Name: <Process to deploy a prediction using flask>

Report date: <27/08/2021>

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 Flask

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 smallller 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'))
```

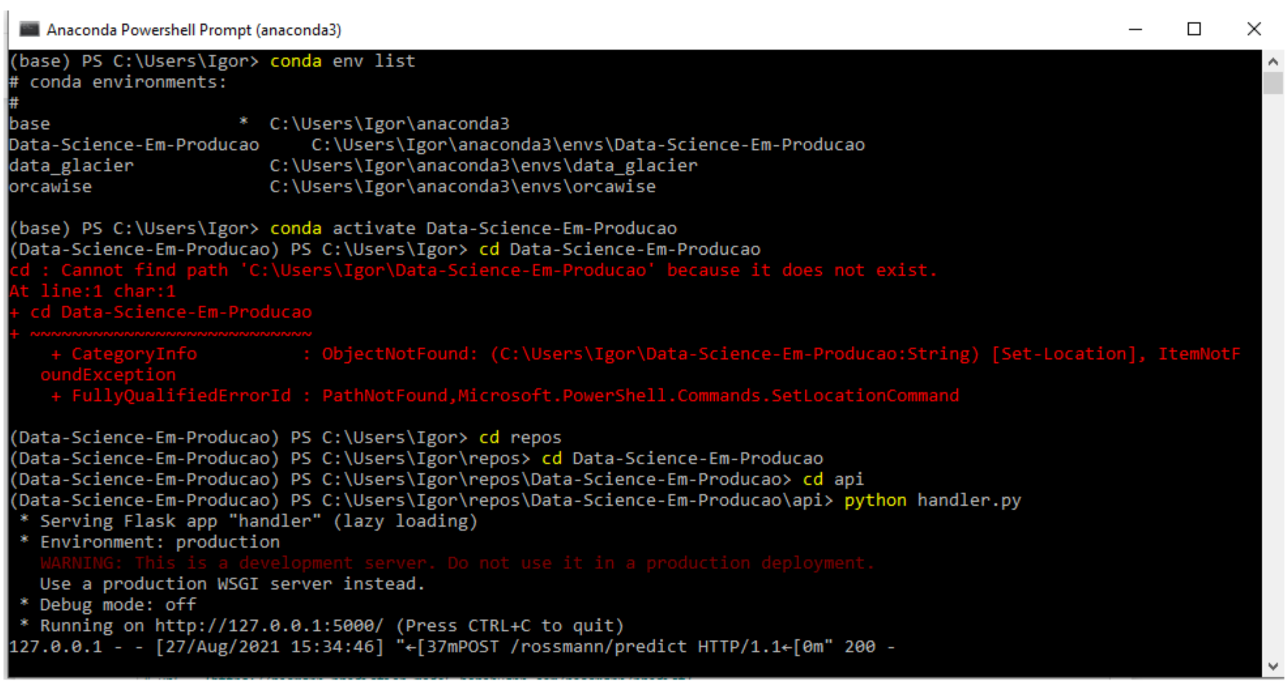
Creating Flask API

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')
```

Activating API Flask

To activate it is necessary access the CMD and activate the environment responsible for all libraries used, in this case a library created by me called data_glacier with all necessary functions. After that just active the API flask named handler using python.

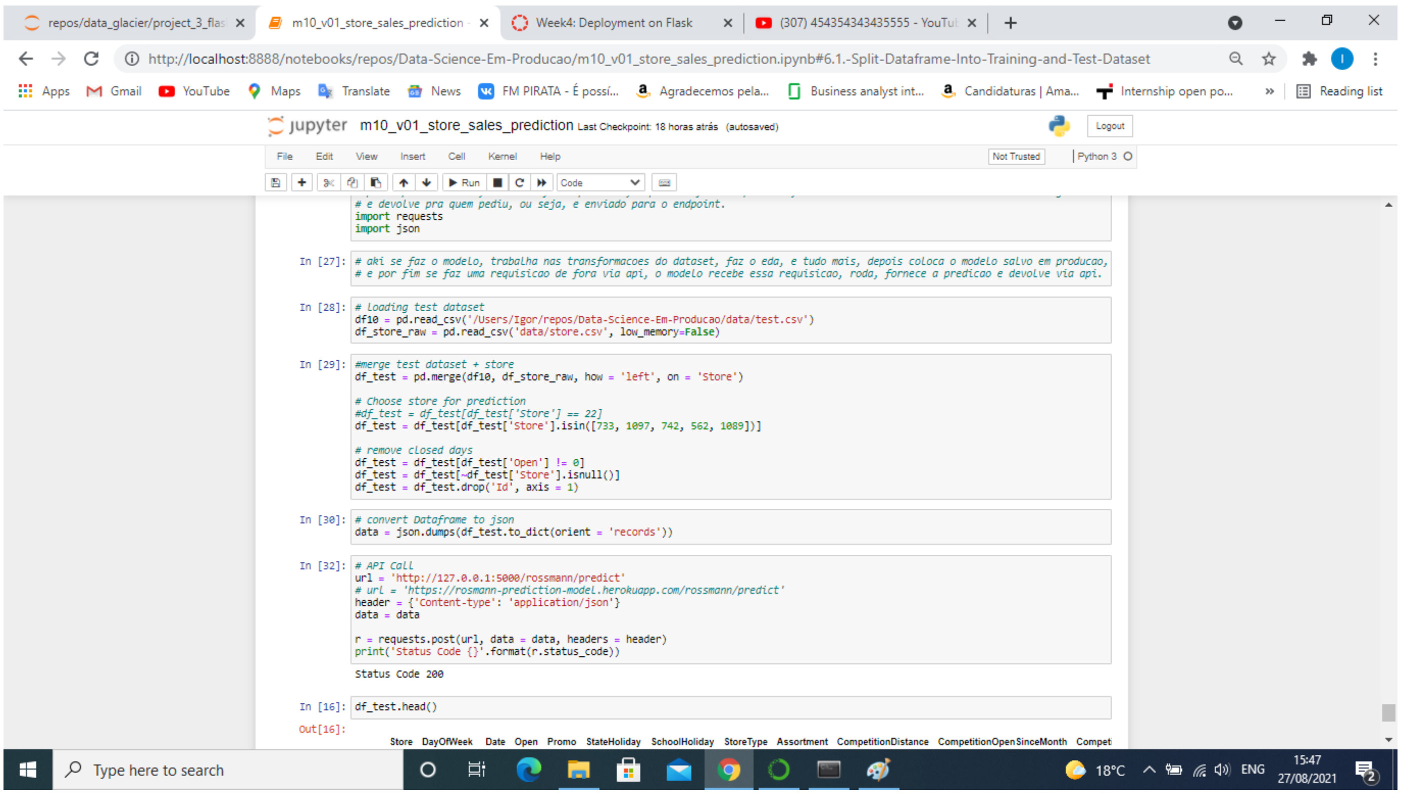


Anaconda Powershell Prompt (anaconda3)

```
(base) PS C:\Users\Igor> conda env list
# conda environments:
#
base                  *  C:\Users\Igor\anaconda3
Data-Science-Em-Producao     C:\Users\Igor\anaconda3\envs\Data-Science-Em-Producao
data_glacier             C:\Users\Igor\anaconda3\envs\data_glacier
orcawise                 C:\Users\Igor\anaconda3\envs\orcawise

(base) PS C:\Users\Igor> conda activate Data-Science-Em-Producao
(Data-Science-Em-Producao) PS C:\Users\Igor> cd Data-Science-Em-Producao
cd : Cannot find path 'C:\Users\Igor\Data-Science-Em-Producao' because it does not exist.
At line:1 char:1
+ cd Data-Science-Em-Producao
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\Users\Igor\Data-Science-Em-Producao:String) [Set-Location], ItemNotF
   oundException
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

(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 api
(Data-Science-Em-Producao) PS C:\Users\Igor\repos\Data-Science-Em-Producao\api> python handler.py
 * Serving Flask app "handler" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [27/Aug/2021 15:34:46] "←[37mPOST /rossmann/predict HTTP/1.1←[0m" 200 -
```

Testing the Flask API

In order to test th Flask API it is tested 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.

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
```