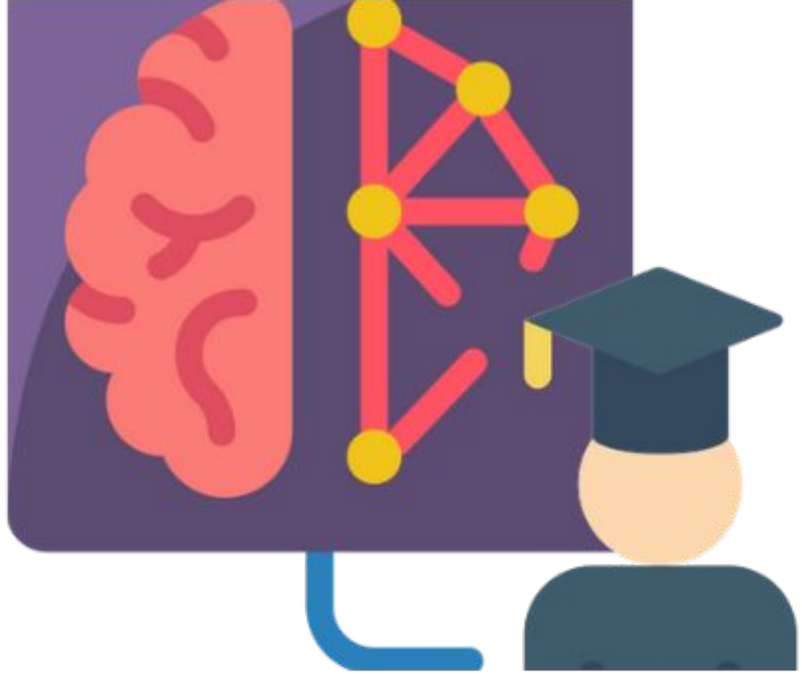
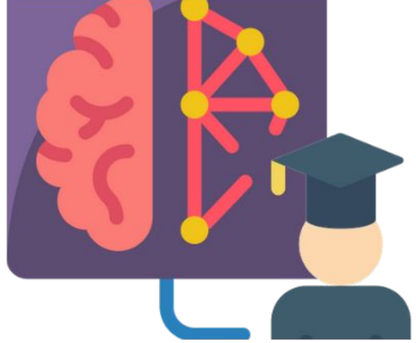
marcusRB



AI-school

Data Science - Big Data Analytics

Presentación Programa



AI-school

Data Science - Big Data Analytics

marcusRB

1. Primeros Pasos
2. Instalación de software
3. GIT

6. Lenguajes de programación
7. R language
8. Python

AI-school

Data Science - Big Data Analytics

Primeros pasos: Equipos

marcusRB

					
Model	Apple 15" MacBook Pro	ASUS ROG Strix GL553VE	Lenovo Yoga 730	HP ZBook F1M37UT#ABA 15.6-Inch Laptop	Acer Aspire VX 15 Gaming Laptop
BLW's Score	95	90	87	85	83
MSRP	from $2,049.99	from $1,699.85	from $655.16	$499.99	$1,899.95
More Information	VIEW ON AMAZON	VIEW ON AMAZON	VIEW ON AMAZON	VIEW ON AMAZON	VIEW ON AMAZON
Processor	Intel i7 2.2 GHz (4.1 GHz Turbo Boost)	Intel Core i7-7700U 2.8 GHz	Intel Core i7-8250U	Intel Core i7 4th Gen 4810MQ 2.8 GHz	Intel Core i7-7700HQ 3.8 GHz
RAM	16GB DDR3	16GB DDR4	8GB DDR4	16GB DDR3	16GB DDR4
Hard Disk	512GB SSD	256GB SSD + 1TB HDD	256GB SSD	256GB SSD	256GB SSD
Display	15" Retina Display (2880 x 1800)	15.6" FHD (1920 x 1080)	15.6 FHD IPS (1920x1080)	15.6" FHD (1920 x 1080)	15.6" Full HD (1920 x 1080)
Graphics	Radeon Pro 560X	NVIDIA GeForce GTX 1050TI	Intel UHD Graphics 620	NVIDIA Quadro K2100M	NVIDIA GeForce GTX 1050 Ti
OS	MacOS	Windows 10	Windows 10	Windows 8	Windows 10
Weight	4.02 pounds	5.5 pounds	2.65 pounds	9.95 pounds	5.5 pounds
Battery Life	10 hours	5 hours	8 hours	6 hours	6 hours

AI-school
Data Science - Big Data Analytics

Primeros pasos

Primeros pasos

open source hardware

open source

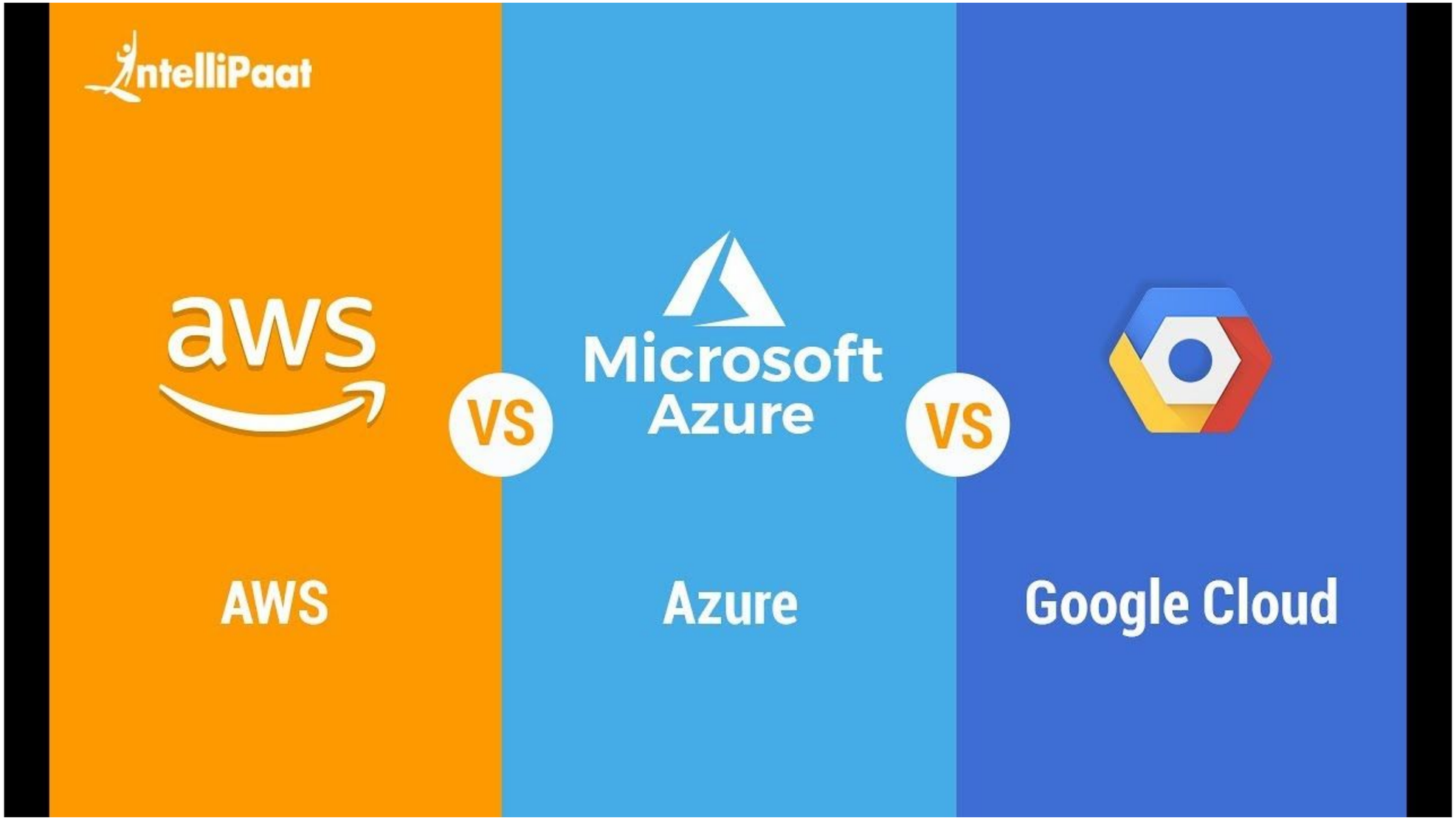
AI-school
Data Science - Big Data Analytics

