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_Plus_Cloud_and_API_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.

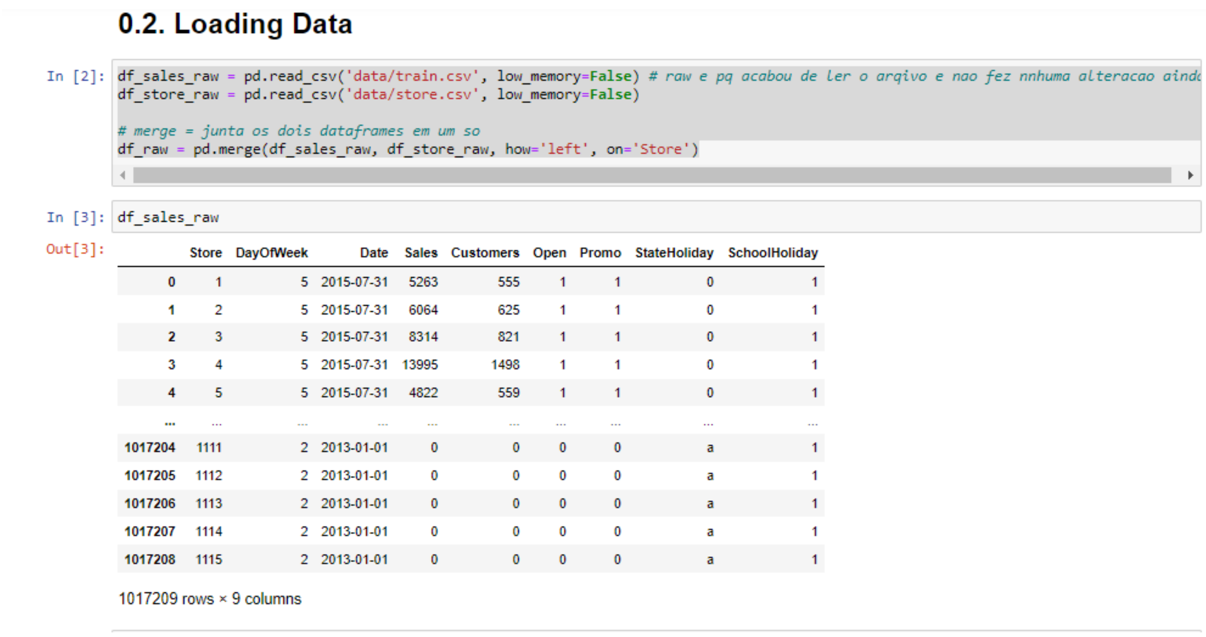
So, the main objective here is making a prediction of sales for 6 weeks in advance.

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.



0.2. Loading Data

In [2]:
```
df_sales_raw = pd.read_csv('data/train.csv', low_memory=False) # raw e pq acabou de ler o arqivo e nao fez nnhuma alteracao aind
df_store_raw = pd.read_csv('data/store.csv', low_memory=False)

# merge = junta os dois dataframes em um so
df_raw = pd.merge(df_sales_raw, df_store_raw, how='left', on='Store')
```

In [3]:
```
df_sales_raw
```

Out[3]:

	Store	DayOfWeek	Date	Sales	Customers	Open	Promo	StateHoliday	SchoolHoliday
0	1	5	2015-07-31	5263	555	1	1	0	1
1	2	5	2015-07-31	6064	625	1	1	0	1
2	3	5	2015-07-31	8314	821	1	1	0	1
3	4	5	2015-07-31	13995	1498	1	1	0	1
4	5	5	2015-07-31	4822	559	1	1	0	1
...	...	...	...	...	...	...	...	...	...
1017204	1111	2	2013-01-01	0	0	0	0	a	1
1017205	1112	2	2013-01-01	0	0	0	0	a	1
1017206	1113	2	2013-01-01	0	0	0	0	a	1
1017207	1114	2	2013-01-01	0	0	0	0	a	1
1017208	1115	2	2013-01-01	0	0	0	0	a	1

1017209 rows × 9 columns

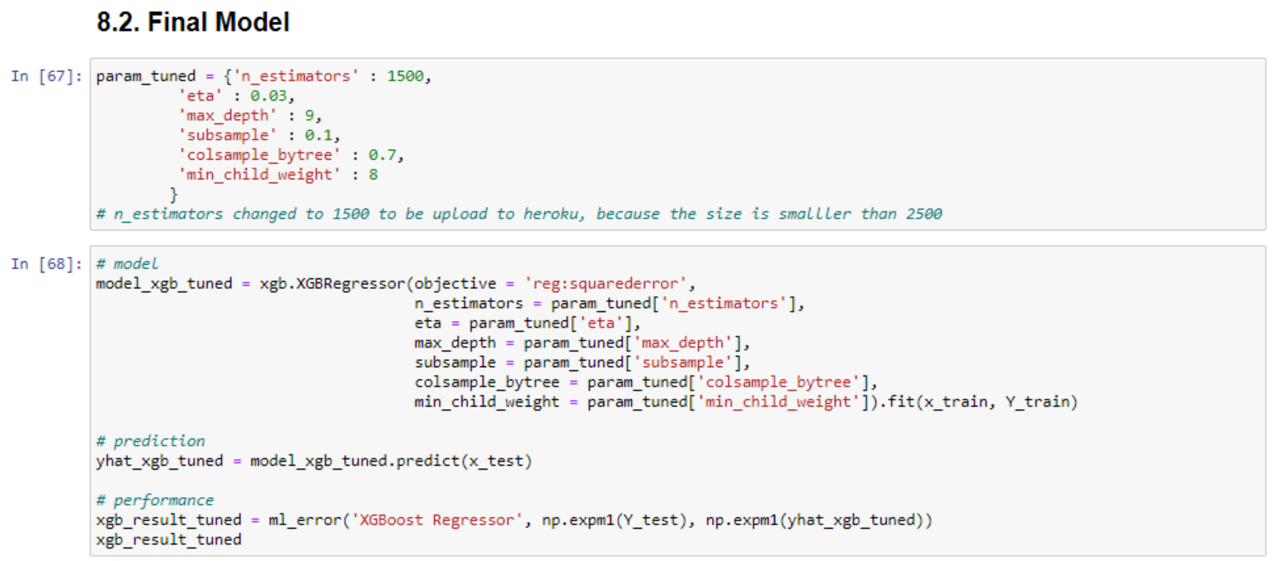
After this step, a sequence of process is made in order to have a property dataset to apply the ML model, such as:

- 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.



