Machine learning: Business case

Présentation technique:

“ Rossmann Store Sales ”

Présenté par :

Ahmed BEJAOUI

Aymen DABGHI

Aymen MEJRI

Med Rostom GHARBI

Salma JERIDI

Plan

1. Dataset & Problématique
2. Preprocessing
3. Feature Engineering
4. Stratégie de validation
5. Training
6. Evaluation

1. Problématique & Dataset

Problématique :

Prévoir la vente quotidienne de 1115 magasins Rossman individuels situés dans toute l'Allemagne, 6 semaines à l'avance.

Impact de cette solution :

- Meilleure gestion des horaires du personnel.
- Prévoir suffisamment de temps pour que les directeurs de magasin se concentrent sur les clients et leurs équipes.
- Augmenter l'efficacité des employés.

Dataset :

Dans ce problème, on dispose de 3 datasets:

- **Train_set :** Représente l'historique des données de ventes quotidiennes de 1115 magasins à partir du 01/01/2013 au 31/07/2015. Cette partie des données compte environ 1 million d'entrées et comprend de multiples variables explicatives qui pourraient avoir un impact sur la vente.
- **Store_set :** Représente des informations supplémentaires sur les magasins.
- **Test_set :** Représente des données similaires à la Train_set (à l'exception de "customers" et "sales") pour les 6 semaines suivantes.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1017209 entries, 0 to 1017208
Data columns (total 9 columns):
Store            1017209 non-null int64
DayOfWeek        1017209 non-null int64
Date             1017209 non-null datetime64[ns]
Sales            1017209 non-null int64
Customers        1017209 non-null int64
Open             1017209 non-null int64
Promo            1017209 non-null int64
StateHoliday     1017209 non-null object
SchoolHoliday    1017209 non-null int64
```

Tain_set

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41088 entries, 0 to 41087
Data columns (total 8 columns):
Id               41088 non-null int64
Store            41088 non-null int64
DayOfWeek        41088 non-null int64
Date             41088 non-null datetime64[ns]
Open             41077 non-null float64
Promo            41088 non-null int64
StateHoliday     41088 non-null object
SchoolHoliday    41088 non-null int64
```

Test_set

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1115 entries, 0 to 1114
Data columns (total 10 columns):
Store                        1115 non-null int64
StoreType                    1115 non-null object
Assortment                   1115 non-null object
CompetitionDistance          1112 non-null float64
CompetitionOpenSinceMonth    761 non-null float64
CompetitionOpenSinceYear     761 non-null float64
Promo2                       1115 non-null int64
Promo2SinceWeek              571 non-null float64
Promo2SinceYear              571 non-null float64
PromoInterval                571 non-null object
```

Store_set

2. Preprocessing

Preprocessing

- **Stores :** On élimine les observations où les magasins sont fermés (open=0) car un magasin fermé génère un profit nul.

- **Sales :** On élimine les observations où les ventes sont nulles malgré l'ouverture des magasins car elles n'apportent rien à la prédiction.

Preprocessing

- **Missing values :**

➔ La table Store contient 6 variables explicatives avec des valeurs manquantes : *CompetitionDistance, CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2SinceWeek, Promo2SinceYear, PromoInterval*.

➔ La variable Open de la table Test présente des valeurs manquantes. On a supposé que ces magasins sont ouverts.

```
store_no_promo = store[store['Promo2'] == 0]
```

```
store_no_promo.shape
```

```
(544, 10)
```

```
store_no_promo.isnull().sum()
```

```
Store                         0
StoreType                     0
Assortment                    0
CompetitionDistance           0
CompetitionOpenSinceMonth     0
CompetitionOpenSinceYear      0
Promo2                        0
Promo2SinceWeek             544
Promo2SinceYear             544
PromoInterval               544
dtype: int64
```

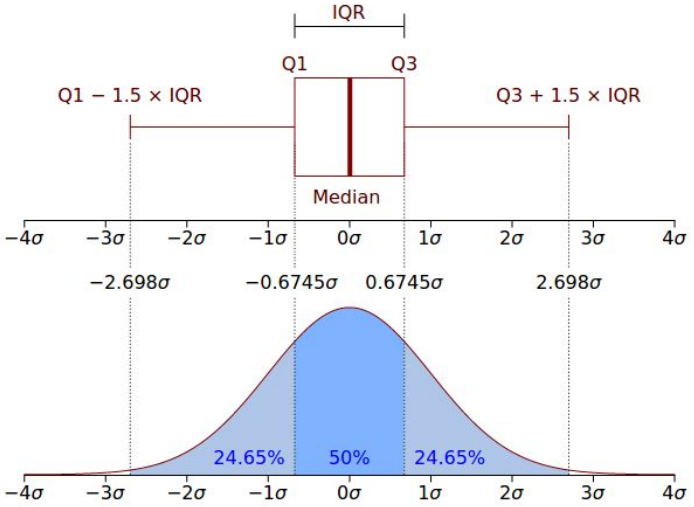
```
store.isnull().sum()
```

