AXA Data Challenge

Yellow – Zhengying Liu, Chia-Man Hung, Maxime Bellec

Summary

- Data pre-processing & Feature engineering
- Data visualization & Preliminary analysis
- Our approaches:
  - A first simple approach
  - A generalized version: Linear LinEx Regression
  - Random Forest
- Conclusion

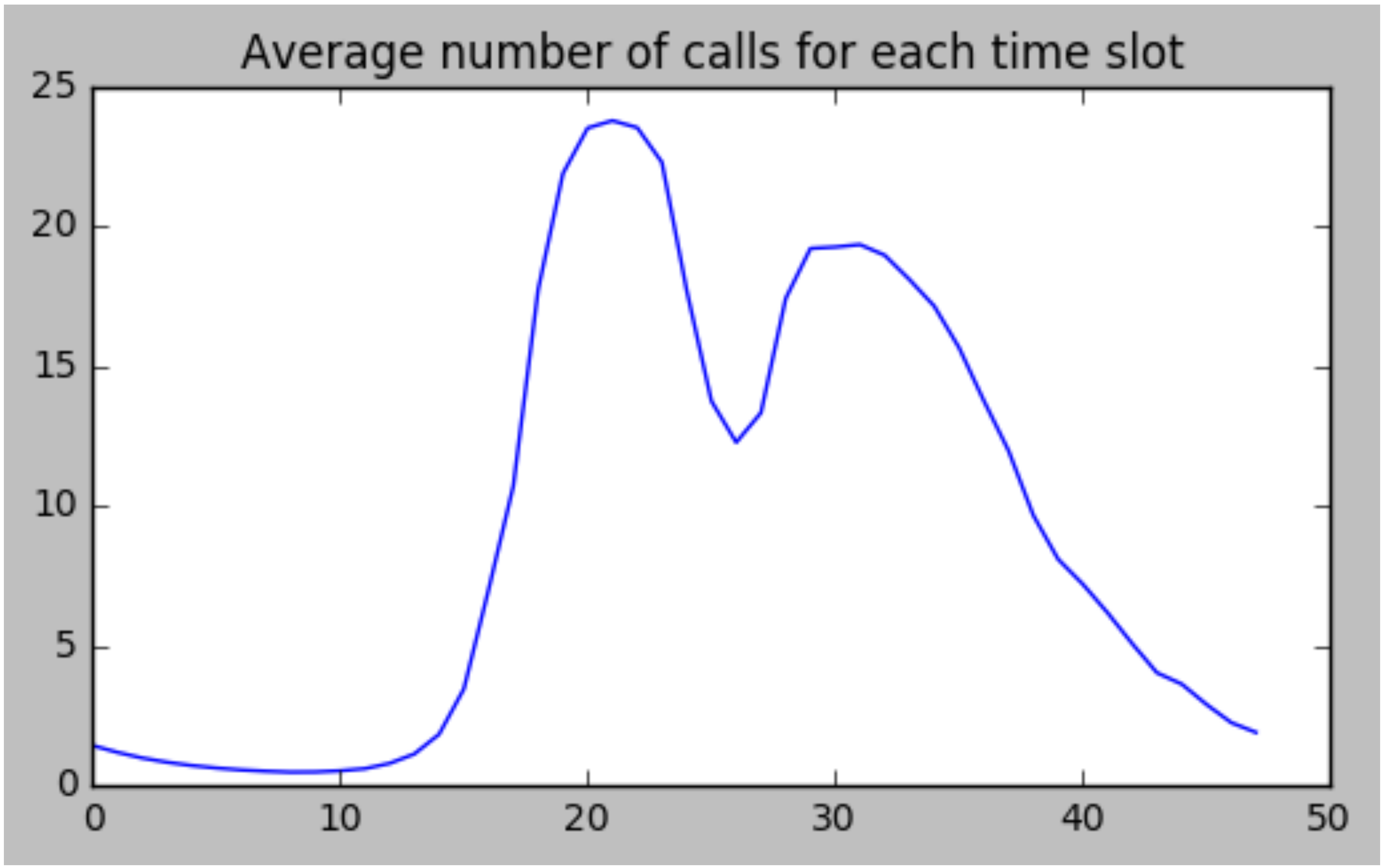
Data pre-processing

- All columns of the training data are not in test data
=> they are not useful, except for DATE
- Multiple columns for each (date, ASS_ASSIGNMENT)
=> sum the DSLP_RECEIVED_CALLS

