# Processing

December 1, 2024

## 1 Nettoyage et Pre processing des donées

[2]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression, Ridge, Lasso
```

[3]:
```
# Load dataset
data_path = 'sales_train.csv'
df = pd.read_csv(data_path)
```

[4]:
```
# Step 1: Data Exploration
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
Data columns (total 6 columns):
 #   Column          Dtype
---  ------          -----
 0   date            object
 1   date_block_num  int64
 2   shop_id         int64
 3   item_id         int64
 4   item_price      float64
 5   item_cnt_day    float64
dtypes: float64(2), int64(3), object(1)
memory usage: 134.4+ MB
```

nous utilisons la méthode info() pour examiner la structure du dataset. Le DataFrame contient 2,935,849 lignes et 6 colonnes, avec des types de données variés (entiers, flottants). Nous remarquons

que la colonne date est de type object, ce qui suggère qu'elle devrait être convertie en format datetime pour faciliter les analyses temporelles.

[5]:
```
df.describe()
```

[5]:

| | date_block_num | shop_id | item_id | item_price | item_cnt_day |
|---|---|---|---|---|---|
| count | 2.935849e+06 | 2.935849e+06 | 2.935849e+06 | 2.935849e+06 | 2.935849e+06 |
| mean | 1.456991e+01 | 3.300173e+01 | 1.019723e+04 | 8.908532e+02 | 1.242641e+00 |
| std | 9.422988e+00 | 1.622697e+01 | 6.324297e+03 | 1.729800e+03 | 2.618834e+00 |
| min | 0.000000e+00 | 0.000000e+00 | 0.000000e+00 | -1.000000e+00 | -2.200000e+01 |
| 25% | 7.000000e+00 | 2.200000e+01 | 4.476000e+03 | 2.490000e+02 | 1.000000e+00 |
| 50% | 1.400000e+01 | 3.100000e+01 | 9.343000e+03 | 3.990000e+02 | 1.000000e+00 |
| 75% | 2.300000e+01 | 4.700000e+01 | 1.568400e+04 | 9.990000e+02 | 1.000000e+00 |
| max | 3.300000e+01 | 5.900000e+01 | 2.216900e+04 | 3.079800e+05 | 2.169000e+03 |

Nous utilisons la méthode describe() pour obtenir des statistiques descriptives sur les colonnes numériques du dataset. Cela nous permet de repérer des valeurs extrêmes ou anormales, comme des prix négatifs dans item_price et des ventes négatives dans item_cnt_day, ce qui pourrait indiquer des erreurs de données.

[6]:
```
df.head()
```

[6]:

| | date | date_block_num | shop_id | item_id | item_price | item_cnt_day |
|---|---|---|---|---|---|---|
| 0 | 02.01.2013 | 0 | 59 | 22154 | 999.00 | 1.0 |
| 1 | 03.01.2013 | 0 | 25 | 2552 | 899.00 | 1.0 |
| 2 | 05.01.2013 | 0 | 25 | 2552 | 899.00 | -1.0 |
| 3 | 06.01.2013 | 0 | 25 | 2554 | 1709.05 | 1.0 |
| 4 | 15.01.2013 | 0 | 25 | 2555 | 1099.00 | 1.0 |

[7]:
```
df = df.dropna()
```

Nous utilisons dropna() pour supprimer les lignes contenant des valeurs manquantes (NaN).

[8]:
```
df['date'] = pd.to_datetime(df['date'], format='%d.%m.%Y', errors='coerce')
```

Nous convertissons la colonne date en format datetime afin de faciliter les analyses temporelles. Cette transformation permet de manipuler les dates de manière plus efficace et de résoudre les problèmes liés aux valeurs incorrectes

[9]:
```
df['day'] = df['date'].dt.day
df['month'] = df['date'].dt.month
df['year'] = df['date'].dt.year
```

[10]:
```
df
```

[10]:

| | date | date_block_num | shop_id | item_id | item_price | \ |
|---|---|---|---|---|---|---|
| 0 | 2013-01-02 | 0 | 59 | 22154 | 999.00 | |
| 1 | 2013-01-03 | 0 | 25 | 2552 | 899.00 | |

