numpy
conda install scipy
conda install matplotlib
conda install scikit-learn
conda install pandas
conda install jupyter
conda install nltk
```

Ora possiamo accedere alla shell Python con il comando `python`, alla IDLE con il comando `idle3`, alla shell iPython con il comando `ipython` e al notebook con il comando `ipython notebook` o `jupyter notebook`.

La nostra installazione occuperà ora circa 2,5 GB, praticamente il doppio dello spazio occupato con le installazioni illustrate nei Capitoli 2 e 3 senza l'ausilio di Conda, ma abbiamo sicuramente a disposizione qualche cosa di più affidabile, di più stabile e di più pulito.

4.3 Arricchimento con Spyder

Spyder è un ambiente di sviluppo integrato per Python molto bello e funzionale.

Sarebbe possibile installarlo anche con pip ma il suo funzionamento non sarebbe ottimale: in particolare non funzionerebbe l'help per tutti i pacchetti coinvolti nel progetto.

Se lo installiamo con Conda dà il meglio di sé.

La figura 2 nella pagina seguente mostra la finestra di lavoro di Spyder.

Spyder è molto ben documentato, purtroppo solo in lingua inglese, ma, per chi ha un po' di dimestichezza con la programmazione aiuta molto la sua intuitività.

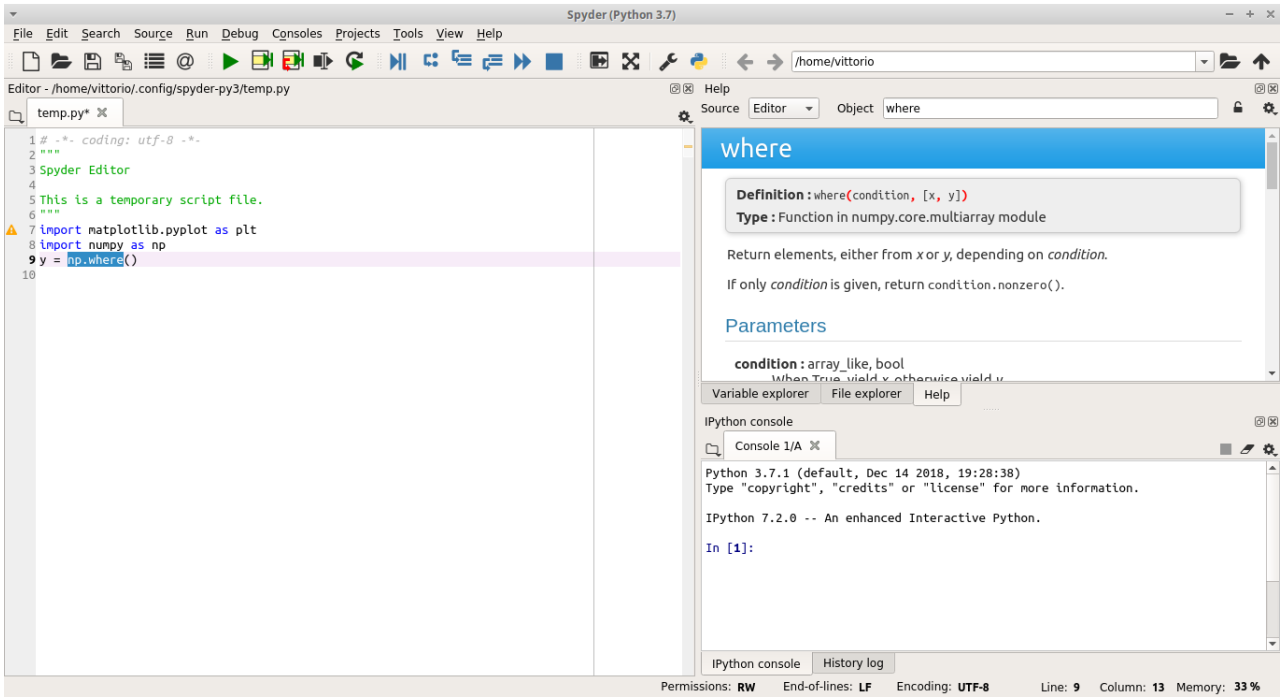

Figura 2: Schermata di lavoro di Spyder

Passando il mouse sulle icone della barra degli strumenti viene descritta la funzione richiamabile.