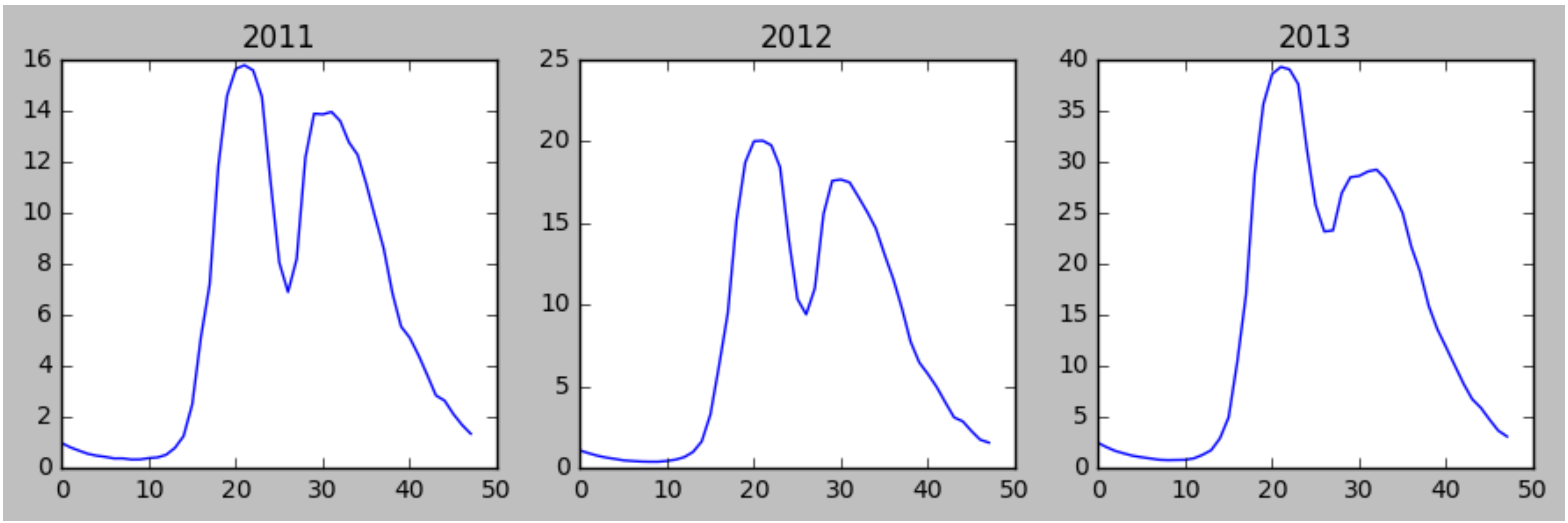
Feature engineering

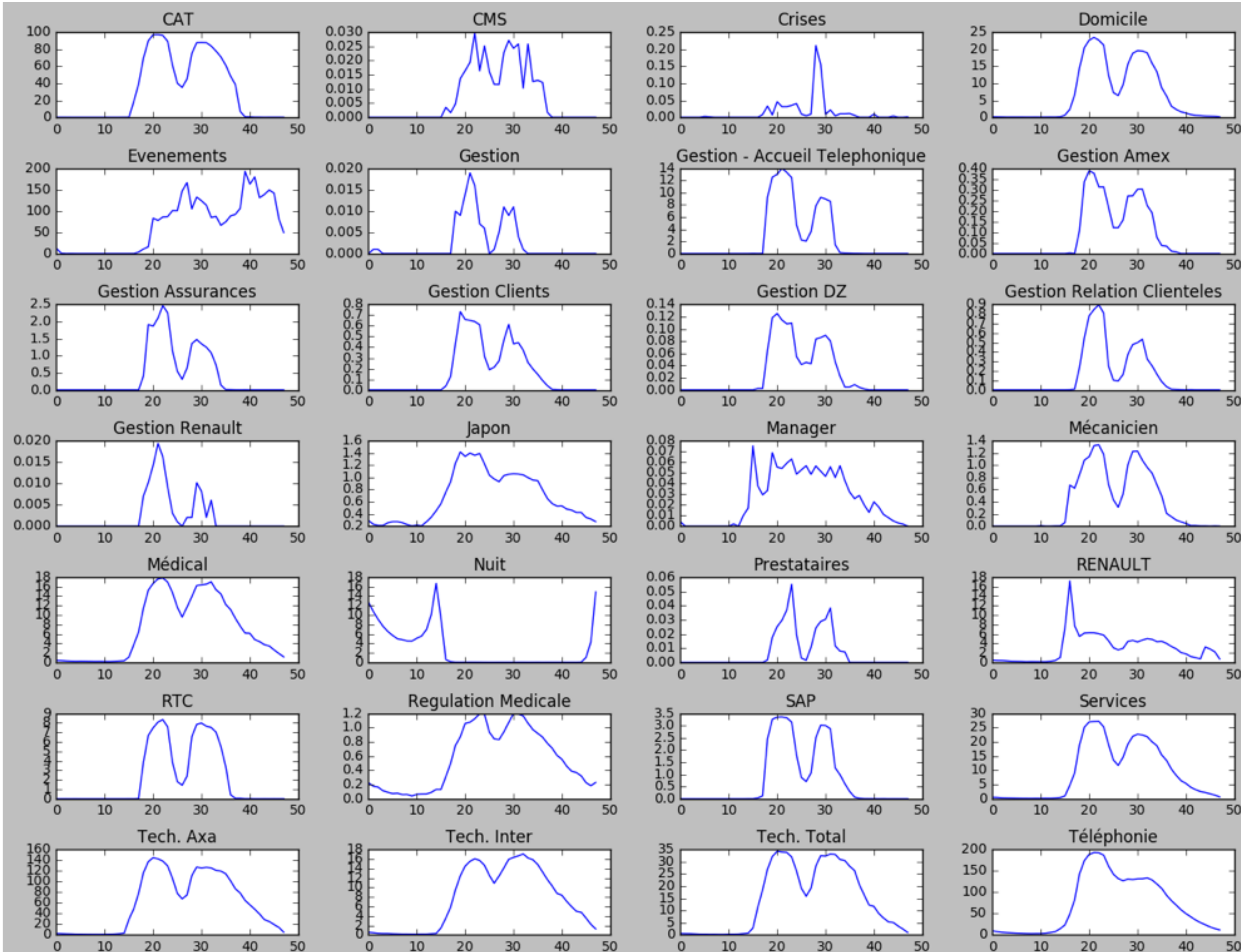
	DATE	ASS_ASSIGNMENT	CSPL_RECEIVED_CALLS	slot	dayofweek	month	year	day_off	day_after_day_off
0	2011-01-01	Crises	0	0	5	1	2011	True	False
1	2011-01-01	Domicile	0	0	5	1	2011	True	False
2	2011-01-01	Gestion	0	0	5	1	2011	True	False
3	2011-01-01	Gestion - Accueil Telephonique	0	0	5	1	2011	True	False
4	2011-01-01	Gestion Amex	0	0	5	1	2011	True	False

Data visualization



Data visualization




CAT
CMS
Crises
Domicile
Evenements
Gestion
Gestion - Accueil Telephonique
Gestion Amex
Gestion Assurances
Gestion Clients
Gestion DZ
Gestion Relation Clienteles
Gestion Renault
Japon
Manager
Mécanicien
Médical
Nuit
Prestataires
RENAULT
RTC
Regulation Medicale
SAP
Services
Tech. Axa
Tech. Inter
Tech. Total
Téléphonie

1st Approach – simple approach

- For a given ASS_ASSIGNMENT and weekly time slot, such as Tuesday 09:00-09:30, the CSPL_RECEIVED_CALLS are +- stationary

- The idea is to predict for each the « best stationary value » by minimizing the empirical loss

$$R(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n} \ell(y_i, \hat{y})$$

$$R'(\hat{y}) = \frac{1}{n}\sum_{i=1}^{n}(-\alpha e^{\alpha(y_i - \hat{y})} + \alpha) = 0$$

$$\implies \frac{1}{n}\sum_{i=1}^{n} e^{\alpha(y_i - \hat{y})} = 1$$

$$\implies \frac{1}{n}\sum_{i=1}^{n} e^{\alpha y_i} = e^{\hat{y}}$$

$$\implies \hat{y} = \log \frac{1}{n}\sum_{i=1}^{n} e^{\alpha y_i} = softmax(\alpha Y) - \log n$$

1^{st} Approach – simple approach

- Advantages:
  - Simple model
  - Explainable
  - Use the real loss function LinEx
- Disadvantages:
  - Too many parameters (about 10000 lines in predict_table)
  - Many of these parameters are correlated
  - Might have large variance
- Result:
  - 2.55 on the leader board

2nd approach – generalized version

- Our 1st approach can be regarded as a **linear regression** model
  - where the feature vector is “one-hot” encoding for all possible
  (ASS_ASSIGNMENT, slot, dayofweek)
  tuples.