```
2       2013-01-05              0       25     2552      899.00
3       2013-01-06              0       25     2554     1709.05
4       2013-01-15              0       25     2555     1099.00
…              …              …        …        …          …
2935844 2015-10-10             33       25     7409      299.00
2935845 2015-10-09             33       25     7460      299.00
2935846 2015-10-14             33       25     7459      349.00
2935847 2015-10-22             33       25     7440      299.00
2935848 2015-10-03             33       25     7460      299.00

         item_cnt_day  day  month  year
0                 1.0    2      1  2013
1                 1.0    3      1  2013
2                -1.0    5      1  2013
3                 1.0    6      1  2013
4                 1.0   15      1  2013
…                 …  …    …   …
2935844           1.0   10     10  2015
2935845           1.0    9     10  2015
2935846           1.0   14     10  2015
2935847           1.0   22     10  2015
2935848           1.0    3     10  2015

[2935849 rows x 9 columns]
```

Nous ajoutons de nouvelles colonnes (day, month, year) en extrayant respectivement le jour, le mois et l'année de la colonne date. Cela permet d'enrichir le dataset avec des caractéristiques temporelles supplémentaires

[11]:
```
df = df[(df['item_cnt_day'] >= 0) & (df['item_price'] >= 0)].copy()
```

[12]:
```
df
```

[12]:
```
              date  date_block_num  shop_id  item_id  item_price  \
0       2013-01-02               0       59    22154      999.00
1       2013-01-03               0       25     2552      899.00
3       2013-01-06               0       25     2554     1709.05
4       2013-01-15               0       25     2555     1099.00
5       2013-01-10               0       25     2564      349.00
…              …              …        …        …          …
2935844 2015-10-10              33       25     7409      299.00
2935845 2015-10-09              33       25     7460      299.00
2935846 2015-10-14              33       25     7459      349.00
2935847 2015-10-22              33       25     7440      299.00
2935848 2015-10-03              33       25     7460      299.00

         item_cnt_day  day  month  year
```

```
0                 1.0    2      1  2013
1                 1.0    3      1  2013
3                 1.0    6      1  2013
4                 1.0   15      1  2013
5                 1.0   10      1  2013
…                 …  …    …   …
2935844           1.0   10     10  2015
2935845           1.0    9     10  2015
2935846           1.0   14     10  2015
2935847           1.0   22     10  2015
2935848           1.0    3     10  2015

[2928492 rows x 9 columns]
```

Nous filtrons les données pour ne conserver que les lignes où les valeurs de item_cnt_day (nombre d'articles vendus par jour) et item_price (prix de l'article) sont supérieures ou égales à zéro. Cela permet d'éliminer les valeurs aberrantes ou erronées, telles que les ventes ou les prix négatifs, qui n'ont pas de sens dans notre contexte

[13]:
```
from sklearn.preprocessing import LabelEncoder, StandardScaler
```

[14]:
```
df.loc[:, 'revenue'] = df['item_price'] * df['item_cnt_day']
```

[15]:
```
df
```

[15]:
```
              date  date_block_num  shop_id  item_id  item_price  \
0       2013-01-02               0       59    22154      999.00
1       2013-01-03               0       25     2552      899.00
3       2013-01-06               0       25     2554     1709.05
4       2013-01-15               0       25     2555     1099.00
5       2013-01-10               0       25     2564      349.00
…              …               …       …        …           …
2935844 2015-10-10              33       25     7409      299.00
2935845 2015-10-09              33       25     7460      299.00
2935846 2015-10-14              33       25     7459      349.00
2935847 2015-10-22              33       25     7440      299.00
2935848 2015-10-03              33       25     7460      299.00

         item_cnt_day  day  month  year  revenue
0                 1.0    2      1  2013   999.00
1                 1.0    3      1  2013   899.00
3                 1.0    6      1  2013  1709.05
4                 1.0   15      1  2013  1099.00
5                 1.0   10      1  2013   349.00
…                 …  …    …   …       …
2935844           1.0   10     10  2015   299.00
2935845           1.0    9     10  2015   299.00
```

```
2935846           1.0   14     10  2015   349.00
2935847           1.0   22     10  2015   299.00
2935848           1.0    3     10  2015   299.00