Nell'illustrazione vediamo l'ampia finestra sulla sinistra dedicata all'editor. Ottima la code completion che ci offre.

Sulla destra in basso abbiamo una finestra dedicata alla shell iPython, utile per vedere un primo risultato dell'esecuzione del nostro programma ma anche, nel durante, per testare l'effetto di certe istruzioni prima di inserirle nell'editor.

Sulla destra in alto abbiamo una finestra dedicata alla visione della documentazione. Per ottenere documentazione su un argomento basta selezionarne il nome nell'editor e premere insieme i tasti CTRL e I.
Nella figura abbiamo la documentazione sulla funzione where del modulo numpy ottenuta premendo Ctrl + i sulla selezione visibile.
Se, per esempio, avessimo selezionato la parola numpy, premendo Ctrl + i avremmo ottenuto la documentazione su numpy.

4.4 Gli environments

Quando installiamo Miniconda creiamo un environment (un ambiente) di base nel quale vengono utilizzate la versione di Python e i pacchetti aggiuntivi più recenti, all'insegna del «latest» di cui ho detto nel precedente paragrafo 4.1.

Possiamo creare altri environments, quanti vogliamo, in cui utilizzare altre versioni di Python e/o altre versioni dei pacchetti aggiuntivi.

Il motivo può essere, per esempio, quello di lavorare su un progetto fatto da altri o da noi in epoca precedente utilizzando una certa versione di Python e una certa versione di Numpy, diverse da quelle nel nostro Miniconda di base.

Un environment si crea con il comando `create` e rispondendo `y` quando Conda ci chiede se vogliamo proprio procedere.

Se vogliamo creare un environment per la versione Python 3.4 e la versione Scipy 0.15 usiamo il comando

```
conda create -n <nome_environment> python=3.4 scipy=0.15.0.
```

Per attivare l'environment usiamo il comando

`conda activate <nome_environment>`
e per disattivarlo e tornare alla base usiamo il comando
`conda deactivate`.

4.5 Miniconda «anacondato»

Se a questo punto lanciamo il comando
`conda install anaconda-navigator`
dotiamo la nostra composizione di software del navigatore di Anaconda e otteniamo praticamente una versione di Anaconda ridotta e contenente solamente i pacchetti che abbiamo scelto noi.

Installato il navigatore lo richiamiamo con il comando
`anaconda-navigator`
e apriamo la finestra riprodotta nella seguente figura 3.

