Soutenance

Recruit Restaurant Visitor Forecasting

Predict how many future visitors a restaurant will receive

Data Science Starter Program

DSSP 7 – 2017

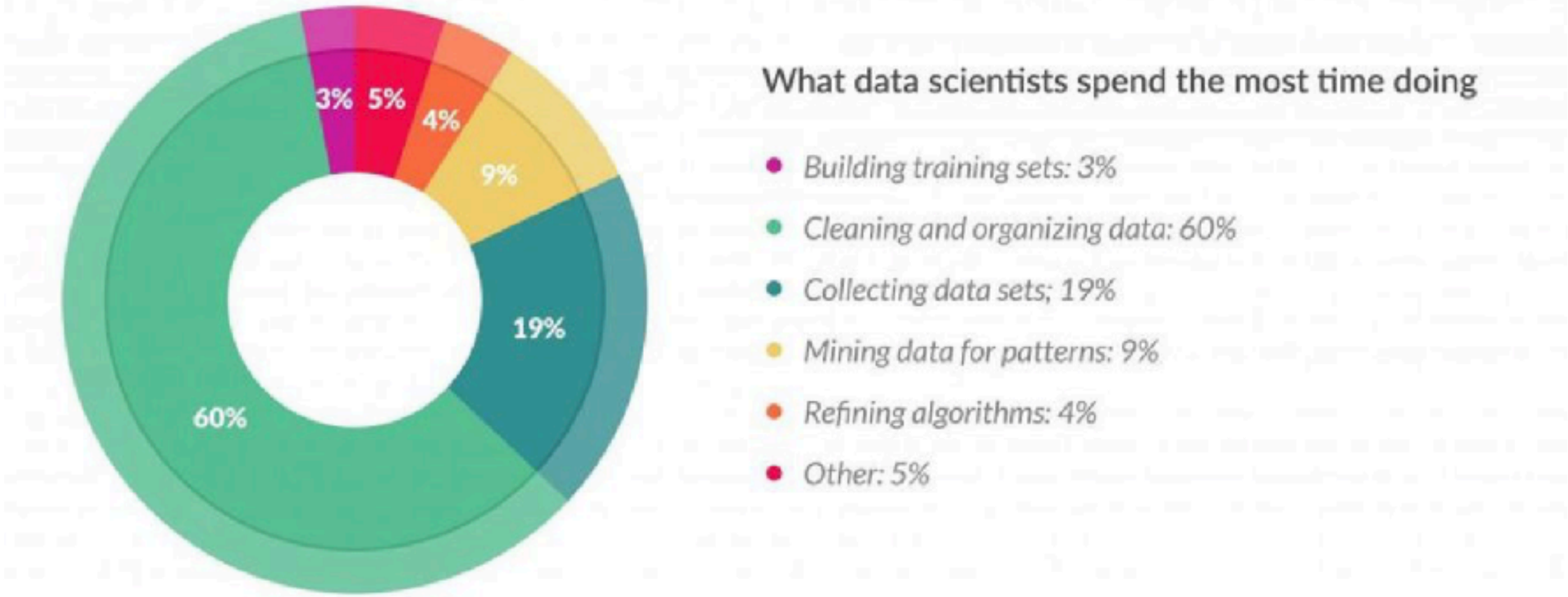
Jordan VIDAL

Introduction

- Objectif : prédire le nombre de visiteurs dans certains restaurants japonais à certaines dates
- Apprentissage supervisé & régression
- Compétition Kaggle
- Projet “data scientist”
- Projet codé en python3
- 7 packages utilisés

Sommaire

Data preparation *accounts for about 80% of the work of data scientists*



Source: www.forbes.com

Présentation des données

- 7 dataset fournis dont 2 sources de données principales (deux systèmes de réservation)

	Hot Pepper Gourmet (HPG)	**AirREGI / Restaurant Board** (AIR)
Réservations	hpg_reserve.csv *(2 000 320, 4)*	air_reserve.csv *(92 378, 4)*
Restaurants	hpg_store_info.csv *(4690, 5)*	air_store_info.csv *(829, 5)*