Primeros pasos

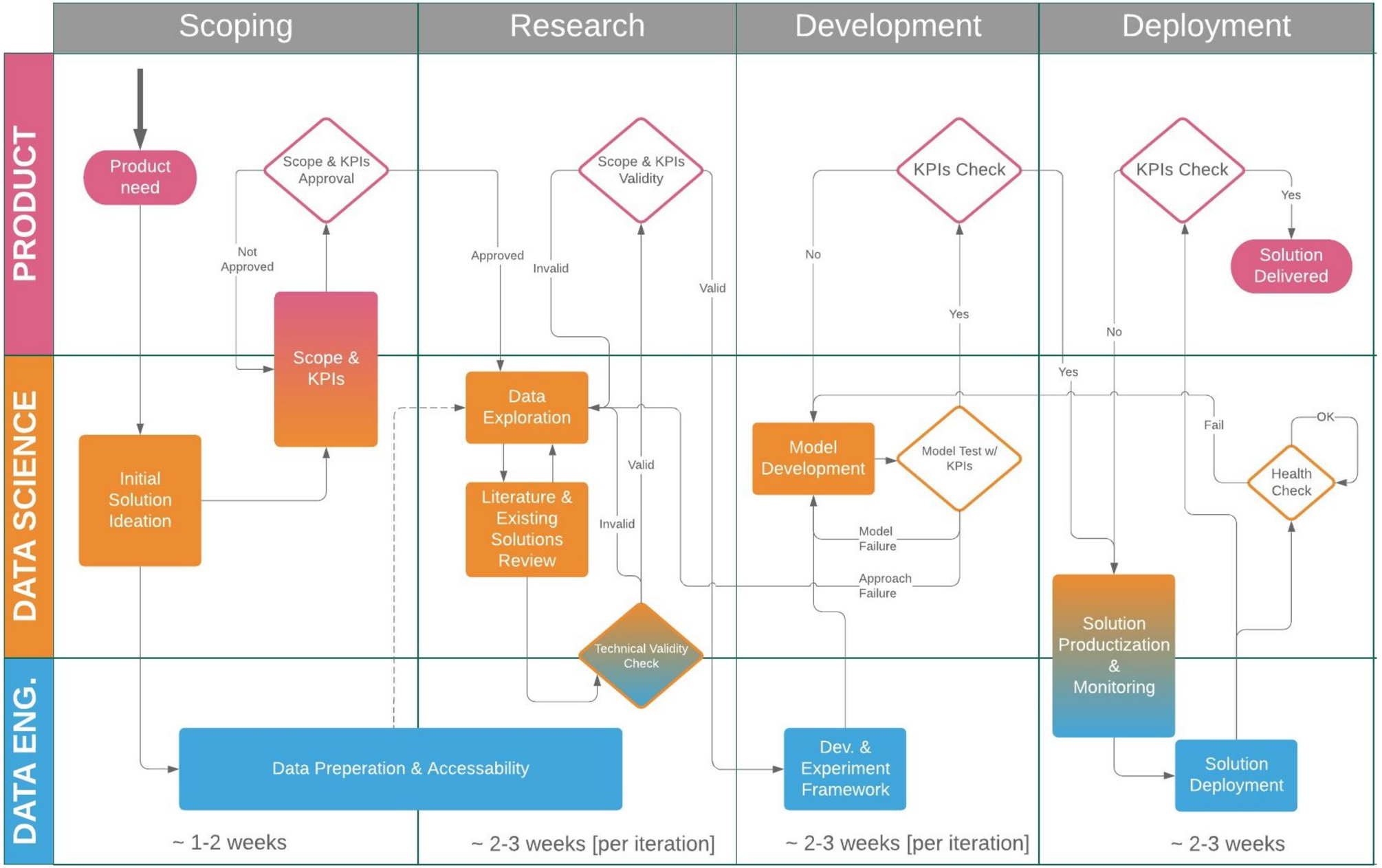
Primeros pasos

Primeros pasos

Primeros pasos



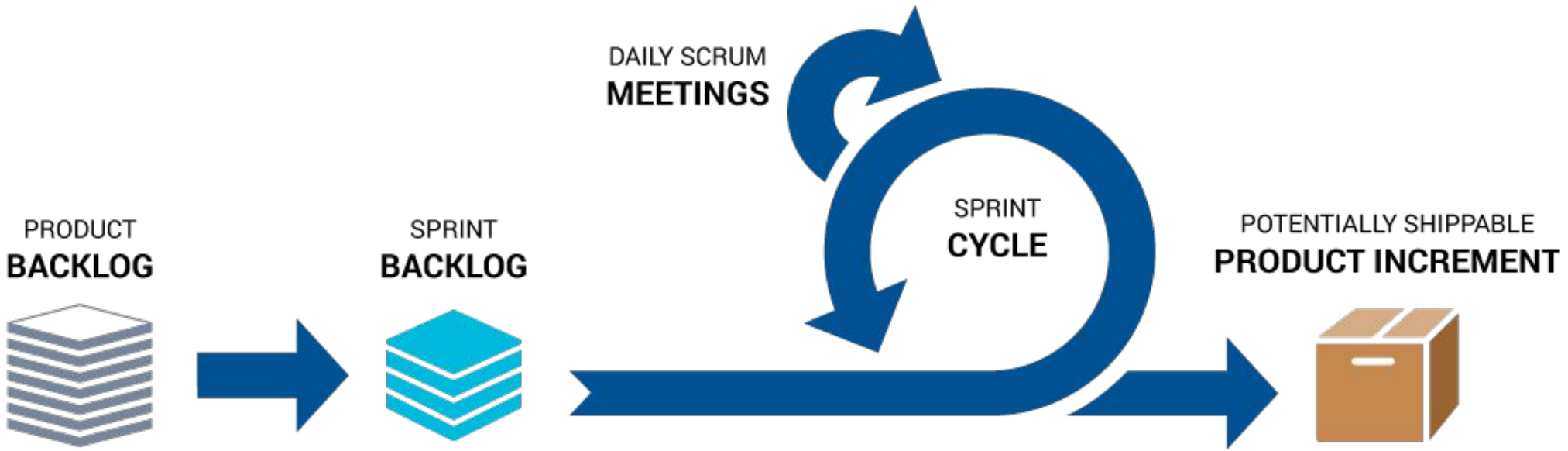