8.2. Final Model

```
In [67]: param_tuned = {'n_estimators' : 1500,
                  'eta' : 0.03,
                  'max_depth' : 9,
                  'subsample' : 0.1,
                  'colsample_bytree' : 0.7,
                  'min_child_weight' : 8
                 }
         # n_estimators changed to 1500 to be upload to heroku, because the size is smaller than 2500
```

```
In [68]: # model
         model_xgb_tuned = xgb.XGBRegressor(objective = 'reg:squarederror',
                                            n_estimators = param_tuned['n_estimators'],
                                            eta = param_tuned['eta'],
                                            max_depth = param_tuned['max_depth'],
                                            subsample = param_tuned['subsample'],
                                            colsample_bytree = param_tuned['colsample_bytree'],
                                            min_child_weight = param_tuned['min_child_weight']).fit(x_train, Y_train)

         # prediction
         yhat_xgb_tuned = model_xgb_tuned.predict(x_test)

         # performance
         xgb_result_tuned = ml_error('XGBoost Regressor', np.expm1(Y_test), np.expm1(yhat_xgb_tuned))
         xgb_result_tuned
```

Deploy Model to Production

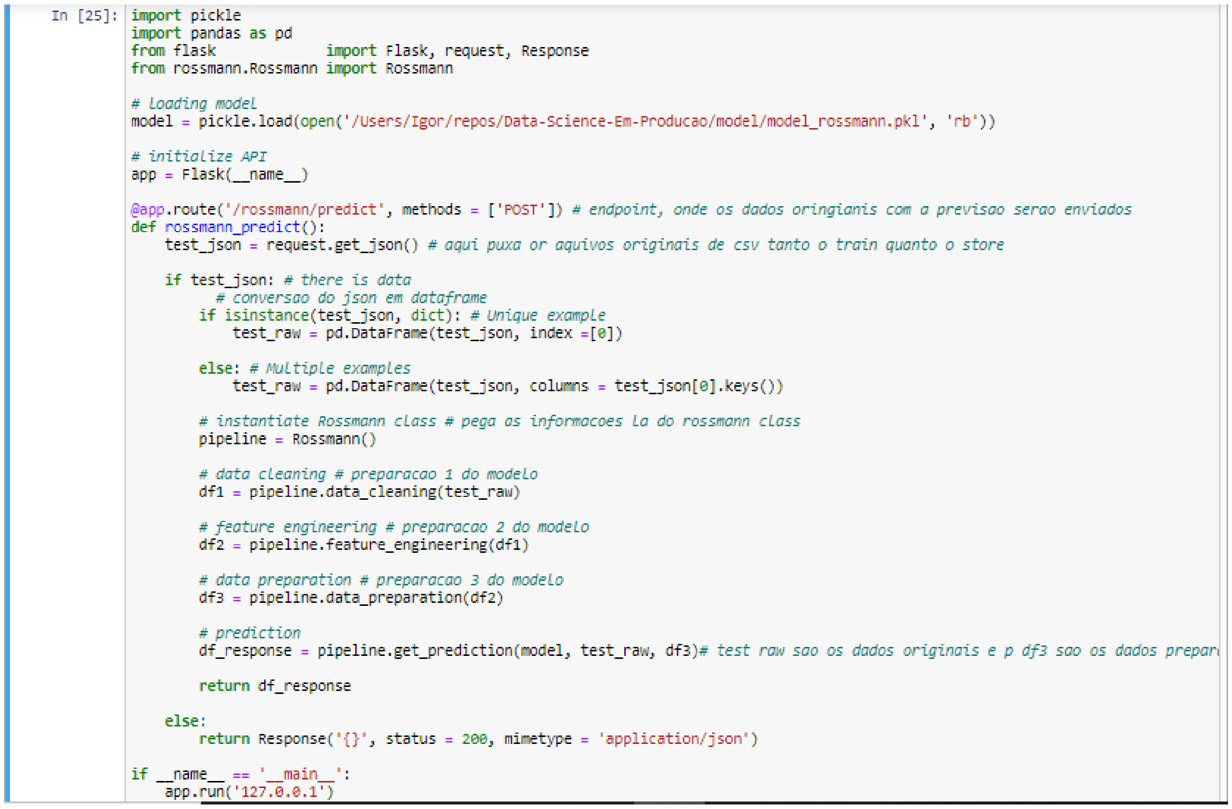
Finally here it is the process of model deployment using flask.

First the model is saved.

```
In [97]: # Save Training Model
         pickle.dump(model_xgb_tuned, open('/Users/Igor/repos/Data-Science-Em-Producao/model/model_rossmann.pkl', 'wb'))
```