- store_id_relation.csv (150, 2)
- sample_submission.csv (32 019, 2)
- date_info.csv (517, 3)

Data Exploration

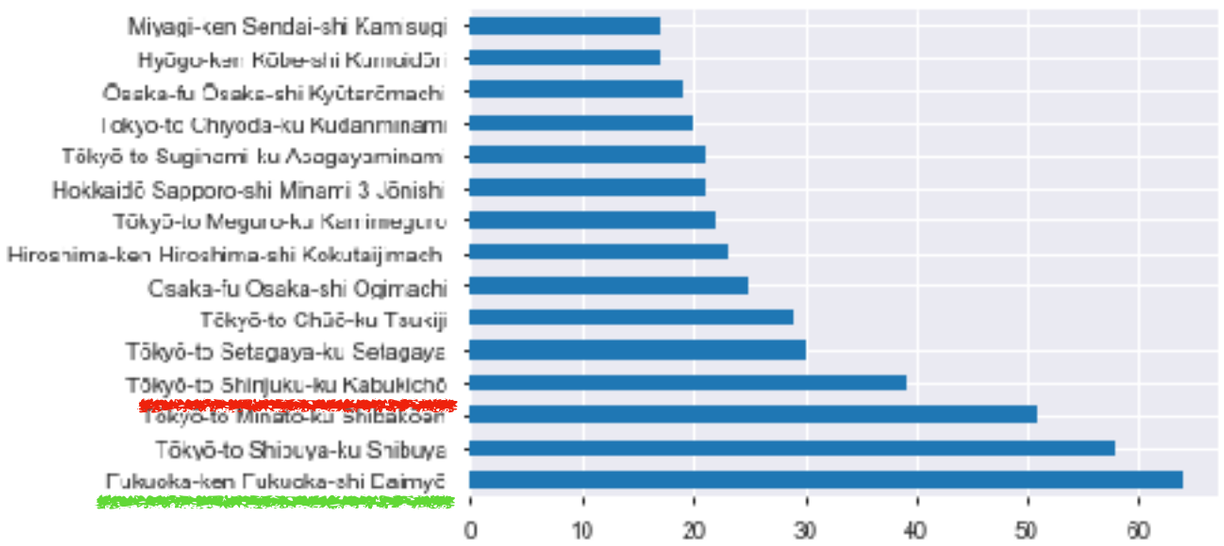
Réservations	Restaurants	Autre
Il y a 20x plus de réservations effectuées via le système HPG que via le système AIR	Le système AIR comporte 829 restaurants de 14 types de cuisines différents et repartis dans 103 “area” du pays	Il n’y a que 150 restaurants présents dans les deux systèmes de réservations (store_id_relation.csv)
Les deux dataset ont enregistré les réservations effectuées sur la même période : du 1er janvier 2016 au 22 avril 2017	Le système HPG comporte 4690 restaurants dans 34 genre de cuisines différentes et répartis dans 119 “area” du pays	Le dataset date_info.csv (“hol”) nous donne des informations sur le jour de la semaine et sur les jours de vacances (“holiday_flg”) entre le 01-01-2016 et le 31-05-2017
La distribution du nombre de visiteurs par réservations sur les deux systèmes AIR et HPG sont similaires : la moyenne du nombre de visiteurs par réservation est de ~5	Les “area_name” sont parfois très similaires (ex: “Tokyo-to Shinjuku-ku None” vs “Tokyo-to Shinjuku-ku Kabukicho”)	Le dataset “sample_submission” va nous servir de test. Il contient 32019 lignes et 2 colonnes : - une colonne “id” qui est une concaténation de “*air_store_id*” et une date au format *YYYY-MM-DD (comprises entre le 23 avril 2017 et le 31 mai 2017)* - colonne “*visitors*” qui est le nombre de visiteurs a prédire