```
Store                         0
StoreType                     0
Assortment                    0
CompetitionDistance           0
CompetitionOpenSinceMonth     0
CompetitionOpenSinceYear      0
Promo2                        0
Promo2SinceWeek             544
Promo2SinceYear             544
PromoInterval               544
dtype: int64
```

Preprocessing

- **Outliers detection:**

On a procédé par la méthode de l'écart interquartile. Les observations inférieures à **Q1 - 3IQR** ou supérieures à **Q3 + 3IQR**, sont considérées comme des outliers extrêmes.



3. Feature Engineering

Feature engineering

- Fusionner la table Train avec la table Store.
- **Variables catégoriques :**
  - ➔ StoreType & Assortment: Stratégie de “One Hot Encoding”.
  - ➔ StateHoliday: Stratégie de “Réduction du nombre de modalités”. Les types sont regroupés en une seule variable binaire.
- **Variable Date :** On a créé les variables Day, Month,Year et WeekOfYear comme elles sont corrélées avec la variables “Sales”.

Feature engineering

- **Variable *CompetitionOpen*:** Variable créée à partir des variables *CompetitionOpenSinceYear* et *CompetitionOpenSinceMonth* dans le but d'exprimer la durée depuis laquelle la compétition existait en mois.

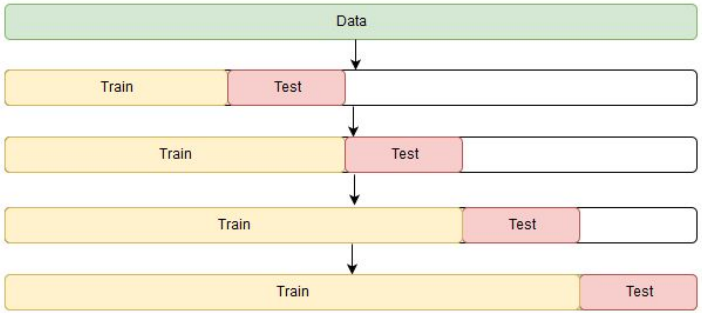
- **Variable *PromoOpen*:** Variable créée à partir des variables *Promo2SinceYear* et *Promo2SinceWeek* pour désigner la durée, en mois, de la promotion en cours.

Feature engineering

- **Variable *Promo2_ongoing_now*:** A partir des variables explicatives *Promo2*, *Promo2SinceYear*, *Promo2SinceWeek* et *PromoInterval,* on détermine si le magasin est en promotion 2 ou pas dans ce jour là.

4. Stratégie de validation

Validation Stratégie



- Nous avons testé 2 approches:
  - Effectuer une validation croisée en divisant la data en des parts égales.
  - Effectuer une validation croisée en divisant notre donnée en des paquets d'une durée de 2 mois

5. Training

Training

- Choix du modèle: XGBoost regressor

- Choix des paramètres du modèle: utilisation de la méthode de l'estimation bayésienne.

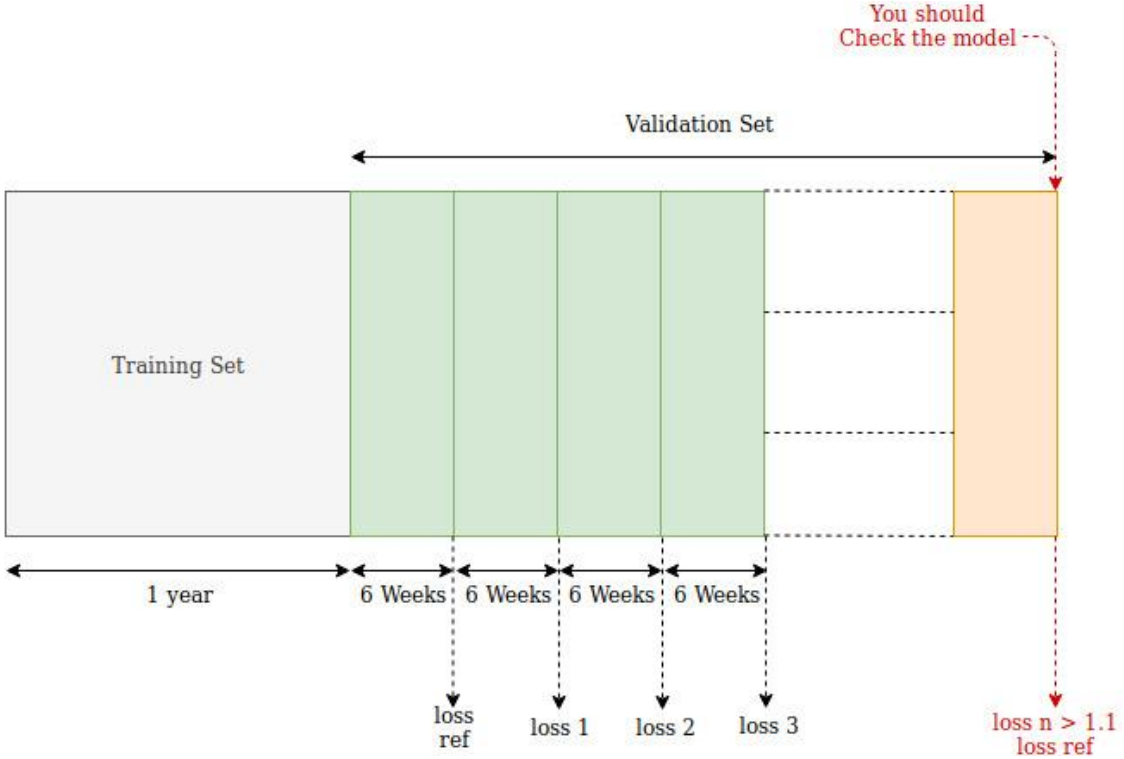
```
params= {
        'learning_rate': [0.1,0.2,0.4,0.8,1.0],
        'max_depth': [10,11,13,15,17],
        'n_estimators': [50, 100],
    }