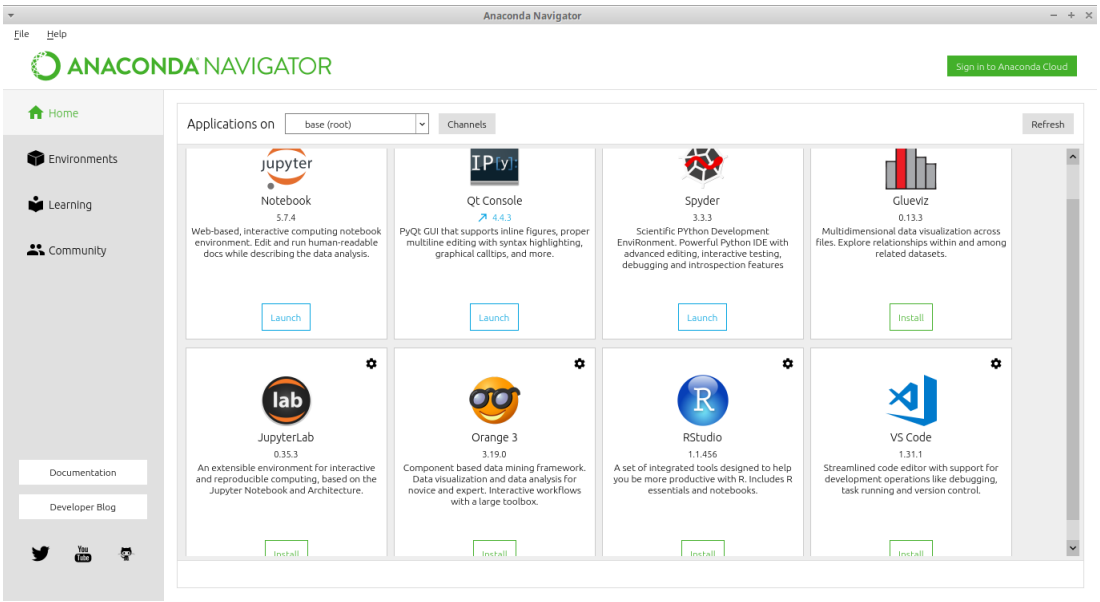

Figura 3: Finestra home di Anaconda Navigator

Quella che vediamo è la finestra home dove abbiamo le icone e la descrizione dei pacchetti principali.
Per i pacchetti installati vediamo un pulsante con scritto LAUNCH, cliccando sul quale lanciamo il pacchetto.
Per i non installati vediamo un pulsante INSTALL, cliccando sul quale installiamo il pacchetto.

Nel nostro caso, viste le installazioni fatte come descritto nei paragrafi 4.1, 4.2 e 4.3, vediamo attivi i pacchetti Jupyter notebook, iPython e Spyder.

Anche le altre finestre funzionano, come se fossimo in Anaconda e non in Miniconda, e le illustrerò nel prossimo Capitolo.

5 Anaconda

Anaconda è una raccolta completa e preconfezionata di pacchetti al servizio non solo del data scientist ma del matematico e dello scienziato in genere.

Installando Anaconda, per esempio, troviamo automaticamente sul nostro computer il modulo Sympy - che potremmo comunque installare anche di nostra iniziativa nei modi visti nei precedenti Capitoli - per lo sviluppo di script di calcolo simbolico che, molto probabilmente, interessano di più a un matematico che a un data scientist. E' questo il modulo con cui ho prodotto lo script che vediamo nella figura 1 nel Capitolo 3.

D'altra parte Anaconda non contiene tutto ciò che è disponibile ma ci offre la possibilità di ulteriori installazioni. Per lo statistico, per esempio, è disponibile l'installatore di R Studio, un ambiente di lavoro che utilizza il linguaggio R in luogo del linguaggio Python.

5.1 Installazione

Se fossimo arrivati ad avere Miniconda secondo quanto visto nel capitolo precedente, compresa l'installazione di anaconda-navigator, potremmo fare il pieno e installare Anaconda risparmiando tempo con il comando a terminale
`conda install anaconda`
e quanto manca per avere Anaconda completo verrebbe installato nella directory Miniconda3.

Per installare Anaconda da zero dobbiamo procurarci l'installer all'indirizzo
https://www.anaconda.com/distribution/
scegliendo opportunamente il sistema operativo.

Sistema Linux Ci vengono proposti gli installer a 64 e a 32 bit per Python 3.7 e per Python 2.7.

Scaricato l'installer ci rechiamo nella cartella dei file scaricati e con il comando a terminale
`bash <nome_file_scaricato>`
eseguiamo l'installazione.

Sistema Mac OS X Ci vengono proposti gli installer per Python 3.7 e per Python 2.7 solo a 64 bit nella forma dell'installatore grafico (file .pkg) e nella forma a riga di comando. In quest'ultimo caso l'installazione si effettua con lo stesso comando visto per il sistema Linux.