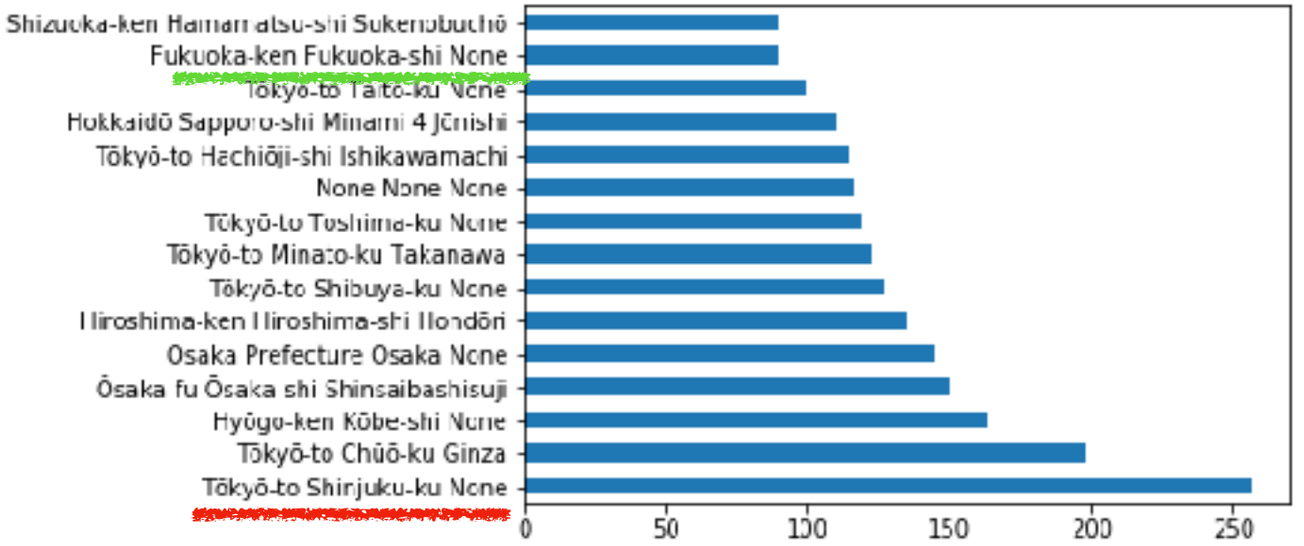
Data Exploration

Réservations	Restaurants	Autre
Il y a **20x plus de réservations** effectuées via le système HPG que via le système AIR	Le système AIR comporte **829 restaurants** de **14 types de cuisines différents** et repartis dans **103 “area”** du pays	Il n’y a que **150 restaurants** présents dans les deux systèmes de réservations (store_id_relation.csv)
Les deux dataset ont enregistré les réservations effectuées sur **la même période** : du 1er janvier 2016 au 22 avril 2017	Le système HPG comporte **4690 restaurants** dans **34 genre de cuisines** différentes et répartis dans **119 “area”** du pays	Le dataset date_info.csv renseigne le jour de la semaine et les jours de vacances entre le 1er Janvier 2016 et le 31 Mai 2017
La distribution du nombre de visiteurs par réservations sur les deux systèmes AIR et HPG sont similaires : **la moyenne du nombre de visiteurs par réservation est de ~5**	Les “area_name” sont parfois très similaires (ex: “Tokyo-to Shinjuku-ku None” vs “Tokyo-to Shinjuku-ku Kabukicho”)	Le dataset “sample_submission” va nous servir de test. Il contient 32019 lignes et 2 colonnes : - une colonne “id” qui est une concaténation de “*air_store_id*” et une date au format *YYYY-MM-DD (comprises entre le 23 avril 2017 et le 31 mai 2017)* - colonne “*visitors*” qui est le nombre de visiteurs a prédire


Distribution du nombre de visiteurs par réservations via le systeme AIR
reservations
nombre de visiteurs
0
5000
10000
15000
20000
25000
30000
35000
40000
0
5
10
15
20
25
30


Distribution du nombre de visiteurs par réservations via le systeme HPG
reservations
nombre de visiteurs
0
100000
200000
300000
400000
500000
600000
700000
0
5
10
15
20
25
30