Primeros pasos



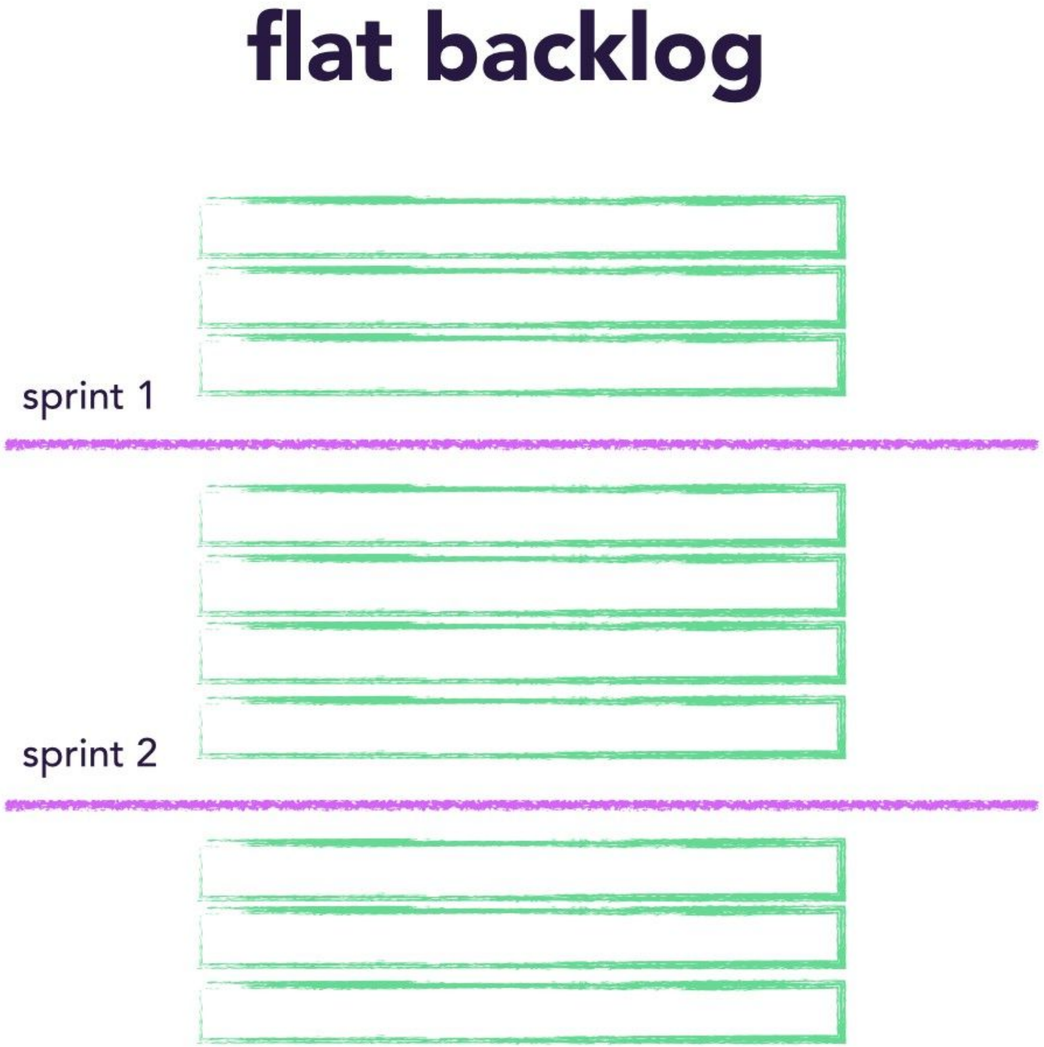
marcusRB

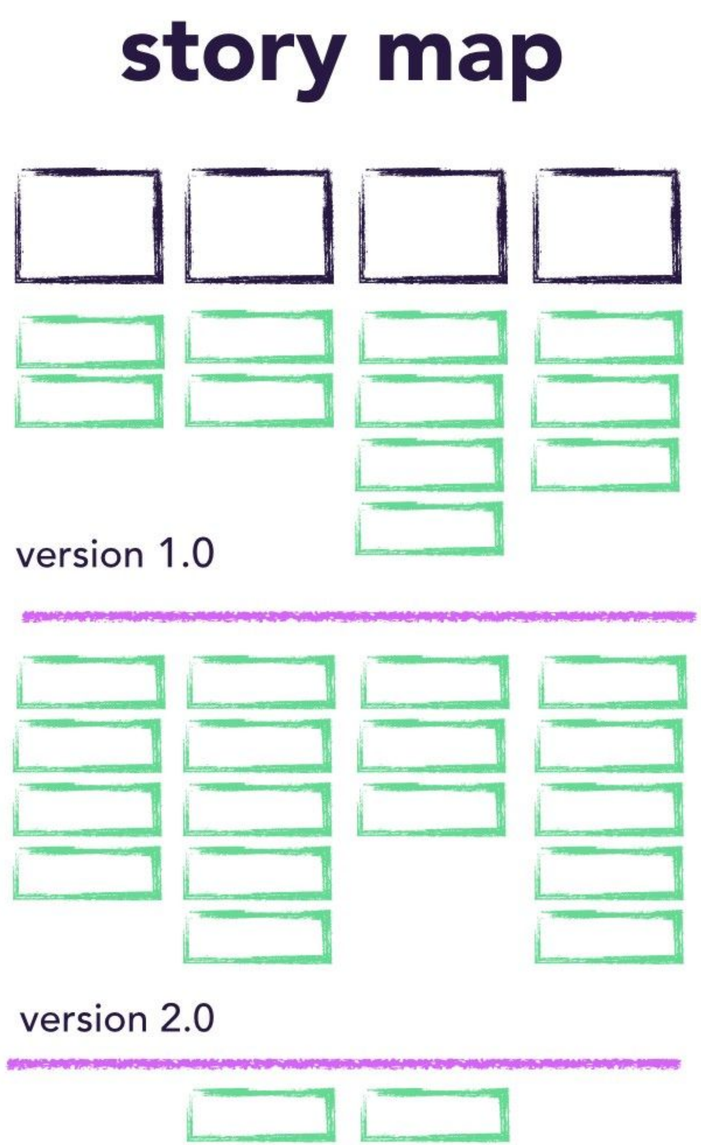
AI-school
Data Science - Big Data Analytics