Sistema Windows Per Windows ci vengono proposti gli installer per Python 3.7 e per Python 2.7 a 64 e a 32 bit sempre nel formato .exe tipico del sistema Windows.

In ogni caso, durante l'installazione è bene che accettiamo e confermiamo le impostazioni che ci vengono proposte per default e che accettiamo la proposta di inizializzare Anaconda nel percorso della nostra home bash. In tal modo con il terminale potremo impartire i comandi a Anaconda senza entrare nella directory che la ospita.
Se abbiamo installato Anaconda per Python 3 questa directory si chiama anaconda3.

Per verificare che l'installazione sia andata a buon fine, riavviato il terminale dopo l'installazione, possiamo dare il comando `conda list`: se compare l'elenco dei pacchetti installati tutto è andato bene.

5.2 Uso

Una volta installato Anaconda, per utilizzare i pacchetti contenuti nella raccolta abbiamo a disposizione Conda e Anaconda Navigator.

Scrivendo gli appropriati comandi Conda nel terminale avviamo le applicazioni che ci interessano.
Così, scrivendo `python` apriamo una shell Python, scrivendo `ipython` apriamo una shell iPython, scrivendo `jupyter notebook` apriamo il notebook di iPython, scrivendo `spyder` apriamo Spyder, ecc.

Sempre con Conda possiamo gestire gli environments come illustrato nel precedente Capitolo.

Per accedere ai pacchetti di uso più ricorrente abbiamo a disposizione Anaconda Navigator, che, dopo l'installazione di Anaconda, appare come in figura 4 nella pagina seguente.
Come si vede è in tutto simile a quello aperto in Miniconda come da figura 3. Unica differenza il fatto che qui compare come lanciabile anche Jupyter Lab, semplice menu per accedere in altro modo a iPython, iPython notebook, ecc.

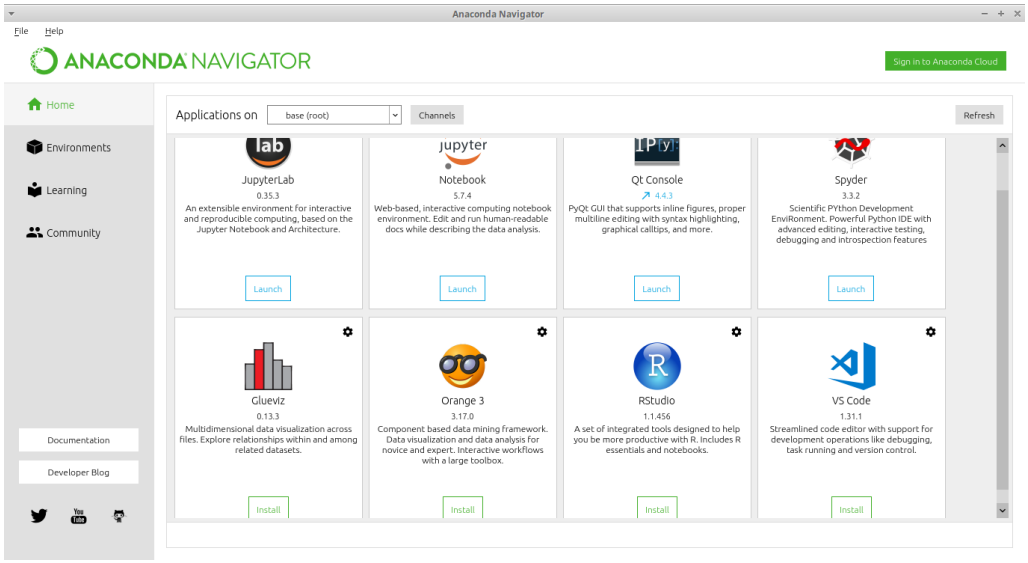

Figura 4: Altra finestra home di Anaconda Navigator

Per aprire anaconda-navigator:
. in Linux scriviamo a terminale il comando `anaconda-navigator`. Qualsiasi linusiano penso sia in grado, se vuole, di costruirsi un launcher, sapendo che l'eseguibile si trova in home/<utente>/anaconda3/bin;
. sul Mac troviamo l'icona di Anaconda Navigator su cui cliccare nel Launchpad;
. in Windows troviamo quanto serve nel menu Start.