Miyagi-ken Sendai-shi Kamsugi
Hyōgo-ken Kōbe-shi Kumoidōri
Ōsaka-fu Ōsaka-shi Kyūtarōmachi
Tokyo-to Chiyoda-ku Kudanminami
Tōkyō-to Suginami-ku Asagayaminami
Hokkaidō Sapporo-shi Minami 3 Jōnishi
Tōkyō-to Meguro-ku Kamimeguro
Hiroshima-ken Hiroshima-shi Kokutaijimach
Osaka-fu Osaka-shi Ogimachi
Tōkyō-to Chūō-ku Tsukiji
Tōkyō-to Setagaya-ku Setagaya
Tōkyō-to Shinjuku-ku Kabukichō
Tōkyō-to Minato-ku Shibakōen
Tōkyō-to Shibuya-ku Shibuya
Fukuoka-ken Fukuoka-shi Daimyō
0
10
20
30
40
50
60


Shizuoka-ken Hamamatsu-shi Sukenobuchō
Fukuoka-ken Fukuoka-shi None
Tōkyō-to Taito-ku None
Hokkaidō Sapporo-shi Minami 4 Jōnishi
Tōkyō-to Hachiōji-shi Ishikawamachi
None None None
Tōkyō-to Toshima-ku None
Tōkyō-to Minato-ku Takanawa
Tōkyō-to Shibuya-ku None
Hiroshima-ken Hiroshima-shi Hondōri
Osaka Prefecture Osaka None
Ōsaka fu Ōsaka shi Shinsaibashisuji
Hyōgo-ken Kōbe-shi None
Tōkyō-to Chūō-ku Ginza
Tōkyō-to Shinjuku-ku None
0
50
100
150
200
250

“You can have data without information, but you cannot have information without data"

~ Daniel Keys Moran, *science fiction writer*

“Coming up with features is difficult, time-consuming, requires expert knowledge”

~ Prof. Andrew Ng. *VP & Chief Scientist of Baidu, Adjunct Professor at Stanford University*

Data preprocessing

1. Cleaning

- Pas de valeurs manquantes
- Pas de valeurs dupliquées
- Pas de valeurs aberrantes

Merci Kaggle

Et Pandas-Profiling package

Data preprocessing

2. Feature Extraction + Engineering

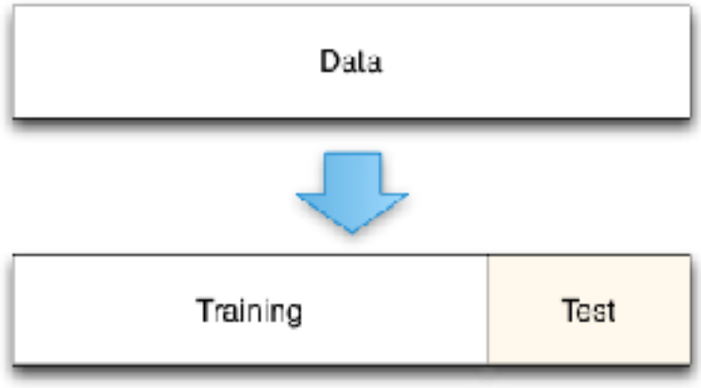
Variables temporelles *Les dates (4)*	dayofweek year month visit_date
Variables catégorielles *(20)*	LabelEncoder sur les 10 plus récurrents “area_name” et “genre_name”
Les nouvelles variables créées **(13)**	min_visitors max_visitors mean_visitors median_visitors count_observations rs1 rs2 rv1 rv2 total_reserv_sum total_reserv_mean total_reserv_dt_diff_mean date_int

Modélisation

RMSE

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}$$

Train/Test split




```
X_train :  (201686, 50)
X_test : (50422, 50)
y_train : (201686,)
y_test : (50422,)
```