Primeros pasos

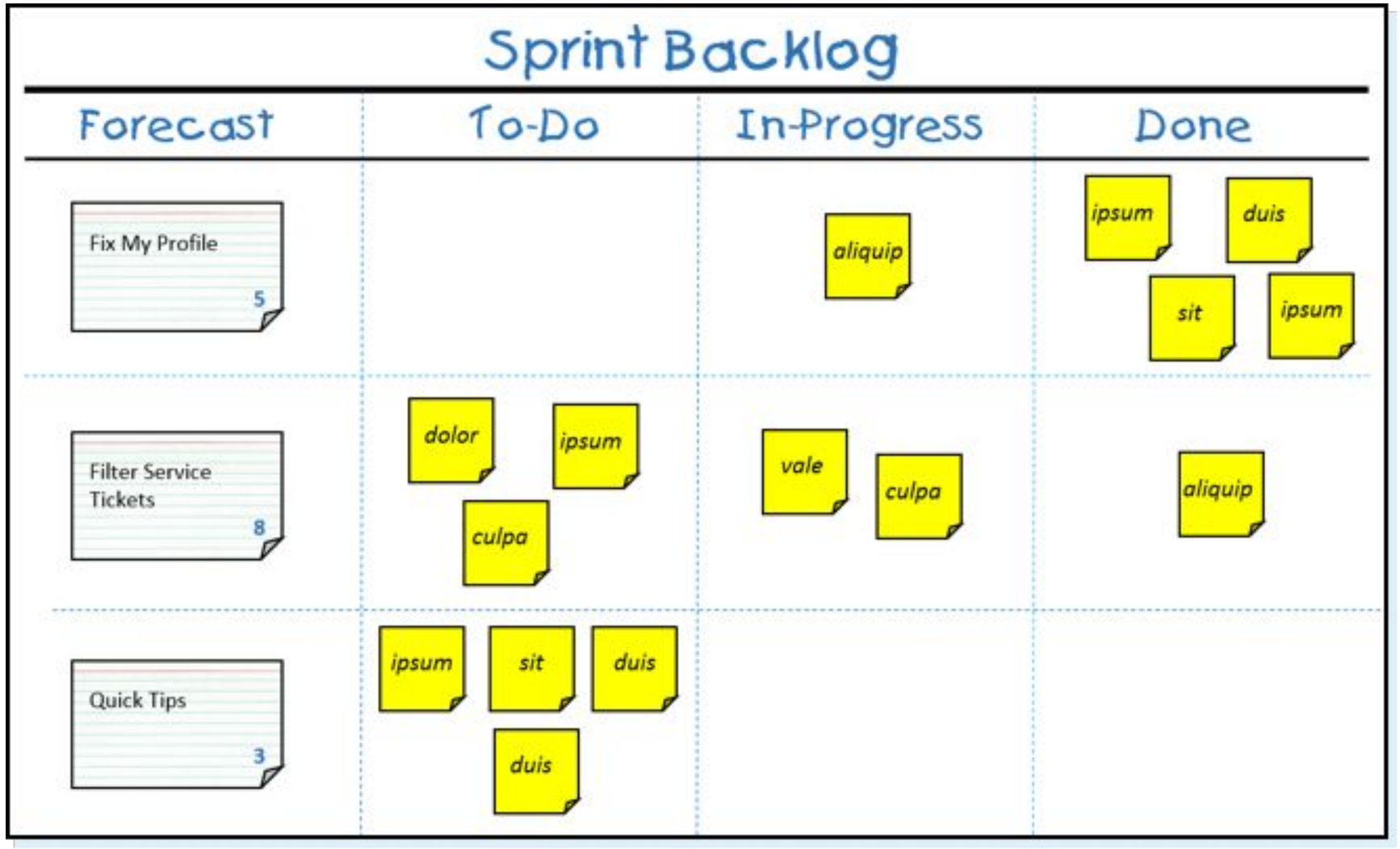


Primeros pasos





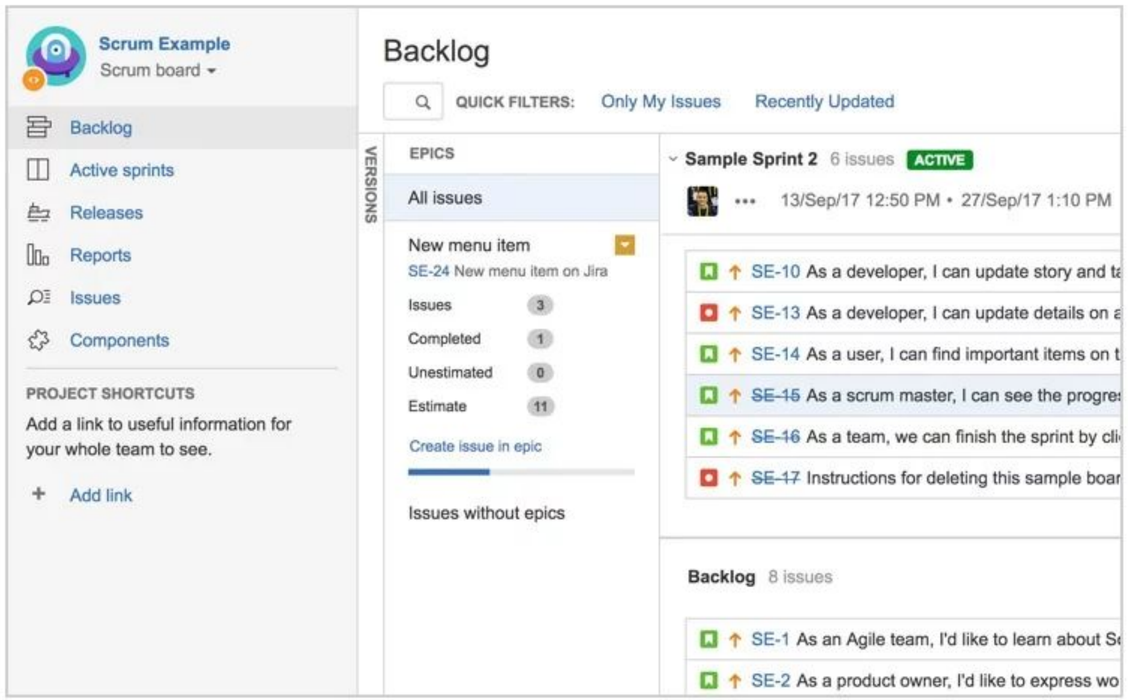
Primeros pasos



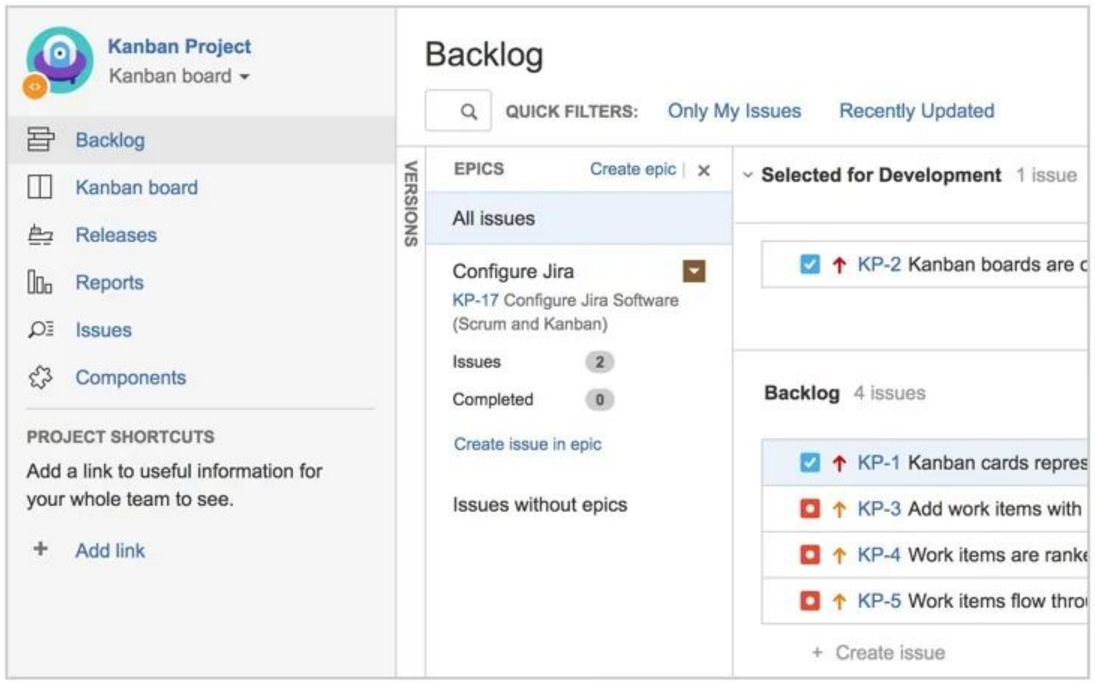
Primeros pasos

marcusRB

Scrum



Kanban

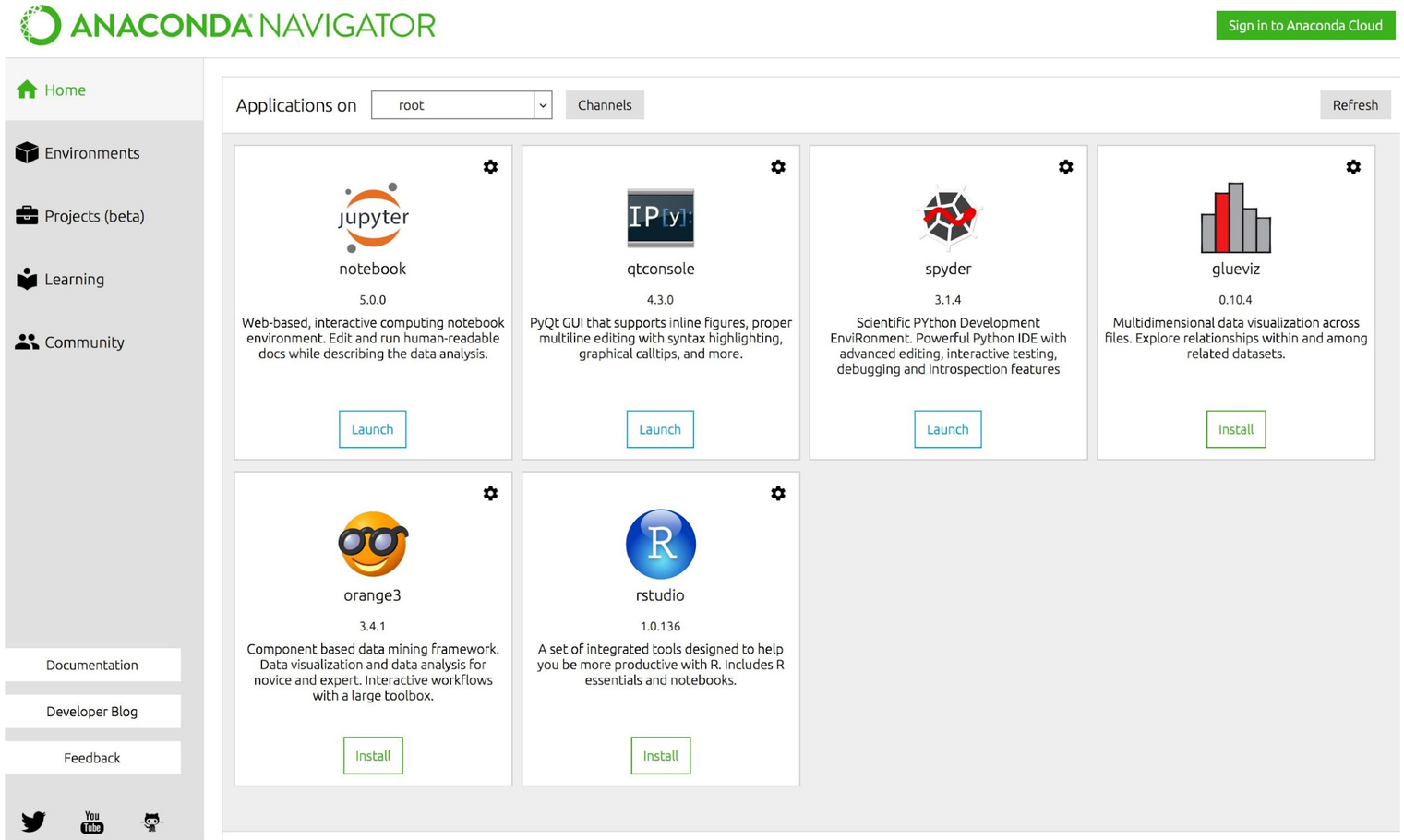


AI-school
Data Science - Big Data Analytics

Instalación software

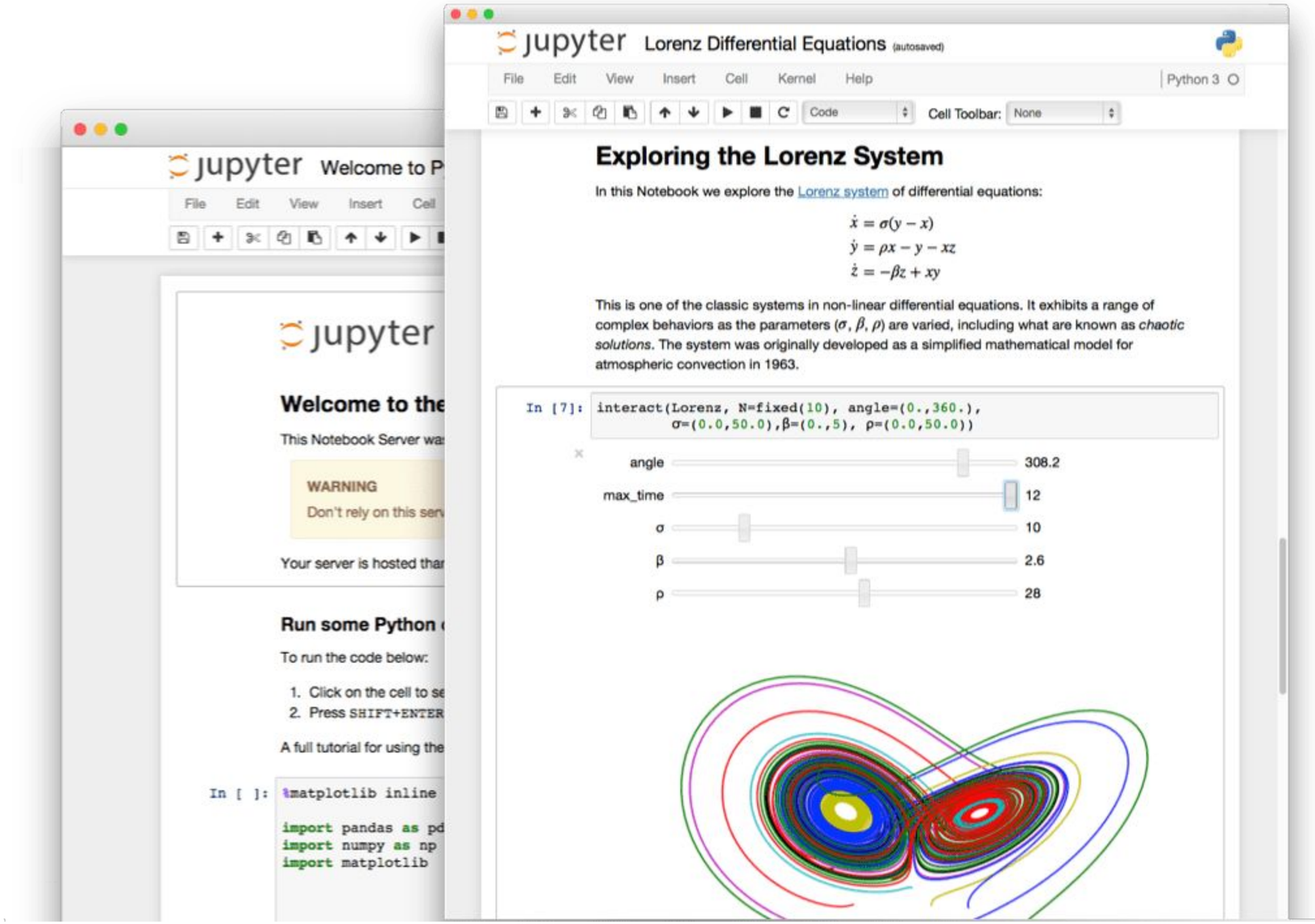
marcusRB

ANACONDA®



AI-school