Creating Flask API

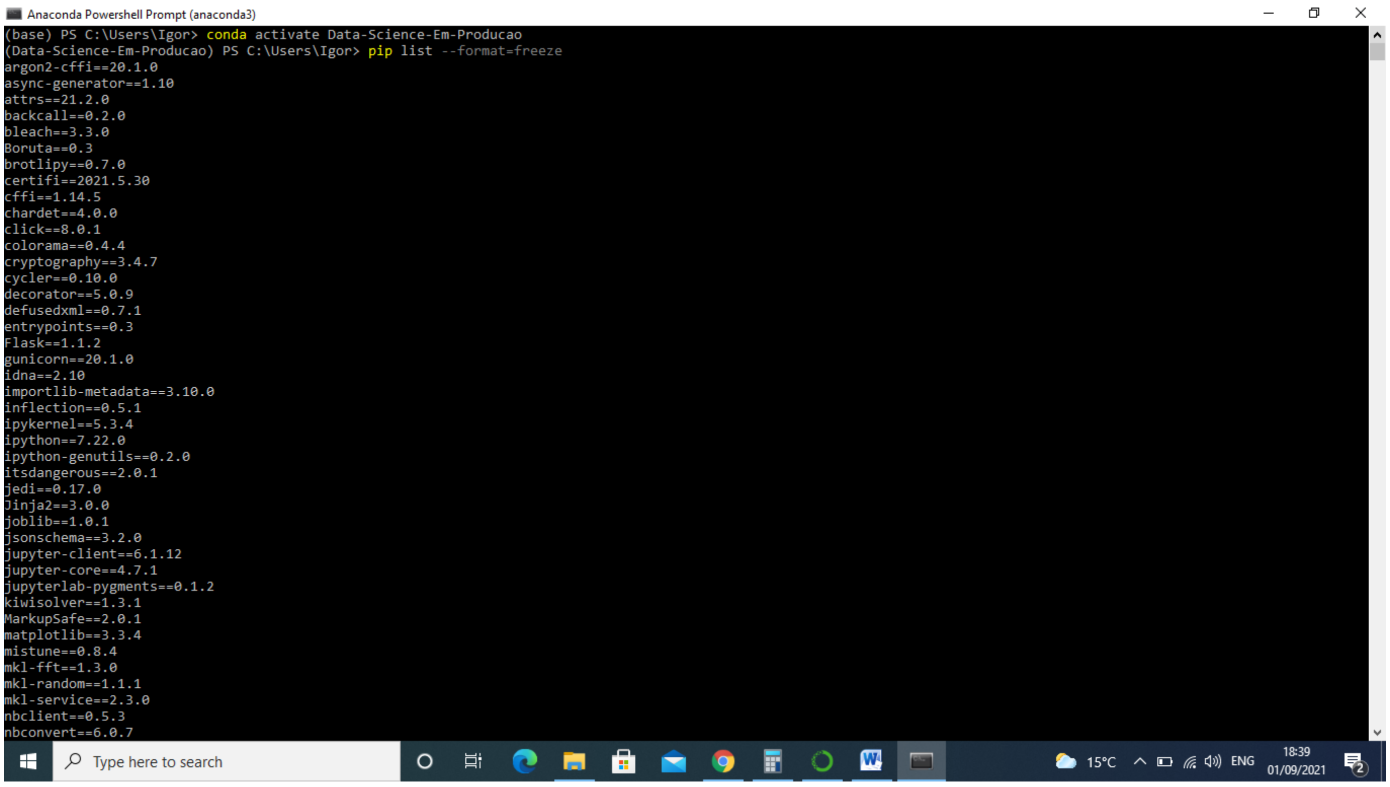
Now the API using flask is created to make the deploy



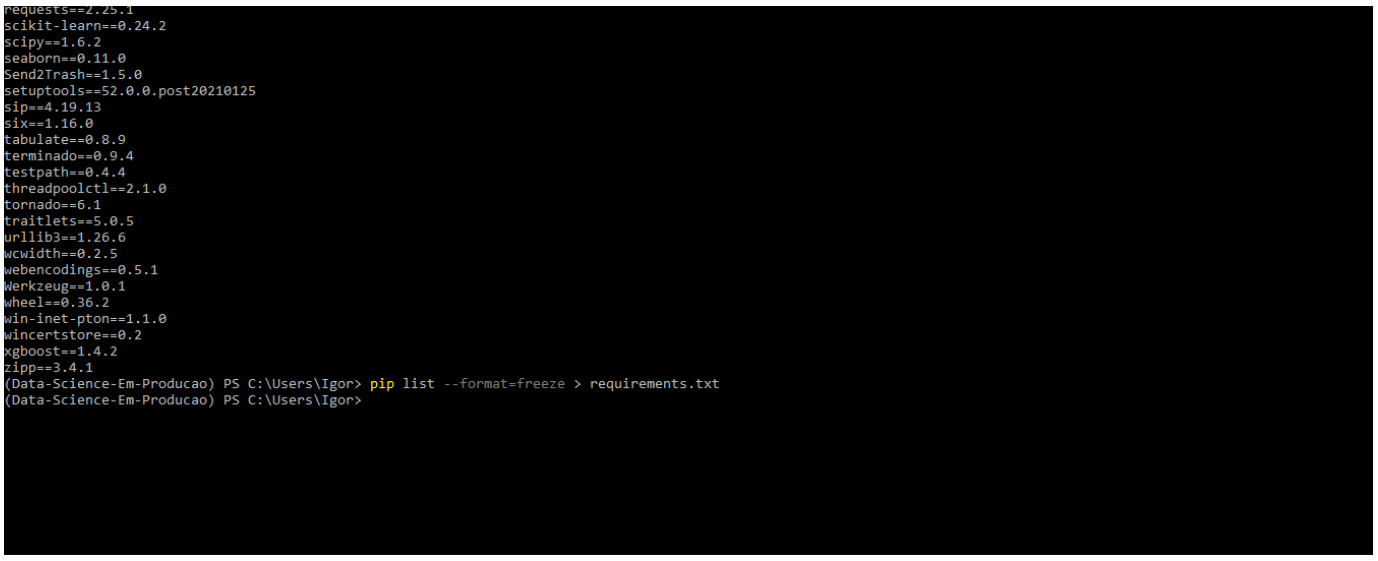
```
In [25]: import pickle
         import pandas as pd
         from flask             import Flask, request, Response
         from rossmann.Rossmann import Rossmann

         # Loading model
         model = pickle.load(open('/Users/Igor/repos/Data-Science-Em-Producao/model/model_rossmann.pkl', 'rb'))

         # initialize API
         app = Flask(__name__)

         @app.route('/rossmann/predict', methods = ['POST']) # endpoint, onde os dados oringianis com a previsao serao enviados
         def rossmann_predict():
             test_json = request.get_json() # aqui puxa or aquivos originais de csv tanto o train quanto o store

             if test_json: # there is data
                   # conversao do json em dataframe
                 if isinstance(test_json, dict): # Unique example
                     test_raw = pd.DataFrame(test_json, index =[0])

                 else: # Multiple examples
                     test_raw = pd.DataFrame(test_json, columns = test_json[0].keys())

                 # instantiate Rossmann class # pega as informacoes la do rossmann class
                 pipeline = Rossmann()

                 # data cleaning # preparacao 1 do modelo
                 df1 = pipeline.data_cleaning(test_raw)

                 # feature engineering # preparacao 2 do modelo
                 df2 = pipeline.feature_engineering(df1)

                 # data preparation # preparacao 3 do modelo
                 df3 = pipeline.data_preparation(df2)

                 # prediction
                 df_response = pipeline.get_prediction(model, test_raw, df3)# test raw sao os dados originais e p df3 sao os dados prepar

                 return df_response

             else:
                 return Response('{}', status = 200, mimetype = 'application/json')

         if __name__ == '__main__':
             app.run('127.0.0.1')
```

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.



