Name: <Process to deploy a prediction using cloud and API heroku>

Report date: <01/09/2021>

Internship Batch:<LISUM02>

Version:<1.0>

Data intake by:<Igor Azevedo de Queiroz>

Data intake reviewer:< >

Data storage location: <https://github.com/IgorQueiroz23031988/Flask_Deployment>

Process to Deploy a Prediction Using Cloud and API Heroku

Here will be explained the process of deploy a prediction using flask step by step.

To reach the final step of prediction, of course the dataset has to pass through other process, those process will be explained here, however in a very succinct way, due the main objection of this article is to demonstrate just the deploy method.

Dataset

The data used was collected on kaggle website: <https://www.kaggle.com/c/rossmann-store-sales>.

Rossmann operates over 3,000 drug stores in 7 European countries. Currently, Rossmann store managers are tasked with predicting their daily sales for up to six weeks in advance. Store sales are influenced by many factors, including promotions, competition, school and state holidays, seasonality, and locality. With thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results can be quite varied.

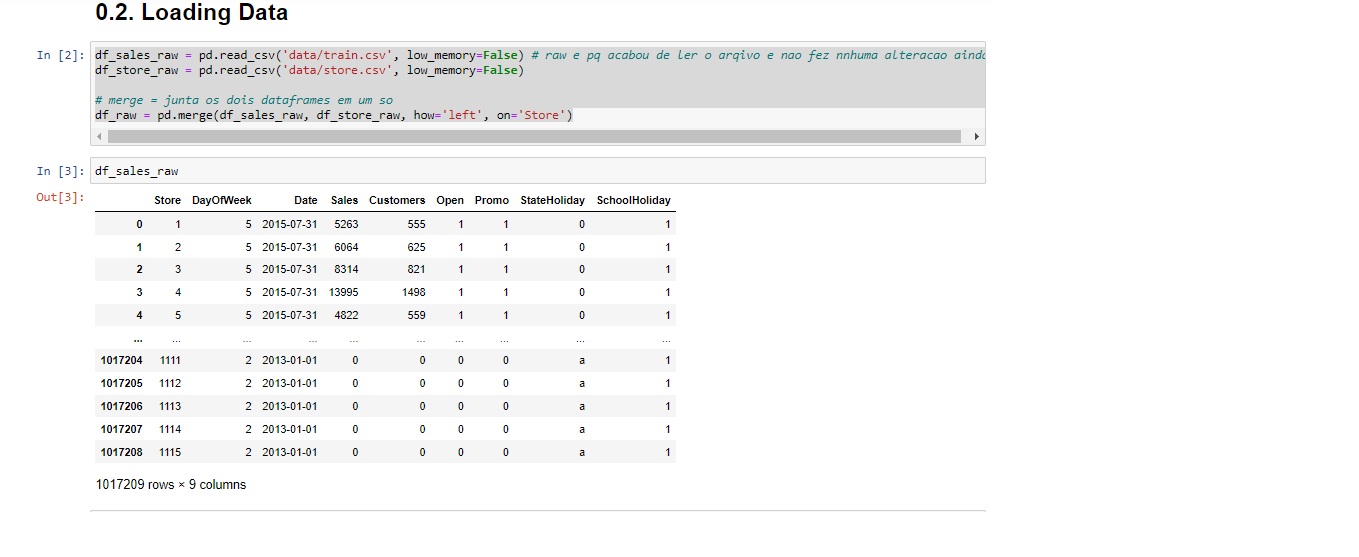
So, the main objective here is making a prediction of sales for 6 weeks in advance.

Loading Dataset

The data is shared in three datasets: train.csv, store.csv and test.csv.

In order the prepare data and test the machine learning algorithm, first will be use train.csv and store.csv. After all steps to prepare data, will be used test.csv and store.csv. Here is where the explanation about deploy using flask is demonstrated.

Now first step is merge the train.csv and store.csv.