Anche in questi ultimi casi vale comunque la possibilità di dare il comando a terminale come per Linux.

5.3 Il mondo di Anaconda

A parte la relativamente importante utilità di fornirci una sorta di menu per avviare i più importanti strumenti di lavoro (ciò che troviamo nella finestra Home illustrata nelle precedenti figure 3 e 4), Anaconda navigator ci consente di amministrare l'ambiente attraverso la finestra che apriamo scegliendo la scheda ENVIRONMENT nel menu verticale sulla sinistra.

La finestra è riprodotta nella seguente figura 5.

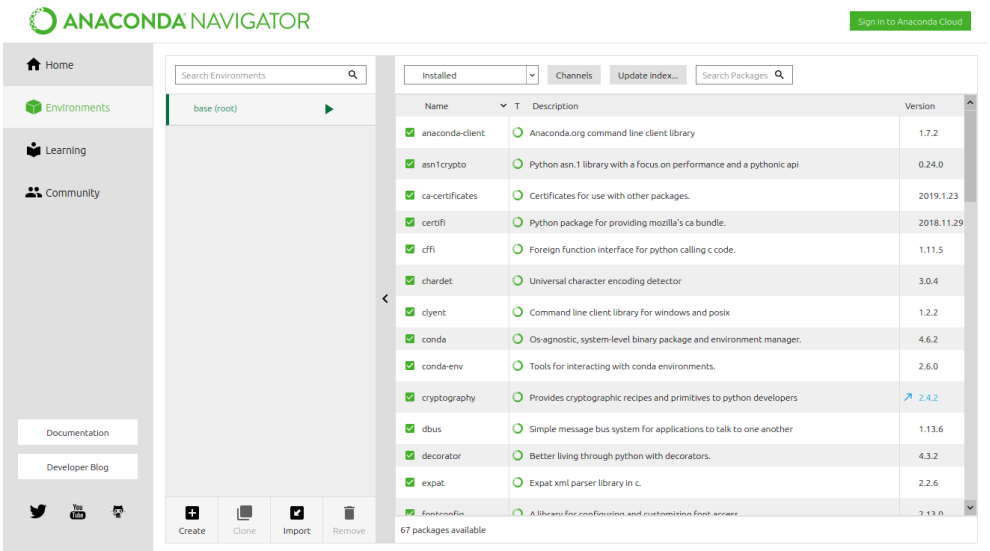

Figura 5: Finestra environments di Anaconda Navigator

Quello che vediamo sulla destra è l'elenco dei pacchetti installati nell'environment base.

Aprendo il menu a discesa che vediamo in alto, dove sta scritto INSTALLED, possiamo scegliere NOT INSTALLED per elencare i pacchetti disponibili e non installati ed agire sull'elenco selezionando quelli che vogliamo eventualmente installare.

Se da questo menu scegliamo UPDATABLE elenchiamo i pacchetti per i quali è disponibile un aggiornamento e selezionando la finestrella VERSION in fondo a destra possiamo lanciare l'aggiornamento.

Tutto ciò può essere riferito ad eventuali environment diversi da quello di base, che, se costruiti, possiamo selezionare nel menu a discesa dove vediamo scritto BASE (ROOT).

Altra finestra molto utile è quella che apriamo scegliendo la scheda LEARNING. In questa, riprodotta nella seguente figura 6, troviamo i link che ci rendono disponibile la migliore documentazione esistente su vari argomenti selezionabili attraverso le relative icone.