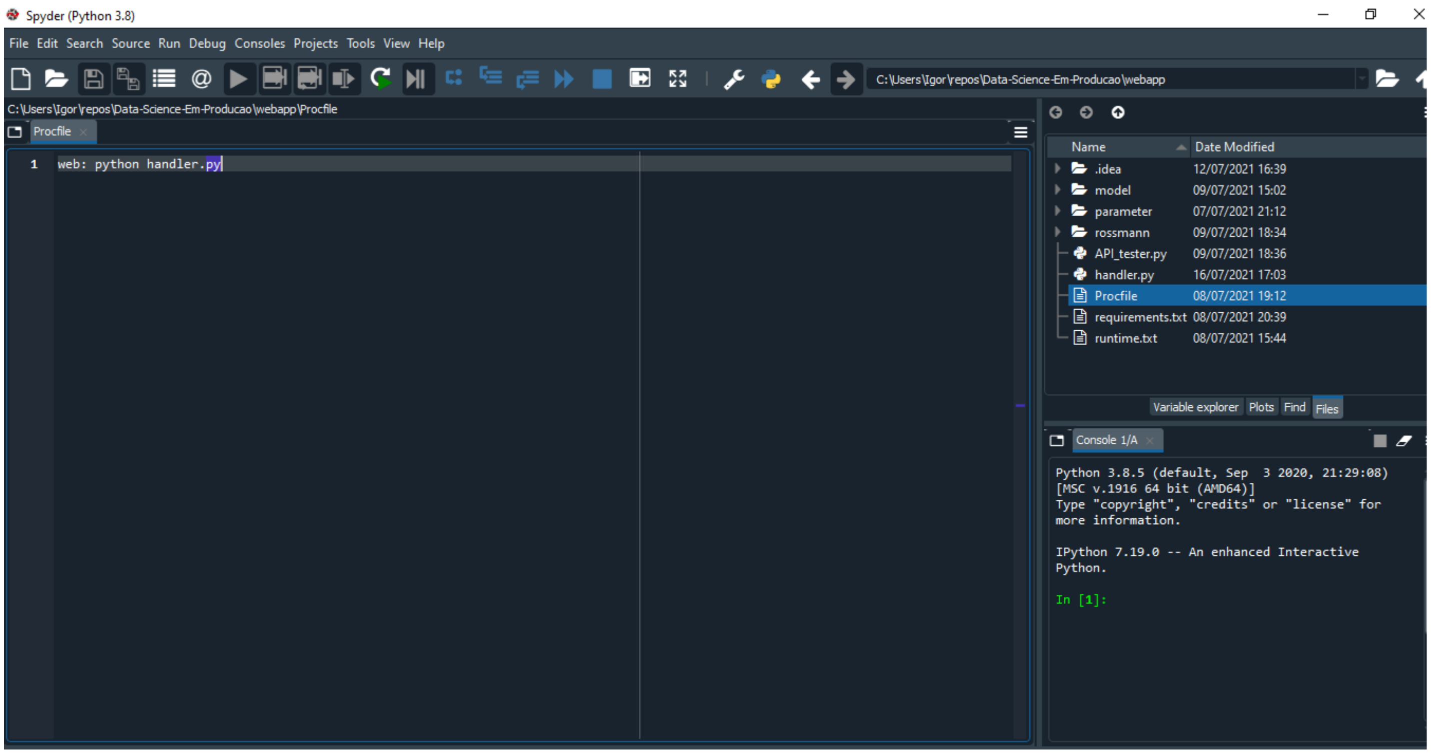
```
(base) PS C:\Users\Igor> conda activate Data-Science-Em-Producao
(Data-Science-Em-Producao) PS C:\Users\Igor> pip list --format=freeze
argon2-cffi==20.1.0
async-generator==1.10
attrs==21.2.0
backcall==0.2.0
bleach==3.3.0
Boruta==0.3
brotlipy==0.7.0
certifi==2021.5.30
cffi==1.14.5
chardet==4.0.0
click==8.0.1
colorama==0.4.4
cryptography==3.4.7
cycler==0.10.0
decorator==5.0.9
defusedxml==0.7.1
entrypoints==0.3
Flask==1.1.2
gunicorn==20.1.0
idna==2.10
importlib-metadata==3.10.0
inflection==0.5.1
ipykernel==5.3.4
ipython==7.22.0
ipython-genutils==0.2.0
itsdangerous==2.0.1
jedi==0.17.0
Jinja2==3.0.0
joblib==1.0.1
jsonschema==3.2.0
jupyter-client==6.1.12
jupyter-core==4.7.1
jupyterlab-pygments==0.1.2
kiwisolver==1.3.1
MarkupSafe==2.0.1
matplotlib==3.3.4
mistune==0.8.4
mkl-fft==1.3.0
mkl-random==1.1.1
mkl-service==2.3.0
nbclient==0.5.3
nbconvert==6.0.7
```



```
requests==2.25.1
scikit-learn==0.24.2
scipy==1.6.2
seaborn==0.11.0
Send2Trash==1.5.0
setuptools==52.0.0.post20210125
sip==4.19.13
six==1.16.0
tabulate==0.8.9
terminado==0.9.4
testpath==0.4.4
threadpoolctl==2.1.0
tornado==6.1
traitlets==5.0.5
urllib3==1.26.6
wcwidth==0.2.5
webencodings==0.5.1
Werkzeug==1.0.1
wheel==0.36.2
win-inet-pton==1.1.0
wincertstore==0.2
xgboost==1.4.2
zipp==3.4.1
(Data-Science-Em-Producao) PS C:\Users\Igor> pip list --format=freeze > requirements.txt
(Data-Science-Em-Producao) PS C:\Users\Igor>
```

Creating Procfile

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


Spyder (Python 3.8)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Igor\repos\Data-Science-Em-Producao\webapp
C:\Users\Igor\repos\Data-Science-Em-Producao\webapp\Procfile
Procfile
Variable explorer Plots Find Files
Console 1/A

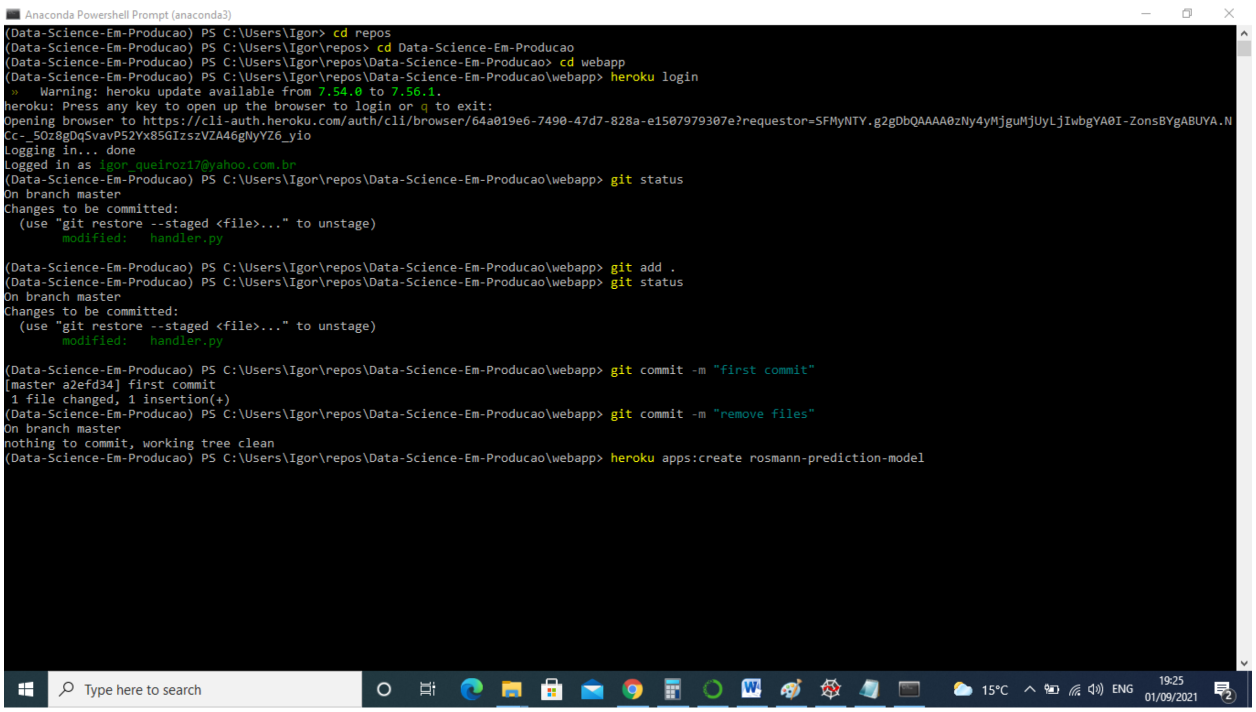
```
web: python handler.py
```

Name	Date Modified
.idea	12/07/2021 16:39
model	09/07/2021 15:02
parameter	07/07/2021 21:12
rossmann	09/07/2021 18:34
API_tester.py	09/07/2021 18:36
handler.py	16/07/2021 17:03
Procfile	08/07/2021 19:12
requirements.txt	08/07/2021 20:39
runtime.txt	08/07/2021 15:44

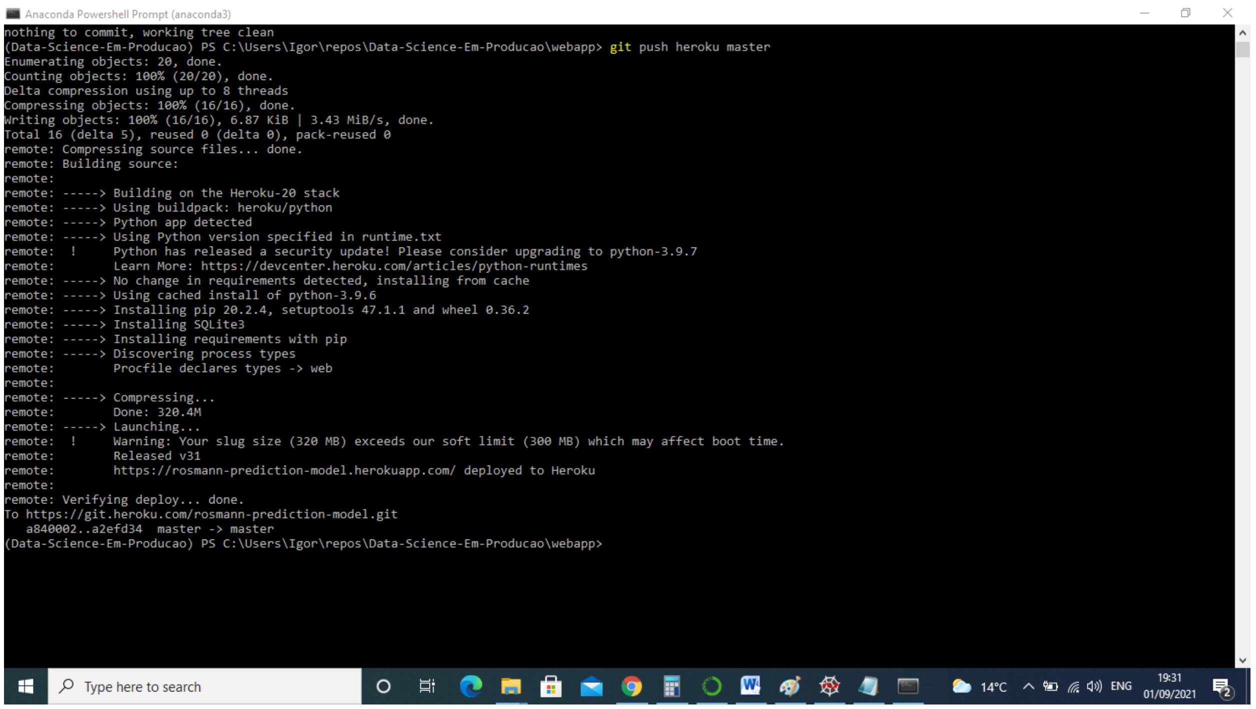
```
Python 3.8.5 (default, Sep  3 2020, 21:29:08)
[MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for
more information.

IPython 7.19.0 -- An enhanced Interactive
Python.

In [1]:
```