After this step, a sequence of process is made in order to have a property dataset to apply the ML model, such as:

* Data description: to understand the attributes;
* Check and fill out NA: to identify missing values and fulfill them based on business understanding;
* Change attributes types: to does not happens errors when make the prediction;
* Feature engineering: to create new attributes that helps the prediction and create hypotheses, to help the CEO to take decisions;
* EDA: to verify which attribute is important and which it is not for the prediction;
* Feature selection: another method to identify which attribute is important and which it is not for the prediction, here is used Boruta to do this task.
* Split data and ML modeling: here data is split in training and test data, and then different ML models are tested to identify which one is the best for this dataset. To improve these results, cross validation is used.
* Hyperparameter fine tuning: after identify the best model, this process is used to identify the best parameters for this model.

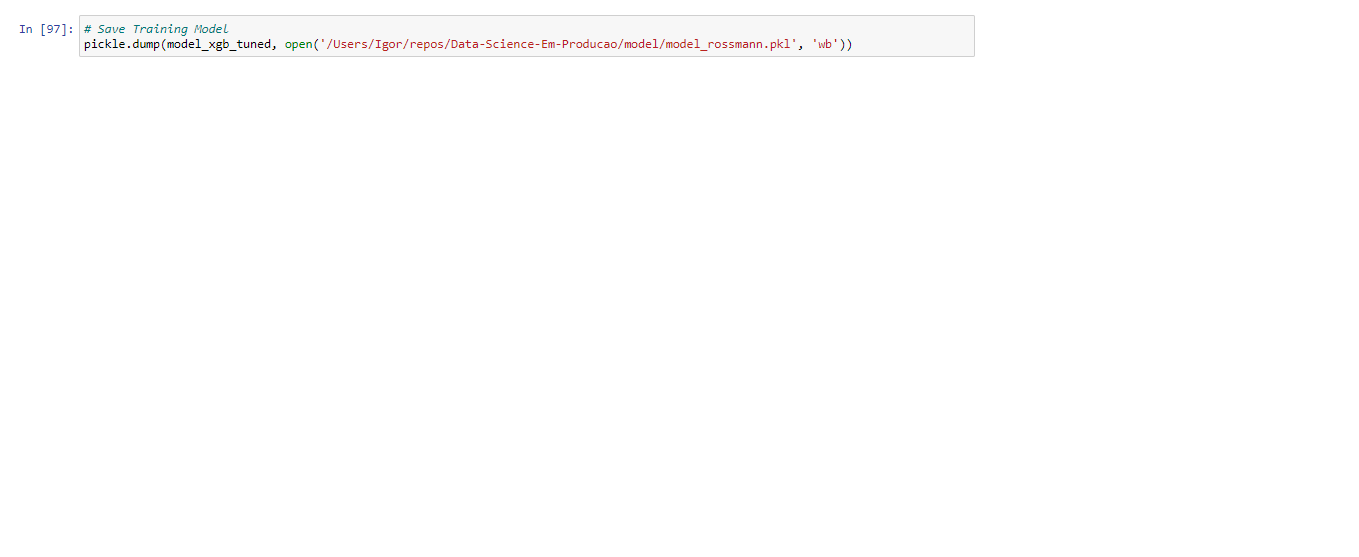
Final Model

After all this process, it was identified that Xgboost is the best ML model.