Modèles linéaires
vs
modèles non linéaires

Avec et sans
recherche d'hyperparamètres

Modélisation

1. Algorithmes Linéaires

Régression Linéaire, Régression Ridge, Régression Lasso

- R^2 assez faible (<0.57)
- Modèle *sous-apprend* : ne se généralise pas à de nouvelles données
    - Train score = 0.57
    - Test score = 0.56
- La cross validation améliore légèrement R^2 (0.6)
- RMSE = 11.2271

```
First 10 Predictions [ 17.65879935  11.49043542  22.94236448  11.22690248  14.88090359
  16.98326204  24.75024193  10.08335482  10.86074349  44.18380776]
First 10 Real Values 160503    33
74437     20
93611     24
36756     16
61315     10
99936     42
189042    48
78456      9
180807    11
47721     12
```

Modélisation

2. Algorithmes non linéaires

Random Forest
(RandomForestRegressor)

Le modèle sur-apprend (“overfit”)
Train score = 0.92
Test score = 0.53

RMSE = 11.7

	values
mean_visitors	0.574544
date_int	0.167078
month	0.026307
rv1_x	0.023451
max_visitors	0.022363
median_visitors	0.018189
air_store_id2	0.017369
count_observations	0.012910
min_visitors	0.011730
air_area_name4	0.009378
air_area_name2	0.008394
dow	0.007861
air_area_name	0.007241
day_of_week	0.006997
holiday_flg	0.006599
air_genre_name1	0.006101
var_max_lat	0.005346
latitude	0.005304
var_max_long	0.005097
lon_plus_lat	0.004921
air_genre_name	0.004782
longitude	0.004709

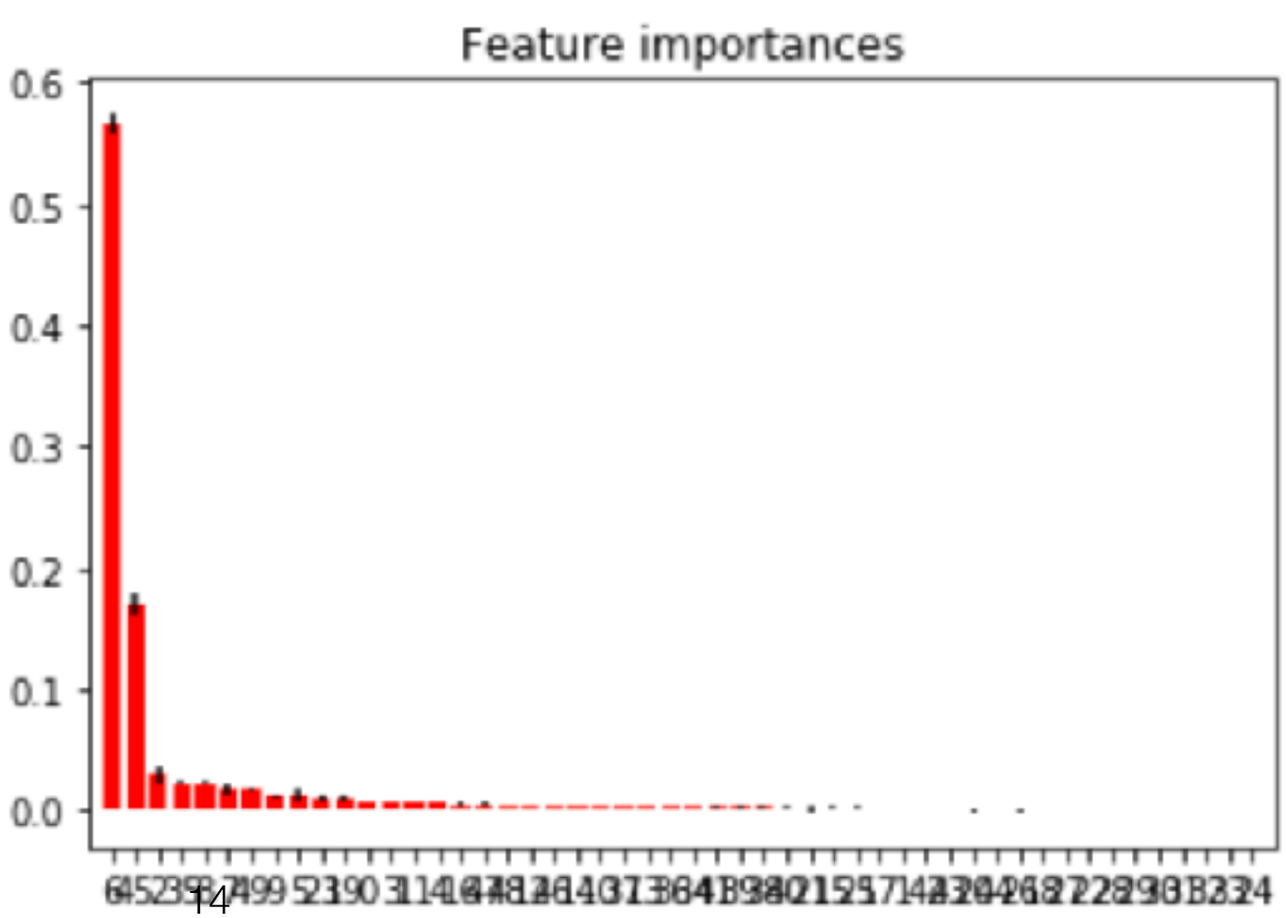
air_genre_name0	0.004598
rs2_x	0.004480
rv2_x	0.004263
rs1_x	0.003644
rv1_y	0.003307
rv2_y	0.002966
rs1_y	0.002929
rs2_y	0.002860
air_area_name0	0.002111
air_area_name1	0.001613
air_area_name5	0.001416
year	0.001362
air_area_name3	0.001279
total_reserv_mean	0.000877
total_reserv_sum	0.000878
total_reserv_dt_diff_mean	0.000594
air_genre_name9	0.000592
air_genre_name6	0.000592
air_area_name7	0.000566
air_genre_name2	0.000375
air_area_name6	0.000246
air_area_name9	0.000000

Random Forest - with GridSearch
(RandomForestRegressor)

Paramètres : n_estimators (100) et max_depth (11)

Modèle ne sur-apprend plus!
Train score = ~~0.92~~ 0.66
Test score = ~~0.53~~ 0.58

RMSE = 11.02



Modélisation

2. Algorithmes non linéaires

XGBoost
(XGBRegressor)

Train score = 0.6
Test score = 0.57

RMSE = 11.05

XGBoost - with GridSearch
(XGBRegressor)

Train score = ~~0.6~~ 0.68
Test score = ~~0.57~~ 0.59

RMSE = 11.02

Paramètres :
n_estimators: 100,
max_depth: 11,
learning_rate: 0.1

XGBoost - with Hyperopt
(XGBRegressor)

Train score = 0.68
Test score = ~~0.59~~ 0.60

RMSE = 10.754

Paramètres :
‘colsample_bytree': 0.8,
'gamma': 2.400000000000004,
'learning_rate': 0,
'max_depth': 4,
'min_child_weight': 4.0,
'n_estimators': 0,
'nthread': 0,
'objective': 0,
'reg_alpha': 1.1,
'reg_lambda': 0.7000000000000001,
'subsample': 1.0

Résultats

	Train/Test	RMSE
Linear Regression	0.5793 / 0.5651	11.2271
Ridge Regression	0.5788 / 0.5648	11.2308
Lasso	0.5749 / 0.5612	11.2767
RandomForest	0.6613 / 0.5807	11.0236
XGBoost	**0.686 / 0.6**	**10.754**

Axes d'améliorations

Feature Engineering:

1. Label encoder sur seulement les 10 premières variables
2. Prise en compte des horaires de réservations, de la géographique (latitude, longitude), de la météo…

Modélisation:

1. Pas de quick&dirty pour démarrer d'une base
2. Pas nécessairement utile de tester plusieurs modèles, l'idée est d'améliorer un ou deux modèles