model = mp.Regressor(XGBRegressor(n_jobs = 3),params)
```

```
model.model.best_params_
```

```
{'learning_rate': 0.4, 'max_depth': 15, 'n_estimators': 97}
```

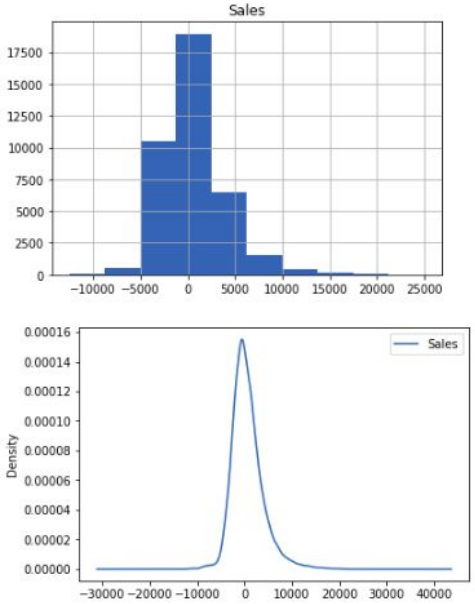
6. Evaluation

Evaluation


You should
Check the model
Validation Set
Training Set
1 year
6 Weeks
6 Weeks
6 Weeks
6 Weeks
loss
ref
loss 1
loss 2
loss 3
loss n > 1.1
loss ref

Evaluation

```
Period =  2014-02-15 00:00:00  to 2014-04-04 00:00:00 : the model is performing well
loss= 0.09924262639149437
====================================


Period =  2014-04-05 00:00:00  to 2014-05-23 00:00:00 : the model is performing well
loss= 0.09254830731114419
====================================


Period =  2014-05-24 00:00:00  to 2014-07-11 00:00:00 : the model is performing well
loss= 0.09594047296003255
====================================


Period =  2014-07-12 00:00:00  to 2014-08-29 00:00:00 : the model is performing well
loss= 0.10374475249197825
====================================


Period =  2014-08-30 00:00:00  to 2014-10-17 00:00:00 : the model is performing well
loss= 0.09028700792877344
====================================


From  2014-10-18 00:00:00 our model starts to perform badly
Warning: you may should review your model and see if it is still valid and operational !
```



Distribution des résidus.

Evaluation

Les causes potentielles:

- Changement de la distribution des données.
- Vente de nouvelles gammes de produits.
- Apparition de nouvelles lois (impôt, etc).
- Des gréves.

Solution:

➢ Réentraîner le modèle sur une durée bien déterminée en tenant compte des différents changements.