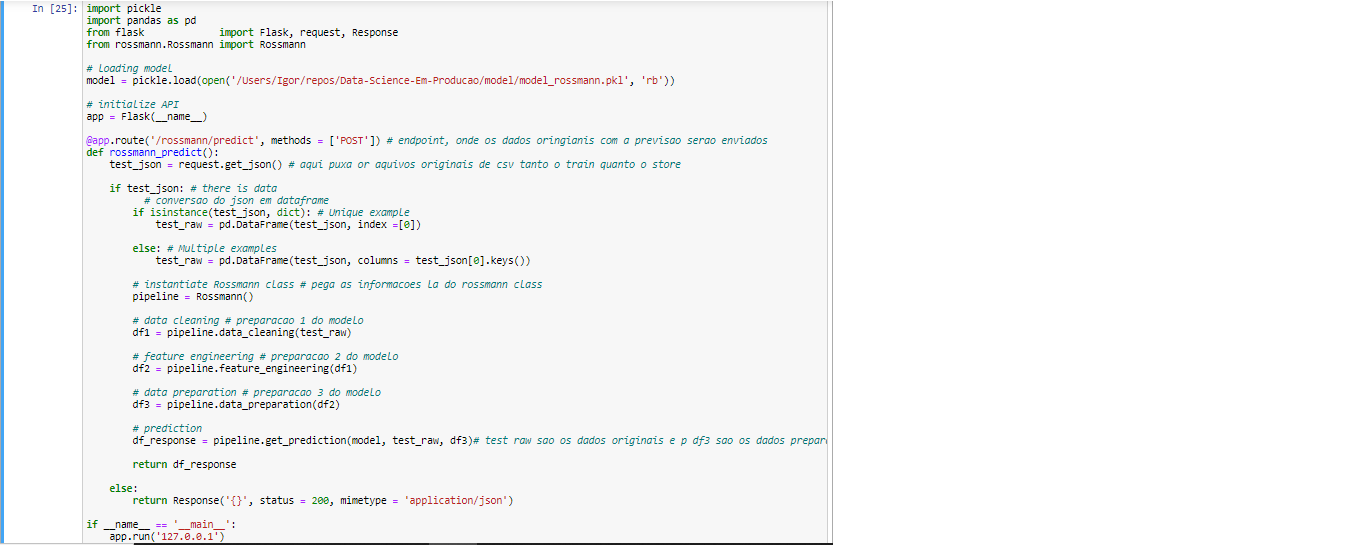
Deploy Model to Production

Finally here it is the process of model deployment using flask.

First the model is saved. 