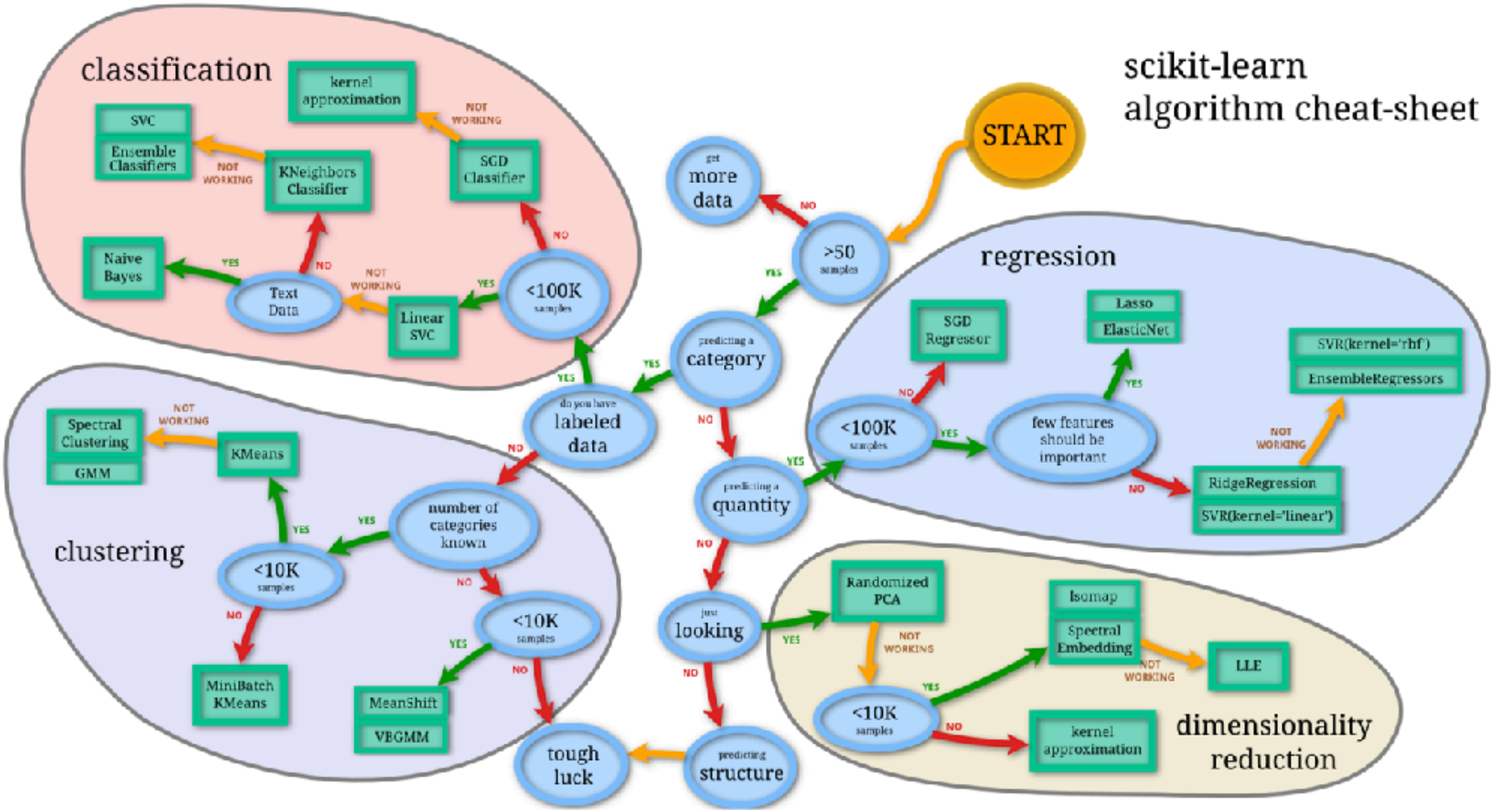
Organisation:

1. Anticipation de temps de calcul des différents modèles
2. Anticipation sur les ressources utilisées par la machine (GridSearch, hyperopt)

Compétences:

1. Maîtrise du python
2. Machine Learning

Axes d'améliorations



Merci de votre attention