Data Science - Big Data Analytics

Instalación software



Instalación software

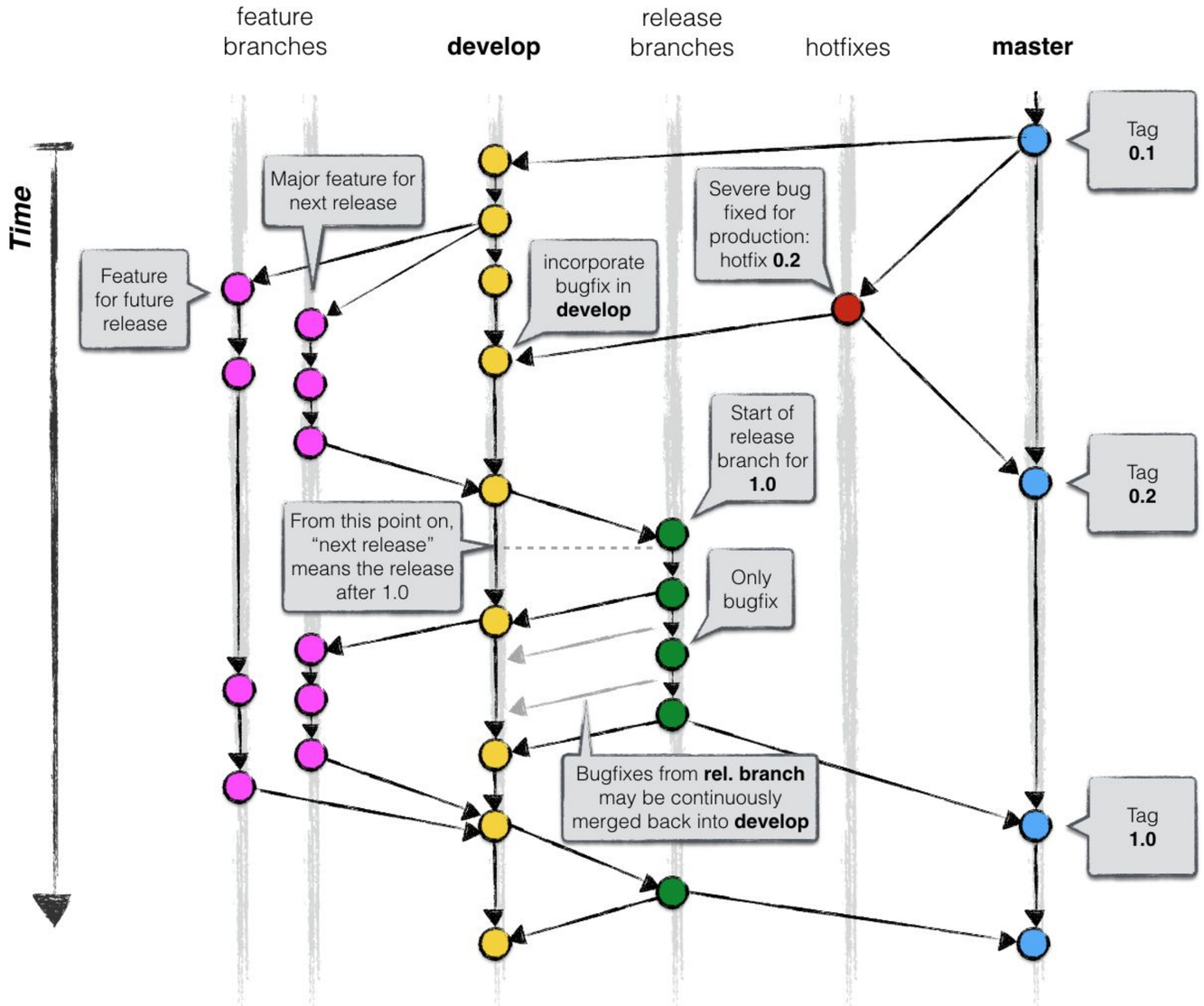
Instalación software

Instalación software

Comparing GitLab Terminology

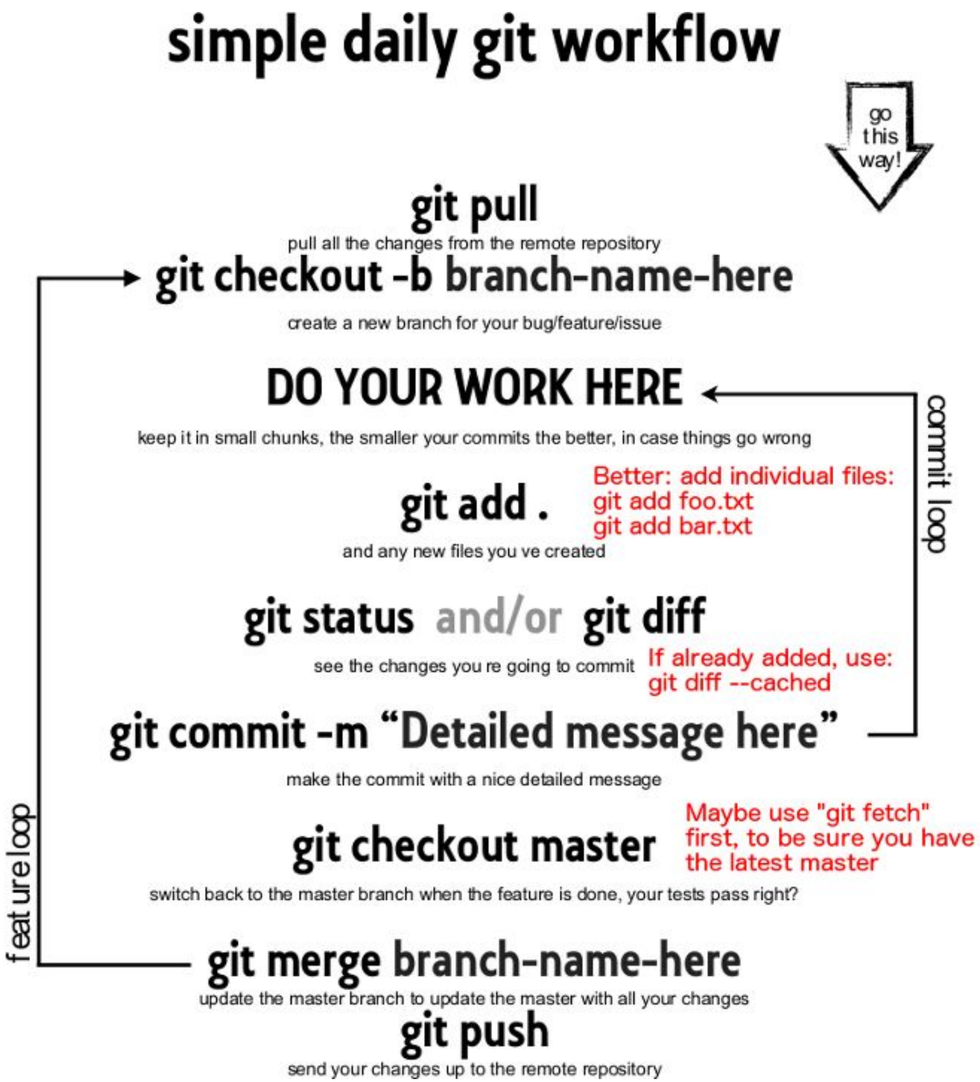
Bitbucket	GitHub	GitLab	So, what does it mean?
Pull Request	Pull Request	**Merge Request**	In GitLab a request to merge a feature branch into the official master is called a Merge Request.
Snippet	Gist	**Snippet**	Share snippets of code. Can be public, internal or private.
Repository	Repository	**Project**	In GitLab a Project is a container including the Git repository, discussions, attachments, project-specific settings, etc.
Teams	Organizations	**Groups**	In GitLab, you add projects to groups to allow for group-level management. Users can be added to groups and can manage group-wide notifications.