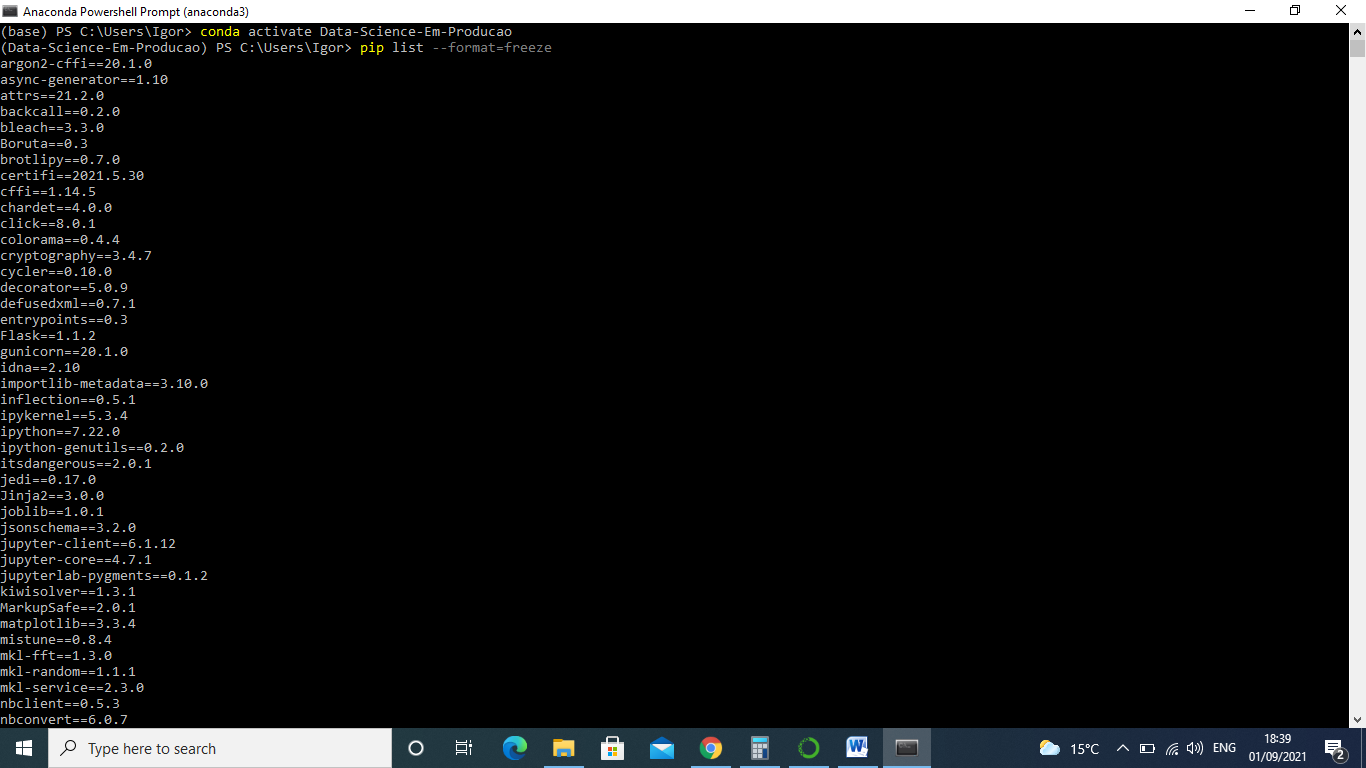
Creating Flask API

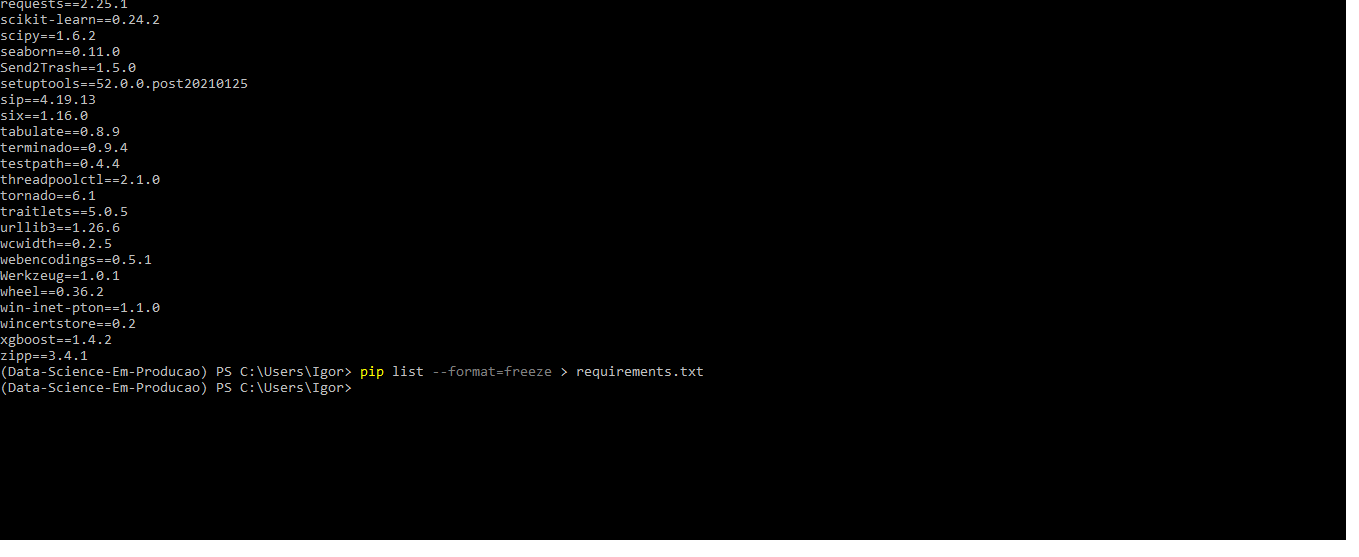
Now the API using flask is created to make the deploy