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.

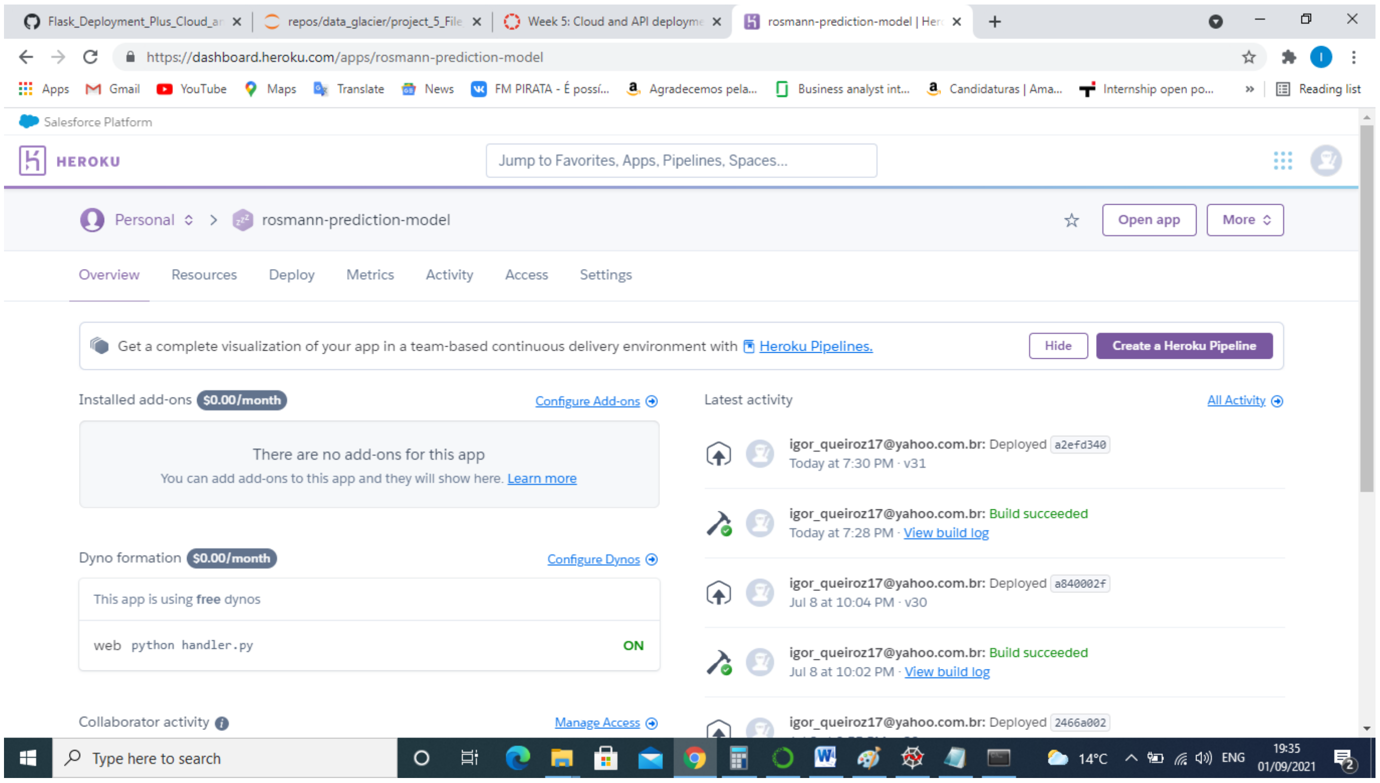

Anaconda Powershell Prompt (anaconda3)

```
(Data-Science-Em-Producao) PS C:\Users\Igor> cd repos
(Data-Science-Em-Producao) PS C:\Users\Igor\repos> cd Data-Science-Em-Producao
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao> cd webapp
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> heroku login
 »   Warning: heroku update available from 7.54.0 to 7.56.1.
heroku: Press any key to open up the browser to login or q to exit:
Opening browser to https://cli-auth.heroku.com/auth/cli/browser/64a019e6-7490-47d7-828a-e1507979307e?requestor=SFMyNTY.g2gDbQAAAA0zNy4yMjguMjUyLjIwbgYA0I-ZonsBYgABUYA.N
Cc-_5Oz8gDqSvavP52Yx85GIzszVZA46gNyYZ6_yio
Logging in... done
Logged in as igor_queiroz17@yahoo.com.br
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   handler.py

(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> git add .
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   handler.py

(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> git commit -m "first commit"
[master a2efd34] first commit
 1 file changed, 1 insertion(+)
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> git commit -m "remove files"
On branch master
nothing to commit, working tree clean
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> heroku apps:create rosmann-prediction-model
```