GIT

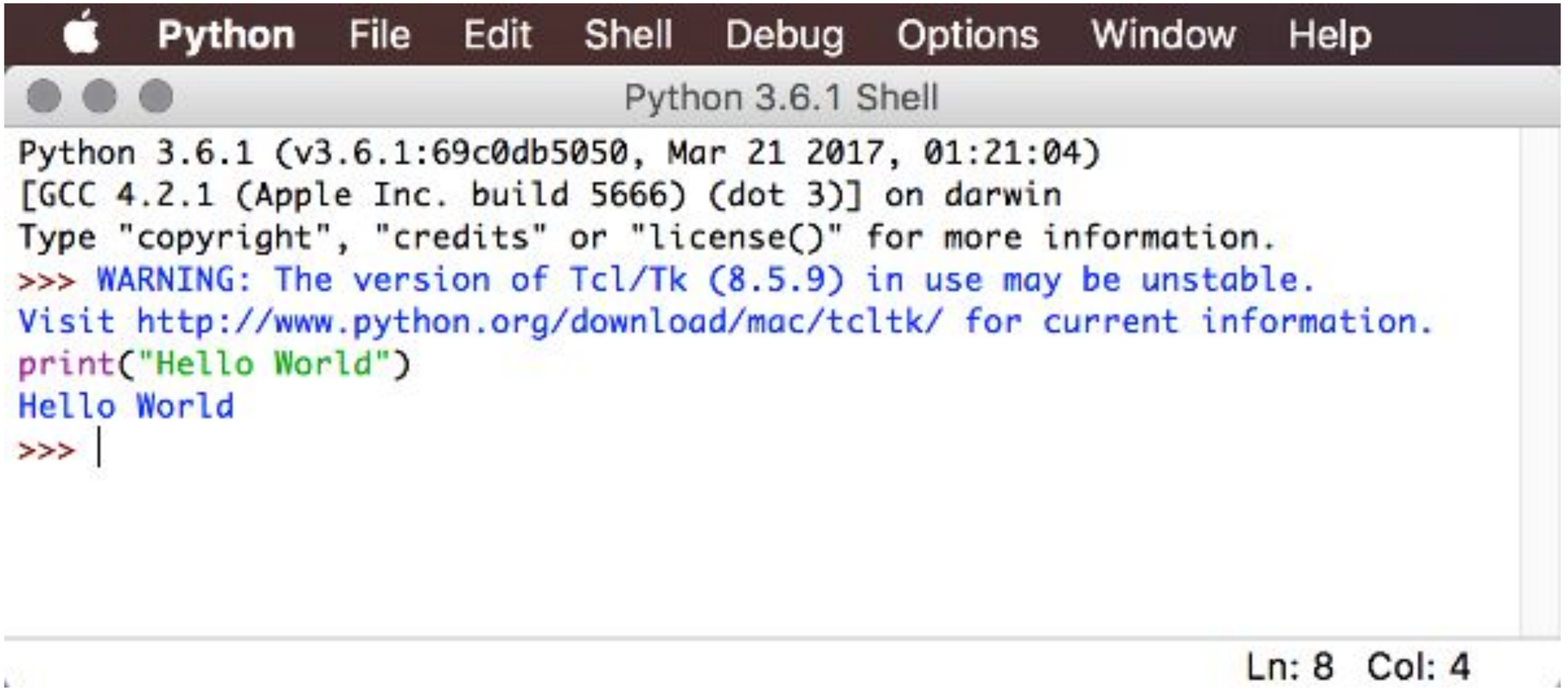


https://rogerdudler.github.io/git-guide/index.es.html

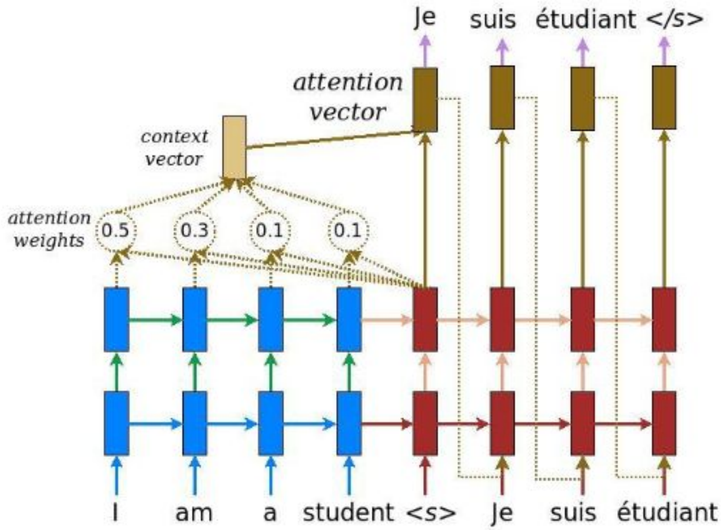
GIT


simple daily git workflow
go this way!
git pull
pull all the changes from the remote repository
git checkout -b branch-name-here
create a new branch for your bug/feature/issue
DO YOUR WORK HERE
keep it in small chunks, the smaller your commits the better, in case things go wrong
git add .
Better: add individual files:
git add foo.txt
git add bar.txt
and any new files you ve created
git status and/or git diff
see the changes you re going to commit
If already added, use:
git diff --cached
git commit -m "Detailed message here"
make the commit with a nice detailed message
commit loop
git checkout master
Maybe use "git fetch" first, to be sure you have the latest master
switch back to the master branch when the feature is done, your tests pass right?
feature loop
git merge branch-name-here
update the master branch to update the master with all your changes
git push
send your changes up to the remote repository

from ‘Hello World’...