Creating Requirements.txt

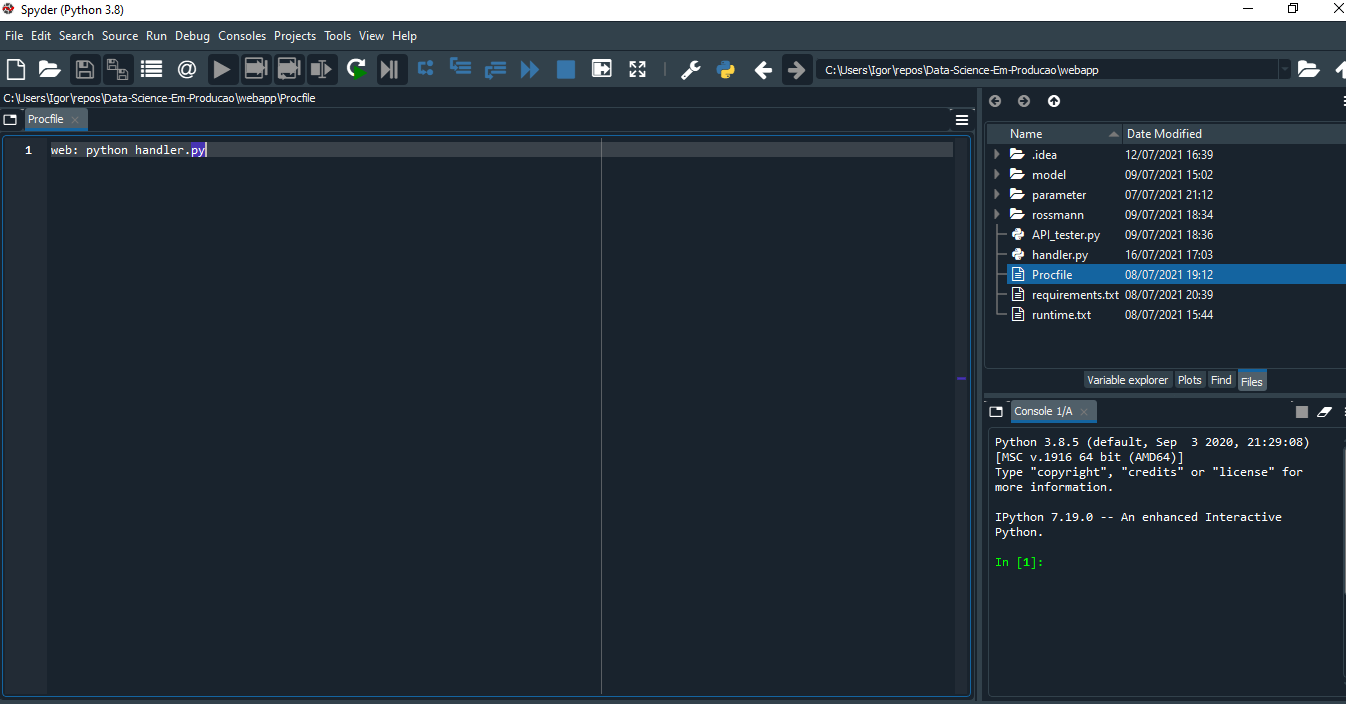
In order to upload the API to heroku, it is necessary estipulate all libraries used by this modeling. So it is copy all those libraries into a file called requirements.txt.