Figura 6: Finestra learning di Anaconda navigator

Infine, scegliendo la scheda COMMUNITY, apriamo la finestra riprodotta nella seguente figura 7

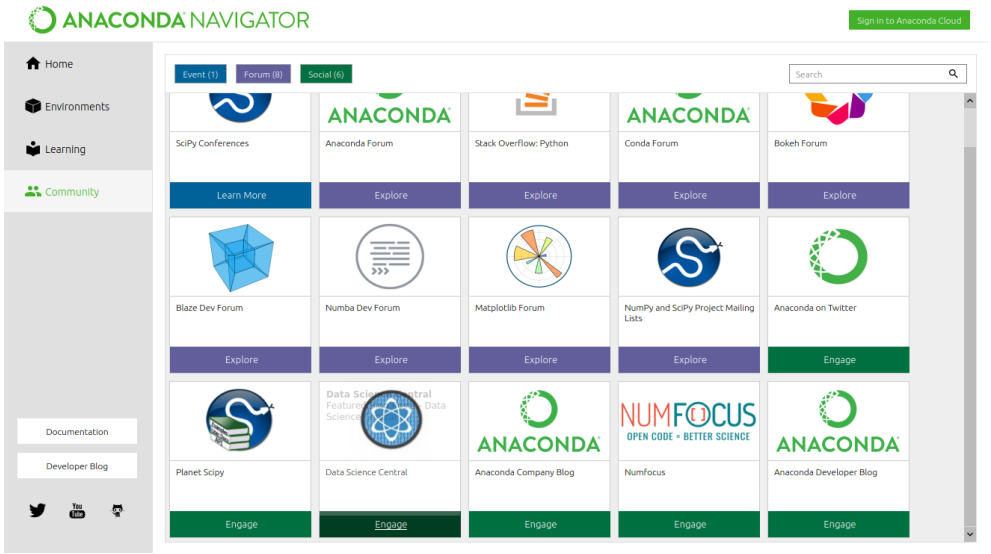

Figura 7: Finestra community di Anaconda navigator

che contiene i link a tutti i forum e gli eventi collegati agli argomenti evidenziati dalle icone.

Ovviamente ogniqualvolta abbiamo a che fare con collegamenti a link, installazioni, aggiornamenti, ecc. dobbiamo essere collegati a Internet.

6 Per saperne di più

La materia al servizio della quale sono stati predisposti gli strumenti che ho illustrato è alquanto complessa e richiede innanzi tutto una preparazione matematico-statistica teorica di base.

Un ottimo testo per acquisire o per consolidare questa preparazione mi pare Lavorare con i big data, La guida completa per il Data Scientist, di Autori Vari (tutti italiani) coordinati da Antonio Teti, edito da Tecniche Nuove. Qui troviamo tutto ciò che viene prima dell'utilizzo degli strumenti.

Ciascun strumento software, poi, ha delle proprie peculiarità di utilizzo che bisogna conoscere.

Per quest'ultima esigenza, almeno per quanto riguarda i pacchetti di uso più ricorrente (numpy, scipy, matplotlib, scikit-learn, pandas, nltk) si trovano in rete, ottime dispense che sono veri e propri manuali, purtroppo quasi sempre in lingua inglese. Notevole, per i primi cinque pacchetti citati, quanto si trova all'indirizzo *https://www.scipy.org/* e, per l'ultimo, quanto si trova all'indirizzo *https://www.nltk.org/*.

Quanto all'applicazione pratica di questi strumenti per la data science posso indicare qualche cosa di utile tradotto in buon italiano o addirittura scritto all'origine in italiano.

Di quest'ultima categoria il testo di Alessandro Cucci - A tu per tu col machine learning edito da thedotcompany.

Tra i tradotti mi pare che il meglio si trovi in Dmitry Zinoviev - Data Science con Python, in Sebastian Raschka - Machine Learning con Python e in Sinan Ozdemir - Data Science, tutti editi da Apogeo.