... to Artificial Intelligence



The input is put through an encoder model which gives us the encoder output of shape *(batch_size,* hidden state of shape *(batch_size, hidden_size)*.

Here are the equations that are implemented:

$$\alpha_{ts} = \frac{\exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s)\right)}{\sum_{s'=1}^{S} \exp\left(\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_{s'})\right)} \quad \text{[Attention weights]}$$

$$\boldsymbol{c}_t = \sum_s \alpha_{ts} \bar{\boldsymbol{h}}_s \quad \text{[Context vector]}$$

$$\boldsymbol{a}_t = f(\boldsymbol{c}_t, \boldsymbol{h}_t) = \tanh(\boldsymbol{W}_c[\boldsymbol{c}_t; \boldsymbol{h}_t]) \quad \text{[Attention vector]}$$

$$\text{score}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) = \begin{cases} \boldsymbol{h}_t^\top \boldsymbol{W} \bar{\boldsymbol{h}}_s & \text{[Luong's multiplicative style]} \\ \boldsymbol{v}_a^\top \tanh\left(\boldsymbol{W_1} \boldsymbol{h}_t + \boldsymbol{W_2} \bar{\boldsymbol{h}}_s\right) & \text{[Bahdanau's additive style]} \end{cases}$$

```
    print("Attention weights shape: (batch_size, sequence_length, 1) {}".format(attention_weights.shape))
```

```
[ ] class Decoder(tf.keras.Model):
      def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = tf.keras.layers.GRU(self.dec_units,
                                       return_sequences=True,
                                       return_state=True,
                                       recurrent_initializer='glorot_uniform')
        self.fc = tf.keras.layers.Dense(vocab_size)

        # used for attention
        self.attention = BahdanauAttention(self.dec_units)

      def call(self, x, hidden, enc_output):
        # enc_output shape == (batch_size, max_length, hidden_size)
        context_vector, attention_weights = self.attention(hidden, enc_output)

        # x shape after passing through embedding == (batch_size, 1, embedding_dim)
        x = self.embedding(x)

        # x shape after concatenation == (batch_size, 1, embedding_dim + hidden_size)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        # passing the concatenated vector to the GRU
        output, state = self.gru(x)

        # output shape == (batch_size * 1, hidden_size)
        output = tf.reshape(output, (-1, output.shape[2]))

        # output shape == (batch_size, vocab)
        x = self.fc(output)

        return x, state, attention_weights
```

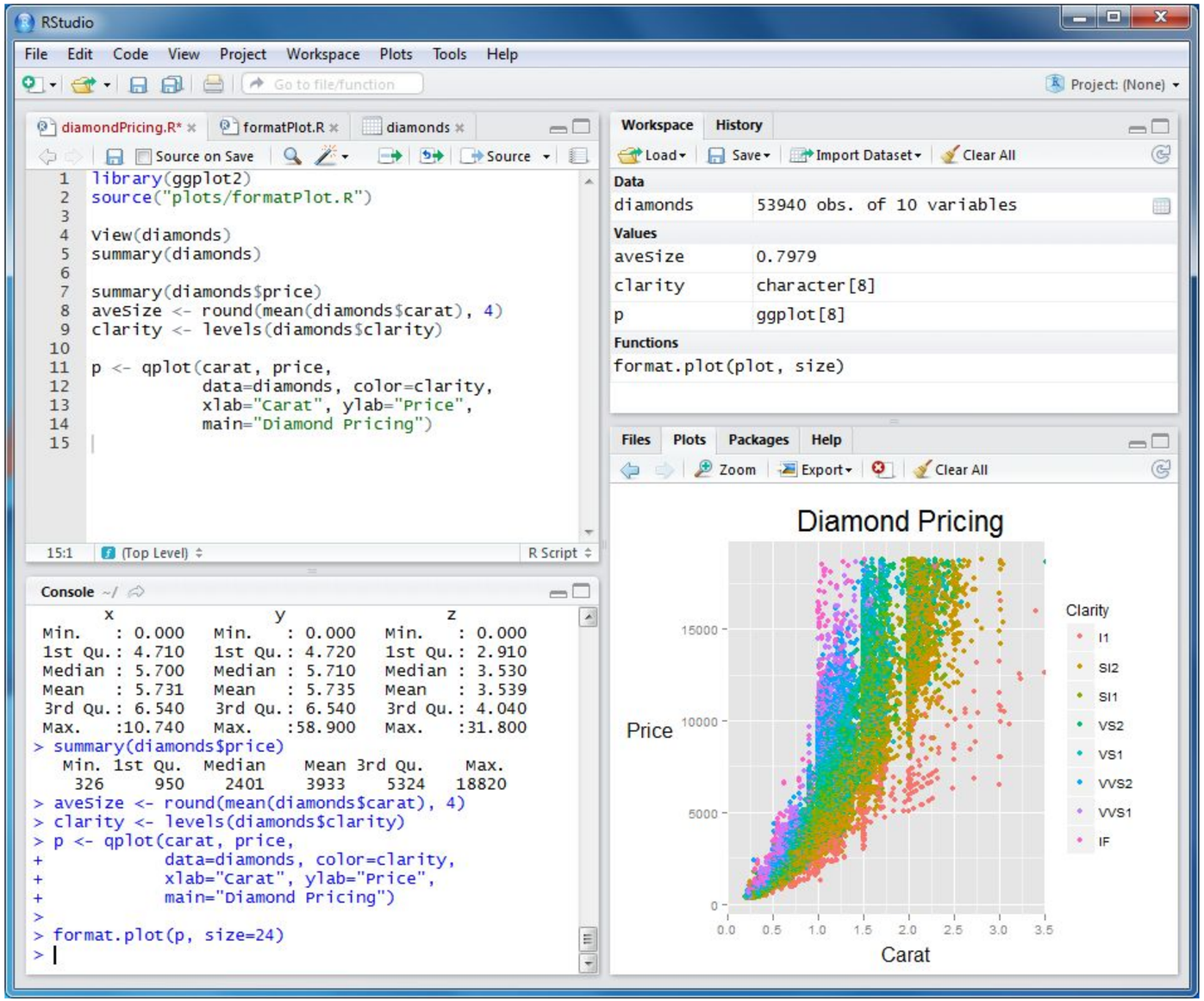
```
[ ] decoder = Decoder(vocab_tar_size, embedding_dim, units, BATCH_S

    sample_decoder_output, _, _ = decoder(tf.random.uniform((64, 1)),
                                          sample_hidden, sample_output
```

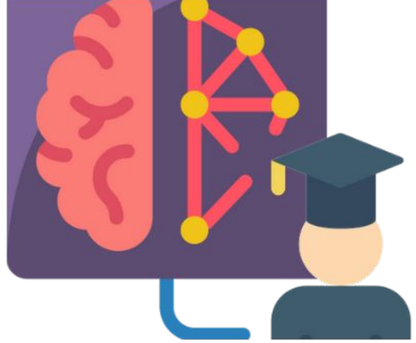
Instalación software

marcusRB

RStudio®



AI-school
Data Science - Big Data Analytics



AI-school

Data Science - Big Data Analytics

marcusRB

¿Dudas - Preguntas?

AI-school

Data Science - Big Data Analytics