Anaconda Powershell Prompt (anaconda3)

```
nothing to commit, working tree clean
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp> git push heroku master
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 8 threads
Compressing objects: 100% (16/16), done.
Writing objects: 100% (16/16), 6.87 KiB | 3.43 MiB/s, done.
Total 16 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Compressing source files... done.
remote: Building source:
remote:
remote: -----> Building on the Heroku-20 stack
remote: -----> Using buildpack: heroku/python
remote: -----> Python app detected
remote: -----> Using Python version specified in runtime.txt
remote:  !     Python has released a security update! Please consider upgrading to python-3.9.7
remote:        Learn More: https://devcenter.heroku.com/articles/python-runtimes
remote: -----> No change in requirements detected, installing from cache
remote: -----> Using cached install of python-3.9.6
remote: -----> Installing pip 20.2.4, setuptools 47.1.1 and wheel 0.36.2
remote: -----> Installing SQLite3
remote: -----> Installing requirements with pip
remote: -----> Discovering process types
remote:        Procfile declares types -> web
remote:
remote: -----> Compressing...
remote:        Done: 320.4M
remote: -----> Launching...
remote:  !     Warning: Your slug size (320 MB) exceeds our soft limit (300 MB) which may affect boot time.
remote:        Released v31
remote:        https://rosmann-prediction-model.herokuapp.com/ deployed to Heroku
remote:
remote: Verifying deploy... done.
To https://git.heroku.com/rosmann-prediction-model.git
   a840002..a2efd34  master -> master
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\webapp>
```

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


https://dashboard.heroku.com/apps/rosmann-prediction-model

Personal > rosmann-prediction-model

Installed add-ons $0.00/month

There are no add-ons for this app
You can add add-ons to this app and they will show here. Learn more

Dyno formation $0.00/month

This app is using free dynos

web python handler.py ON

Latest activity

igor_queiroz17@yahoo.com.br: Deployed a2efd340
Today at 7:30 PM · v31

igor_queiroz17@yahoo.com.br: Build succeeded
Today at 7:28 PM · View build log

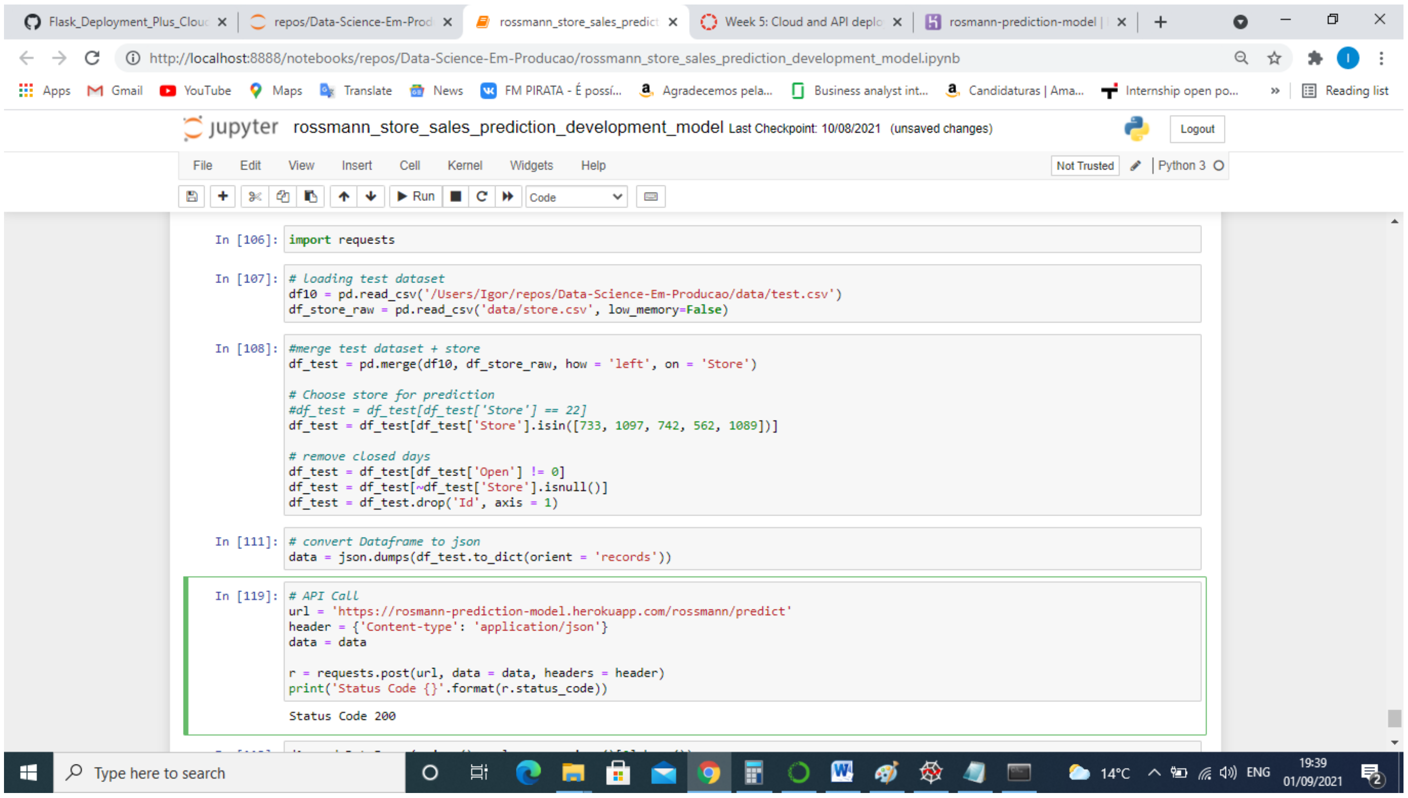
igor_queiroz17@yahoo.com.br: Deployed a840002f
Jul 8 at 10:04 PM · v30