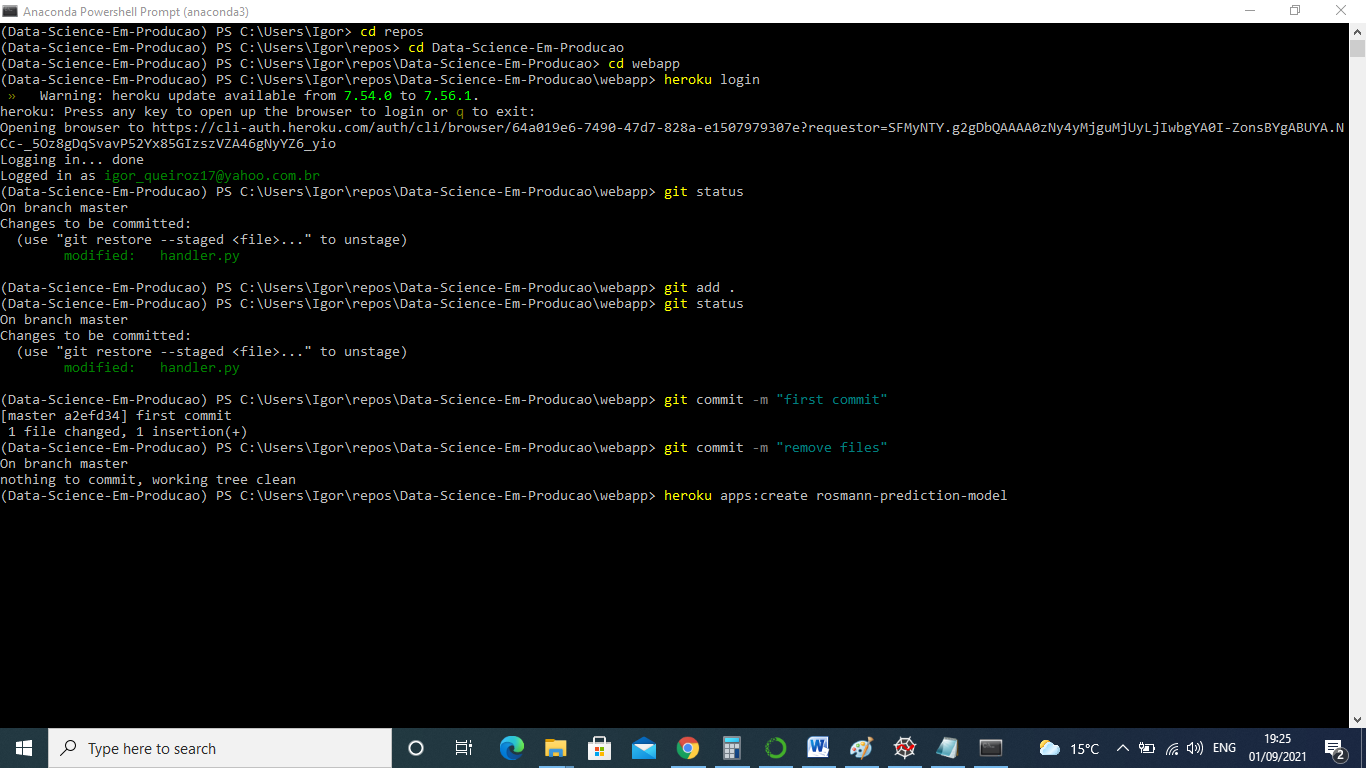
Creating Procfile

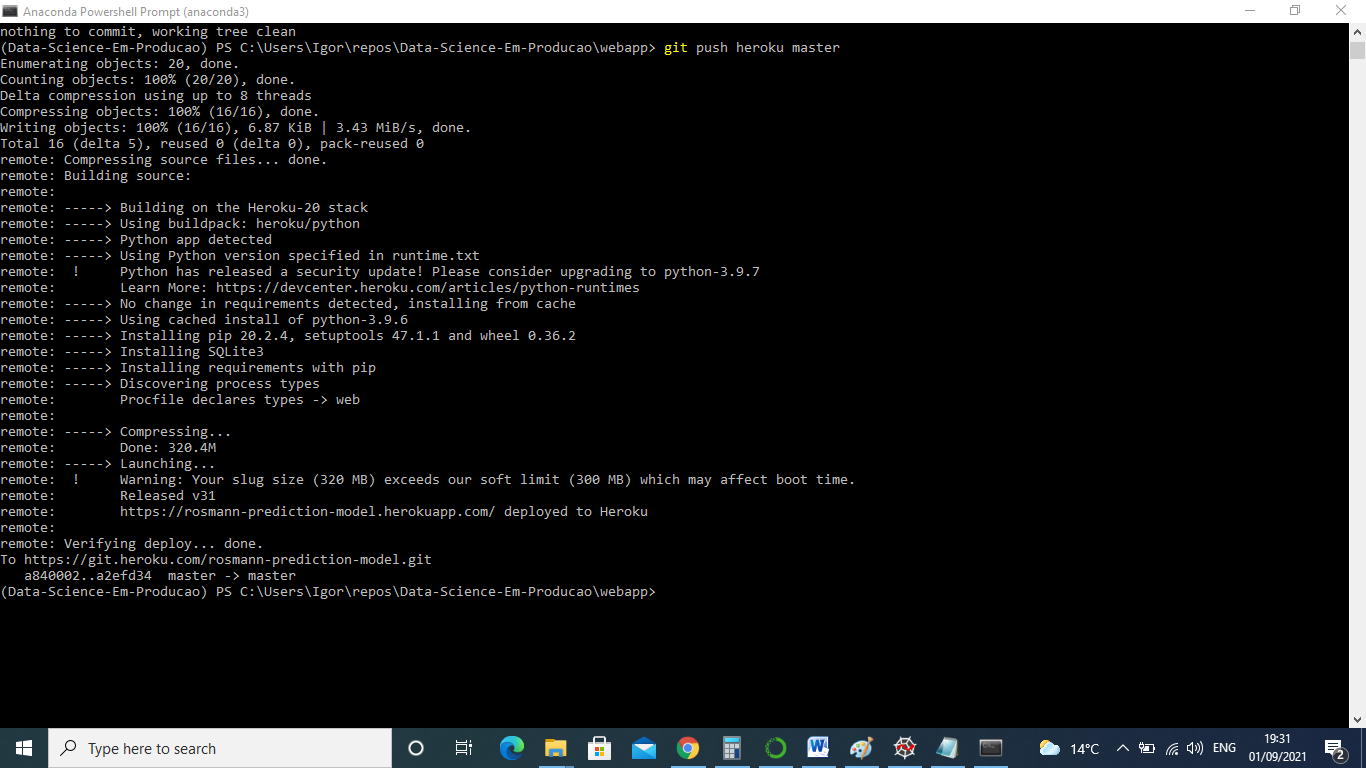
Now to let the app functional it is necessary create a file called procfile.



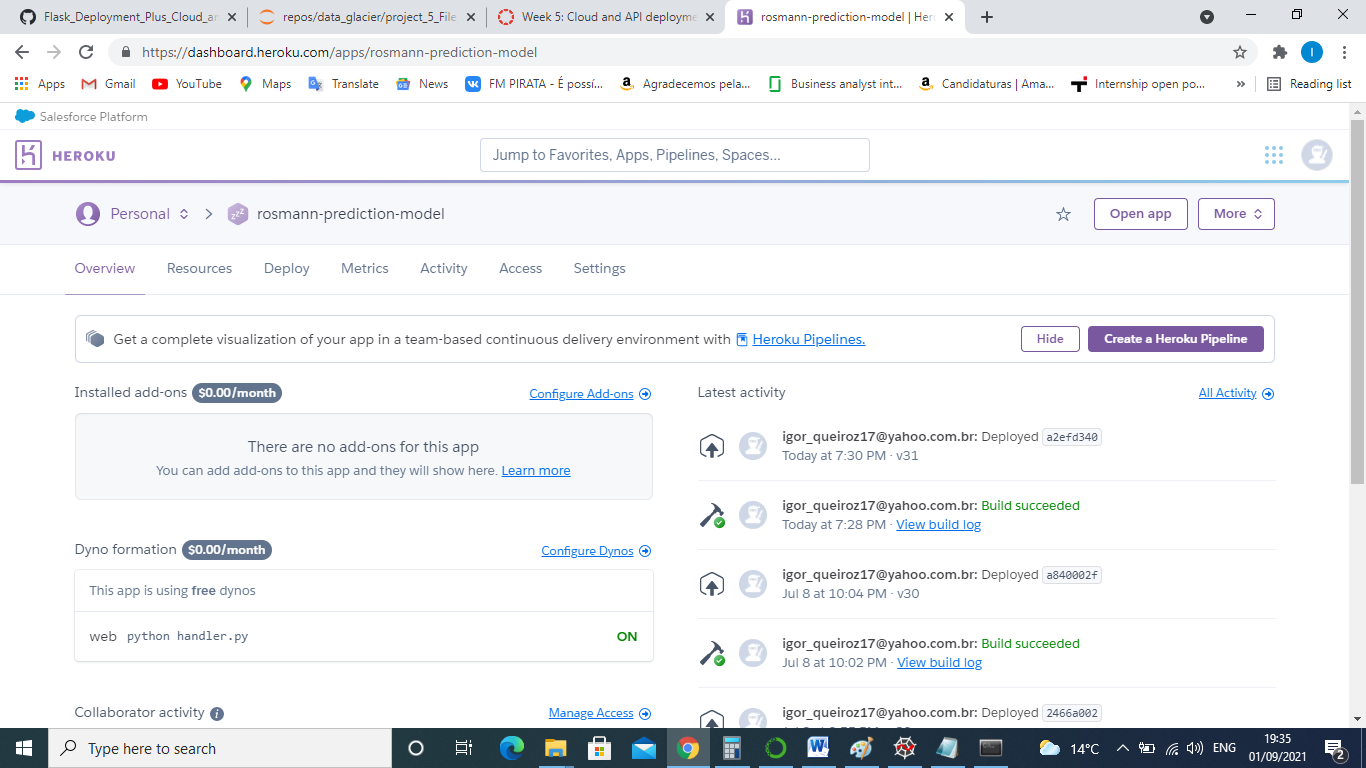
Uploading model to Heroku

To upload the model at heroku it is necessary access the CMD and set a sequence of commands showed below on the picture. All the files, such as: the machine learning model, handler API, procfile and requirements.txt are uploaded to heroku website as webapp.