- We extend it by considering more features (month, day_off)

- More general: consider **all possible combinations of all features**!

- => Linear LinEx Regression on Combined Features

Combined Feature matrix

- For the row (ASS_ASSIGNMENT, dayofweek, month, slot) = (Crises, 5, 1, 0)

- We associate a vector of 0 and 1's, where the 1's are in columns corresponding to

Example for (ASS_ASSIGNMENT, dayofweek, month, slot) = (Crises,5,1,0)

- (ASS_ASSIGNMENT=Crises, dayofweek=5, month=1, slot=0)
- (dayofweek=5, month=1, slot=0)
- (ASS_ASSIGNMENT=Crises, month=1, slot=0)
- (ASS_ASSIGNMENT=Crises, dayofweek=5, slot=0)
- (ASS_ASSIGNMENT=Crises, dayofweek=5, month=1)
- (month=1, slot=0)
- (dayofweek=5, slot=0)
- (dayofweek=5, month=1)
- (ASS_ASSIGNMENT=Crises, slot=0)
- (ASS_ASSIGNMENT=Crises, month=1)
- (ASS_ASSIGNMENT=Crises, dayofweek=5)
- (slot=0)
- (month=1)
- (dayofweek=5)
- (ASS_ASSIGNMENT=Crises)
- () (intercept)

A feature matrix of shape (1030829,147784)!
But each row has only **16 non-zero terms**
=> We use scipy.sparse.csr_matrix s

Linear linex regression

Loss function

$$\frac{1}{n}\sum_{i=1}^{n}\ell(y_i, x_i^\top\theta) + \frac{\lambda}{2}\|\theta\|_2^2$$

where

$$\ell(x, y) = LinEx(x, y) = \exp(\alpha(x - y)) - \alpha(x - y) - 1$$

Learning algorithm

- Stochastic gradient descent with variance reduction
- SVRG (Stochastic Variance Reduced Gradient) algorithm

SVRG

Input: starting point θ_0, learning rate $\eta > 0$
Put $\tilde{\theta}^1 \leftarrow \theta$
For $k = 1, 2, \ldots$ until convergence do

1. Put $\theta_0^k \leftarrow \tilde{\theta}_0^1$
2. Compute $\mu = \nabla f(\tilde{\theta}^k)$
3. For $t = 0, \ldots, m-1$:
    - Pick uniformly at random i in $\{1, \ldots, n\}$
    - Apply the step

$$\theta_{t+1}^k \leftarrow \theta_t^k - \eta(\nabla f_i(\theta_t^k) - \nabla f_i(\tilde{\theta}^k) + \mu)$$

Set

$$\tilde{\theta}^k \leftarrow \frac{1}{m} \sum_{t=1}^{m} \theta_t^k$$

Return last θ_t^k

2^{nd} approach – generalized version

- Advantages:
  - The loss function is convex
  - Very general, containing many possible approaches as special case
  - Explainable
  - Use the real loss function LinEx

- Disadvantages:
  - Many parameters (about 150000 of them)
  - Hard to optimize

- Result:
  - 1.99 on the leader board

3rd approach – Random Forest

- Use all the features in feature engineering
- No categorical values in sklearn -> one-hot encoding
- Remove Evenements and Gestion Amex
- Cross validation (80% training, 20% testing)
- Multiply by C = 2.4

3^{rd} approach – Random Forest

- Advantages:
  - Robust model
  - Existing library
  - Relatively good results

- Disadvantages:
  - Parameter tuning
  - Hard to use a custom loss function

- Result:
  - 1.175 on the leaderboard

Conclusion

- For prediction, when collecting data on past time, make sure this data will also be available for future times, otherwise they are not useful features for prediction

- A lot of features can be created on DATE and it can be enough when the data actually mostly depends on DATE