igor_queiroz17@yahoo.com.br: Build succeeded
Jul 8 at 10:02 PM · View build log

igor_queiroz17@yahoo.com.br: Deployed 2466a002

Testing the Heroku API

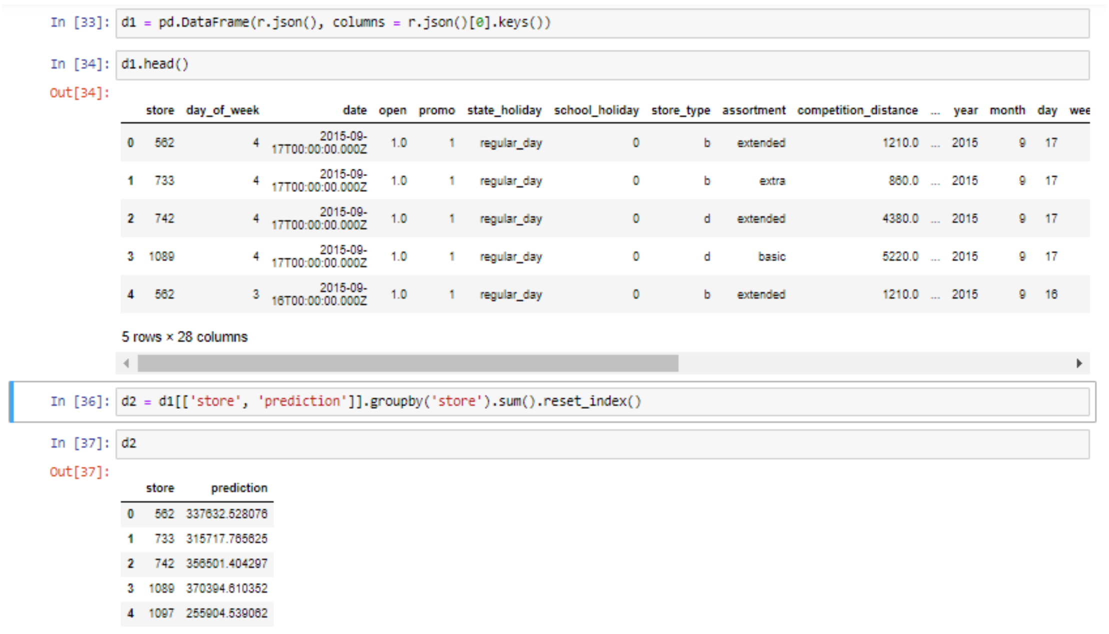
In order to test the Heroku API a test is made at the dataset test.csv and store csv. This test show a code which informs if the API is working or not. Case the test shows the code 200, means that the API is working fine.


jupyter rossmann_store_sales_prediction_development_model Last Checkpoint: 10/08/2021 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3

```
In [106]: import requests

In [107]: # loading test dataset
          df10 = pd.read_csv('/Users/Igor/repos/Data-Science-Em-Producao/data/test.csv')
          df_store_raw = pd.read_csv('data/store.csv', low_memory=False)

In [108]: #merge test dataset + store
          df_test = pd.merge(df10, df_store_raw, how = 'left', on = 'Store')

          # Choose store for prediction
          #df_test = df_test[df_test['Store'] == 22]
          df_test = df_test[df_test['Store'].isin([733, 1097, 742, 562, 1089])]

          # remove closed days
          df_test = df_test[df_test['Open'] != 0]
          df_test = df_test[~df_test['Store'].isnull()]
          df_test = df_test.drop('Id', axis = 1)

In [111]: # convert Dataframe to json
          data = json.dumps(df_test.to_dict(orient = 'records'))

In [119]: # API Call
          url = 'https://rosmann-prediction-model.herokuapp.com/rossmann/predict'
          header = {'Content-type': 'application/json'}
          data = data

          r = requests.post(url, data = data, headers = header)
          print('Status Code {}'.format(r.status_code))

          Status Code 200
```

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.



```
In [33]: d1 = pd.DataFrame(r.json(), columns = r.json()[0].keys())
```

```
In [34]: d1.head()
```

Out[34]:

	store	day_of_week	date	open	promo	state_holiday	school_holiday	store_type	assortment	competition_distance	...	year	month	day	wee
0	562	4	2015-09-17T00:00:00.000Z	1.0	1	regular_day	0	b	extended	1210.0	...	2015	9	17	
1	733	4	2015-09-17T00:00:00.000Z	1.0	1	regular_day	0	b	extra	860.0	...	2015	9	17	
2	742	4	2015-09-17T00:00:00.000Z	1.0	1	regular_day	0	d	extended	4380.0	...	2015	9	17	
3	1089	4	2015-09-17T00:00:00.000Z	1.0	1	regular_day	0	d	basic	5220.0	...	2015	9	17	
4	562	3	2015-09-16T00:00:00.000Z	1.0	1	regular_day	0	b	extended	1210.0	...	2015	9	16	

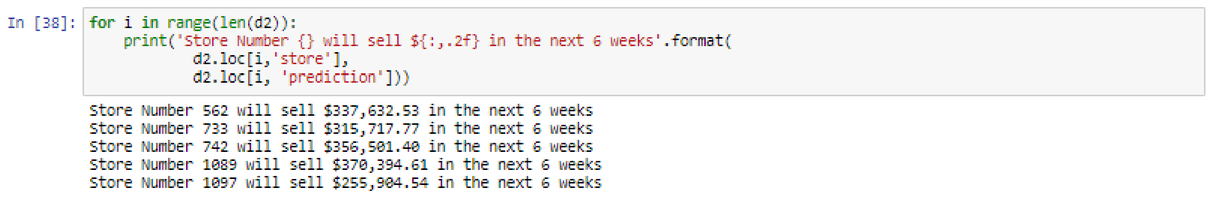
5 rows × 28 columns

```
In [36]: d2 = d1[['store', 'prediction']].groupby('store').sum().reset_index()
```

```
In [37]: d2
```

Out[37]:

	store	prediction
0	562	337632.528076
1	733	315717.765625
2	742	356501.404297
3	1089	370394.610352
4	1097	255904.539062



```
In [38]: for i in range(len(d2)):
             print('Store Number {} will sell ${:,.2f} in the next 6 weeks'.format(
                     d2.loc[i,'store'],
                     d2.loc[i, 'prediction']))
```

```
Store Number 562 will sell $337,632.53 in the next 6 weeks
Store Number 733 will sell $315,717.77 in the next 6 weeks
Store Number 742 will sell $356,501.40 in the next 6 weeks
Store Number 1089 will sell $370,394.61 in the next 6 weeks
Store Number 1097 will sell $255,904.54 in the next 6 weeks
```