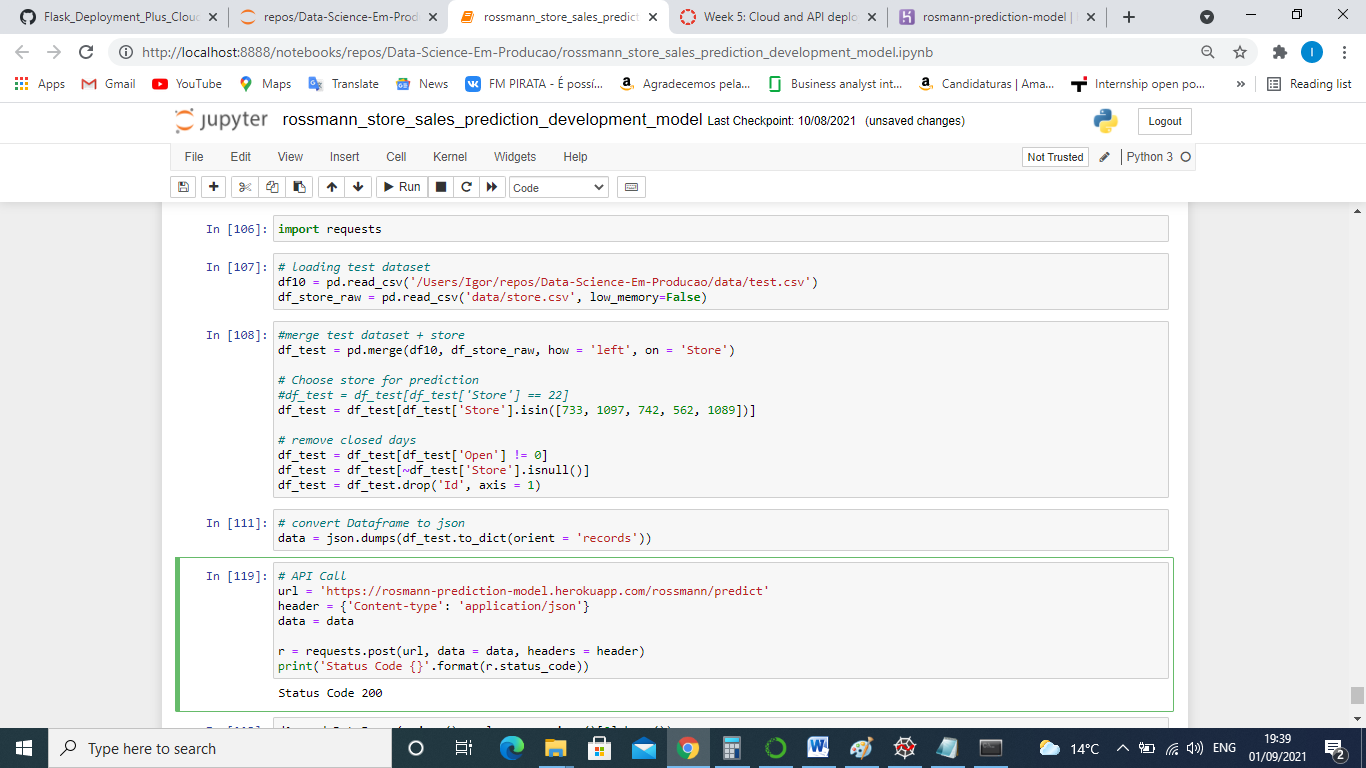


After upload it is possible see the app on heroku webpage.



Testing the Heroku API

In order to test the Heroku API a test is made at the dataset test.csv and store csv. This test show a code which informs if the API is working or not. Case the test shows the code 200, means that the API is working fine.



Making Predictions using Flask API

Now that the API is working well, it is possible to use it to make predictions. In this case I am using it at my private host (my laptop), but it is possible let it available online, for example using heroku.