[2928492 rows x 10 columns]
```

Nous créons une nouvelle colonne revenue, qui représente le revenu généré par chaque transaction, en multipliant le prix de l'article (item_price) par le nombre d'articles vendus par jour (item_cnt_day).

[16]:
```
label_encoder_shop = LabelEncoder()
label_encoder_item = LabelEncoder()

df.loc[:, 'shop_id'] = label_encoder_shop.fit_transform(df['shop_id'])
df.loc[:, 'item_id'] = label_encoder_item.fit_transform(df['item_id'])
df['revenue'] = df['item_price'] * df['item_cnt_day']

# Appliquer StandardScaler pour la colonne numérique
scaler = StandardScaler()
df.loc[:, 'revenue'] = scaler.fit_transform(df[['revenue']])
df.loc[:, 'item_price'] = scaler.fit_transform(df[['item_price']])
```

[17]:
```
df
```

[17]:
```
              date  date_block_num  shop_id  item_id  item_price  \
0       2013-01-02               0       59    21788    0.063406
1       2013-01-03               0       25     2495    0.005519
3       2013-01-06               0       25     2497    0.474434
4       2013-01-15               0       25     2498    0.121293
5       2013-01-10               0       25     2507   -0.312861
…              …               …       …        …          …
2935844 2015-10-10              33       25     7263   -0.341805
2935845 2015-10-09              33       25     7314   -0.341805
2935846 2015-10-14              33       25     7313   -0.312861
2935847 2015-10-22              33       25     7294   -0.341805
2935848 2015-10-03              33       25     7314   -0.341805

         item_cnt_day  day  month  year   revenue
0                 1.0    2      1  2013 -0.029076
1                 1.0    3      1  2013 -0.046658
3                 1.0    6      1  2013  0.095765
4                 1.0   15      1  2013 -0.011494
5                 1.0   10      1  2013 -0.143358
…                 …  …    …   …        …
2935844           1.0   10     10  2015 -0.152149
2935845           1.0    9     10  2015 -0.152149
2935846           1.0   14     10  2015 -0.143358
```

```
2935847           1.0   22     10  2015 -0.152149
2935848           1.0    3     10  2015 -0.152149

[2928492 rows x 10 columns]
```

Nous convertissons les identifiants shop_id et item_id en valeurs numériques à l'aide de LabelEncoder, puis calculons le revenu de chaque transaction. Enfin, nous standardisons les colonnes revenue et item_price avec StandardScaler pour les rendre compatibles avec les modèles d'apprentissage automatique.

[18]:
```
df.to_csv('sales_processed', index=False)
```

[19]:
```
y = df['item_cnt_day']
X = df.drop(columns=['item_cnt_day'])
```

[20]:
```
y
```

[20]:
```
0          1.0
1          1.0
3          1.0
4          1.0
5          1.0
          …
2935844    1.0
2935845    1.0
2935846    1.0
2935847    1.0
2935848    1.0
Name: item_cnt_day, Length: 2928492, dtype: float64
```

[21]:
```
X
```

[21]:
```
              date  date_block_num  shop_id  item_id  item_price  day  month  \
0       2013-01-02               0       59    21788    0.063406    2      1
1       2013-01-03               0       25     2495    0.005519    3      1
3       2013-01-06               0       25     2497    0.474434    6      1
4       2013-01-15               0       25     2498    0.121293   15      1
5       2013-01-10               0       25     2507   -0.312861   10      1
…              …             …       …        …          …   …     …
2935844 2015-10-10              33       25     7263   -0.341805   10     10
2935845 2015-10-09              33       25     7314   -0.341805    9     10
2935846 2015-10-14              33       25     7313   -0.312861   14     10
2935847 2015-10-22              33       25     7294   -0.341805   22     10
2935848 2015-10-03              33       25     7314   -0.341805    3     10

         year   revenue
0        2013 -0.029076
```

```
1         2013 -0.046658
3         2013  0.095765
4         2013 -0.011494
5         2013 -0.143358
…          …        …
2935844   2015 -0.152149
2935845   2015 -0.152149
2935846   2015 -0.143358
2935847   2015 -0.152149
2935848   2015 -0.152149

[2928492 rows x 9 columns]
```

Nous préparons les variables explicatives (X) et la variable cible (y). La variable cible y est définie comme la colonne item_cnt_day (nombre d'articles vendus par jour), tandis que X contient toutes les autres colonnes, à l'exception de item_cnt_day, qui seront utilisées pour prédire cette cible.

[ ]: