Prédiction de ventes d’un article en grande surface

Objectif : Prédire la quantité d’article vendu

- ★ **Selon :**
  - le jour
  - la semaine
  - le mois
  -
- ★ **En fonction de :**
  - température
  - humidité
  - prix de l’article
  - Jour/Semaine/Mois de l’année
  - marge sur l’article
  - vente totale de la famille de l’article (optionnel)
  - vente total du magasin

Sommaire :

Présentation du dataset

Dataset initial

	PRODUCT_ID	TIME_FIELD	QUANTITY	SALE_AMOUNT	SALE_MARGIN	DELIVERY_MARGIN	STORE_SALES	FAM_SALES	TEMPERATURE	HUMIDITY_MAX_PERCENT
0	10101010203	02/01/2016	392.0	936.88	126.50847	32.18963	2.522783e+06	2593.8399	25	90
1	10101010203	03/01/2016	79.0	188.81	25.47283	6.50970	4.720384e+05	486.4500	25	85
2	10101010203	04/01/2016	399.0	953.61	128.75157	32.78047	1.987010e+06	2409.1900	25	86
3	10101010203	05/01/2016	456.0	1089.84	146.44023	38.16785	2.224192e+06	2542.1300	23	86
4	10101010203	06/01/2016	639.0	1527.21	205.12897	53.56521	2.501014e+06	2747.0900	23	86
5	10101010203	07/01/2016	526.0	1257.14	168.79133	44.15570	2.240269e+06	2489.4600	24	88
6	10101010203	08/01/2016	448.0	1070.72	143.46423	37.90509	2.401189e+06	2527.7000	23	89
7	10101010203	09/01/2016	408.0	975.12	130.86154	34.31405	2.576818e+06	2546.1002	23	88
8	10101010203	10/01/2016	85.0	203.15	27.27660	7.64498	4.512337e+05	497.1100	24	89
9	10101010203	11/01/2016	426.0	1018.14	137.05120	37.96759	1.907706e+06	2180.2200	24	90

```
database.shape
```

```
(1293, 10)
```

Nous avons les données pour 1293 jours ce qui est l'équivalent de plus de 3 ans de relevés

Les modification apportées à notre dataset :

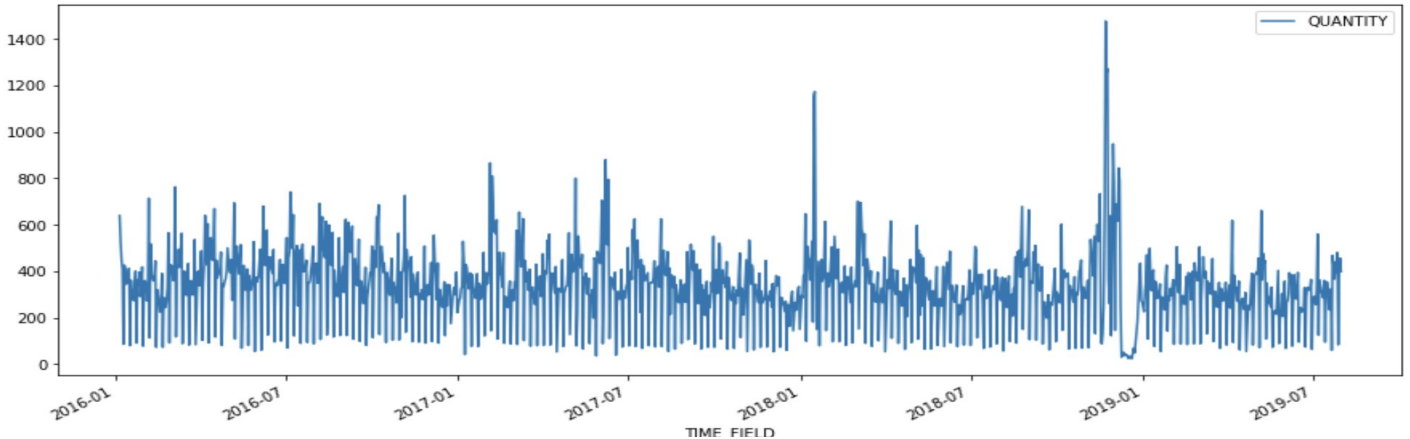
- Nous avons adapté l'index du dataset grâce à la colonne "Time-field"

- Pour gagner en compréhension et pour éviter une dépendance trop forte avec la quantité d'article vendu nous avons reformaté les colonnes "Sale_amont", "Sale_margin","Delivery_Margin" en deux colonnes que sont le prix unitaire de l'article et la marge unitaire faîte sur celui-ci.

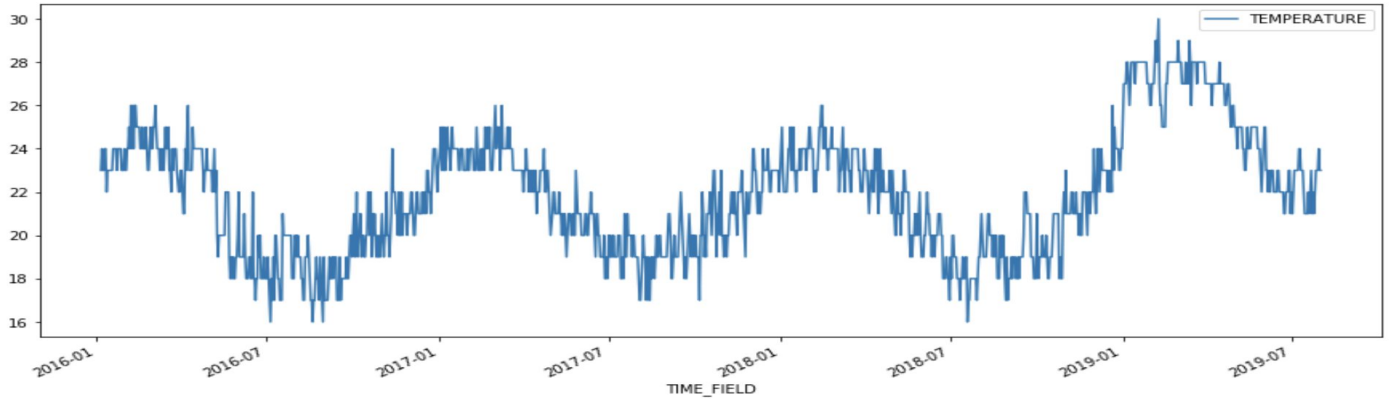
- Nous avons supprimé la colonne "id_article" car elle ne nous servira pas dans les prédiction (nous sommes en présence d'un seul article)

TIME_FIELD	QUANTITY	STORE_SALES	FAM_SALES	TEMPERATURE	HUMIDITY_MAX_PERCENT	Margin_Unitaire	Unit_price
2016-01-06	639.0	2.501014e+06	2747.0900	23.0	86.0	0.404842	2.39
2016-01-07	526.0	2.240269e+06	2489.4600	24.0	88.0	0.404842	2.39
2016-01-08	448.0	2.401189e+06	2527.7000	23.0	89.0	0.404842	2.39
2016-01-09	408.0	2.576818e+06	2546.1002	23.0	88.0	0.404842	2.39
2016-01-10	85.0	4.512337e+05	497.1100	24.0	89.0	0.410842	2.39
...	...	...	...	...	...	...	...
2019-07-27	481.0	2.159779e+06	2747.7600	23.0	87.0	0.206779	2.15
2019-07-28	83.0	4.095582e+05	527.4700	23.0	87.0	0.206779	2.15
2019-07-29	400.0	1.507739e+06	2247.8000	24.0	88.0	0.206779	2.15
2019-07-30	456.0	1.610319e+06	2538.1100	23.0	86.0	0.206779	2.15
2019-07-31	398.0	1.615483e+06	2219.2200	23.0	89.0	0.206779	2.15

Visualisation des données

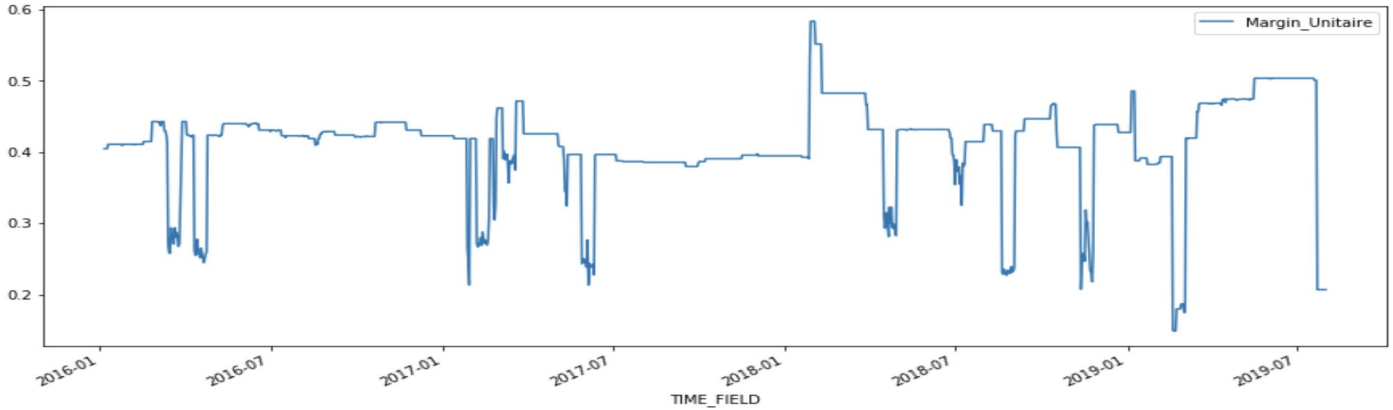


Quantité d'article vendu

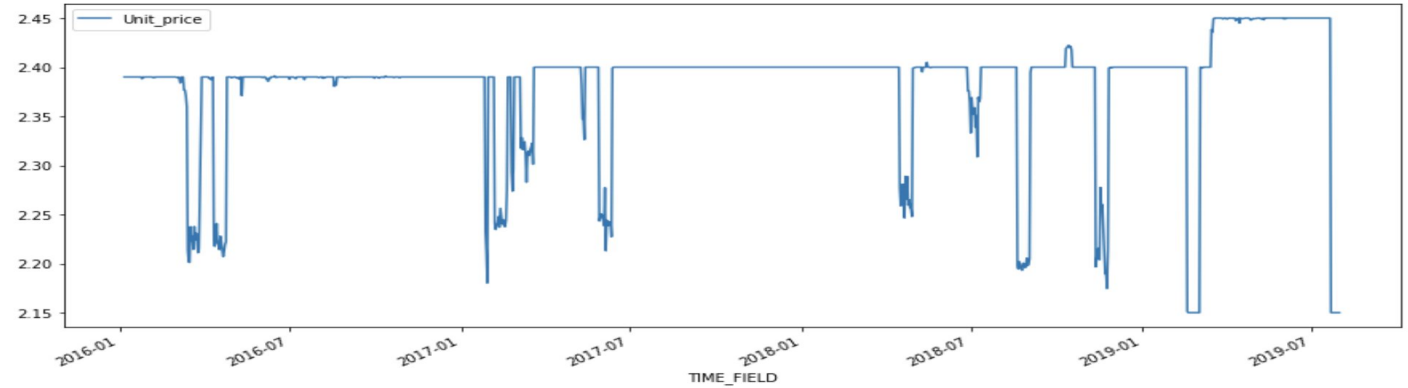


Température dans le magasin

Visualisation des données



Marge faite sur la vente d'un seul article



Prix unitaire de l'article

Etude des corrélations

```
#database.corr(method='pearson')
new_data.corr(method='spearman')
#database.corr(method='kendall')
```

	QUANTITY	STORE_SALES	FAM_SALES	TEMPERATURE	HUMIDITY_MAX_PERCENT	Margin_Unitaire	Unit_price
QUANTITY	1.000000	0.281544	0.922243	-0.167816	0.337582	-0.198660	-0.514162
STORE_SALES	0.281544	1.000000	0.337151	0.139853	0.128882	0.120947	-0.012173
FAM_SALES	0.922243	0.337151	1.000000	-0.169082	0.364809	-0.119392	-0.460759
TEMPERATURE	-0.167816	0.139853	-0.169082	1.000000	-0.102668	0.117607	0.112429
HUMIDITY_MAX_PERCENT	0.337582	0.128882	0.364809	-0.102668	1.000000	-0.190001	-0.388870
Margin_Unitaire	-0.198660	0.120947	-0.119392	0.117607	-0.190001	1.000000	0.473488
Unit_price	-0.514162	-0.012173	-0.460759	0.112429	-0.388870	0.473488	1.000000
day	-0.428146	-0.549472	-0.449557	-0.048218	-0.026441	-0.073219	0.016387
week_day	0.036327	-0.177287	0.019216	-0.170801	0.062056	-0.025440	0.049592
week	-0.106879	-0.001049	-0.131031	-0.632770	-0.099931	-0.041800	0.130852
month	-0.068688	0.048558	-0.091881	-0.638010	-0.099066	-0.030045	0.128828

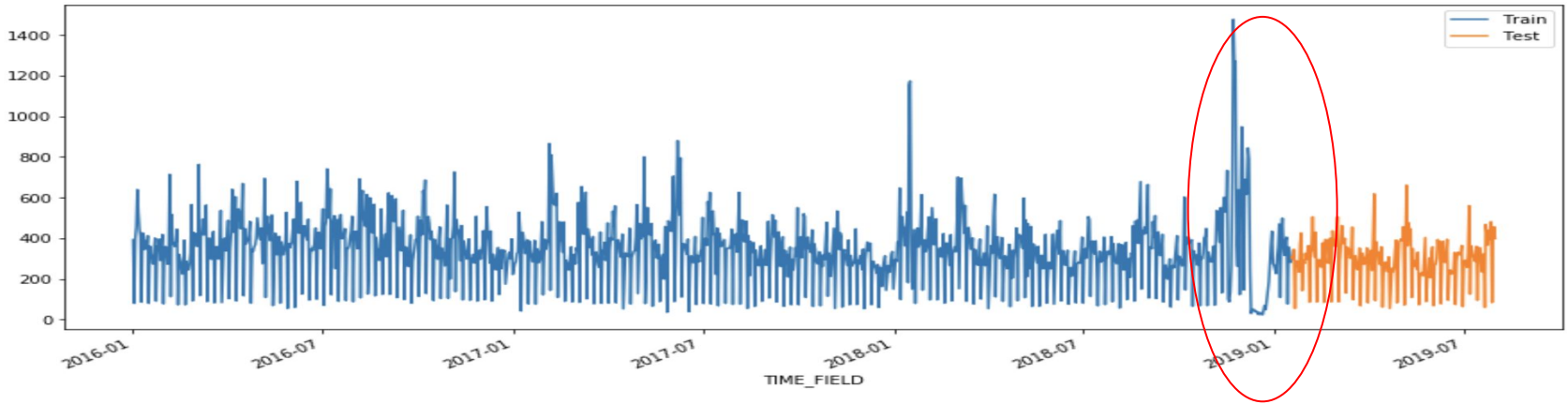
A première vue, les données corrélées sont la marge et le prix unitaire ce qui est logique. On remarque également une forte corrélation entre la quantité et le CA des familles dans cette catégorie -> intéressant pour la future prediction, on sait que cette variable aura un fort poids dans la prise de décision

Découpage des données

Nous avons testé plusieurs découpage de données et nous avons finalement choisis le suivant (85% train et 15% test) pour éviter la zone assez inhabituelle de décembre 2019 (entouré en rouge)

```
print(X_train.shape , y_train.shape , X_val.shape, y_val.shape)
```

```
(1099, 5) (1099,) (194, 5) (194,)
```



Mesure de l’erreur

Pour mesurer l’erreur on comptabilise le nombre de valeur prédite qui ont un pourcentage d’erreur supérieur à un seuil spécifié (5% si on veut voir les petites erreurs, 50% si on veut comptabiliser les grosses erreurs de prédictions)
On va aussi mesurer l’erreur en comparant la moyenne des prédictions sur la semaine (ou le mois) avec la moyenne des valeurs réelles.

```
def pourcentageErreur(yTrue,yPred):
    x=abs(yTrue-yPred)/yTrue*100
    return(round(x,2))

def CreateTableError(y_val_pred,y_val):
    tabPercent=[]
    for i in range (len(y_val_pred)):
        tabPercent.append(pourcentageErreur(y_val[i],y_val_pred[i]))
        if(i<10):
            print('valeur prédite: ',y_val_pred[i],'  valeure réelle :',y_val[i], '  pourcentage d erreur :', pourcentageErreur(y_val[i],y_val_pred[i]),'%' )
    return(tabPercent)

def OverXError(x,tabPercent):
    sumPercent=0
    for i in tabPercent:
        if(i>x):
           sumPercent+=1
    print('pourcentage d erreur supérieur à ',x,'% :',sumPercent/len(tabPercent)*100,'%')

def ErreurSur7jours(yPred,yTrue):
    sumPercent=[]
    sumSemainePred=[]
    sumSemaineTrue=[]
    x=1
    for i in range (len(yPred)):
        if(x<7):
            sumSemainePred.append(yPred[i])
            sumSemaineTrue.append(yTrue[i])
            x+=1
        else:
            x=1
            moyenneErreurSemaine=pourcentageErreur(mean(sumSemaineTrue),mean(sumSemainePred))
            sumPercent.append(moyenneErreurSemaine)
    return(sumPercent)
```

Partie 2: Prévision sans l’aspect temporel

Dans cette partie nous allons nous concentrer sur la prévision de la quantité d’articles vendu en fonction des différentes colonnes sans prendre en compte la dépendance des lignes de notre dataset les unes avec les autres.

Aussi nous n’allons pas prendre en compte la colonne “FAM_sales” representant la vente de la famille de l’article car elle est trop corrélée à la quantité d’article vendu et l'intérêt de l’étude serait réduit car si l’on connaît le CA lié à la famille de l’article, on connaît certainement la quantité d’article vendu.

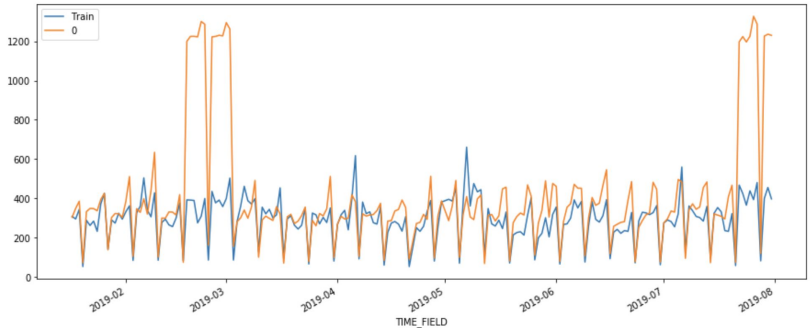
Les modèles de régression que nous avons utilisé parmis ceux proposés par sklearn sont les suivants :

- Réseaux de neurones
- Random Forest
- Linear regression
- KNeighborsRegressor

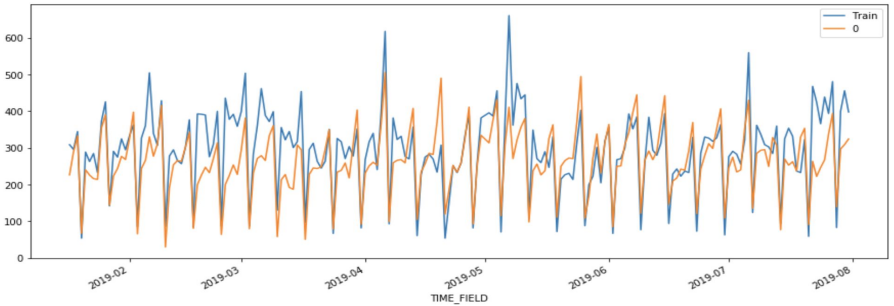
Graphs de prévisions

Ici un petit résumé graphique des prévisions obtenues avec les differents modèles (en jaune la prévision en bleu la valeur réelle)

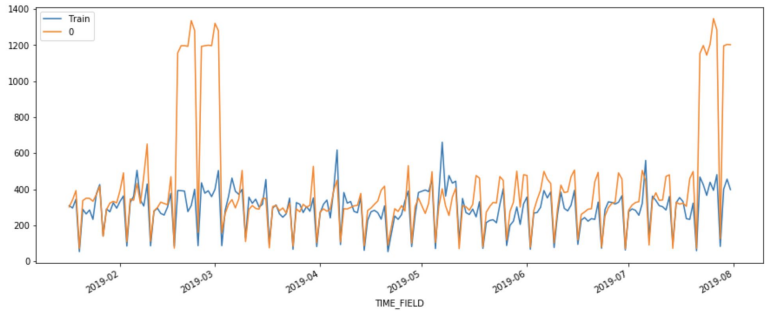
Neural Networks :



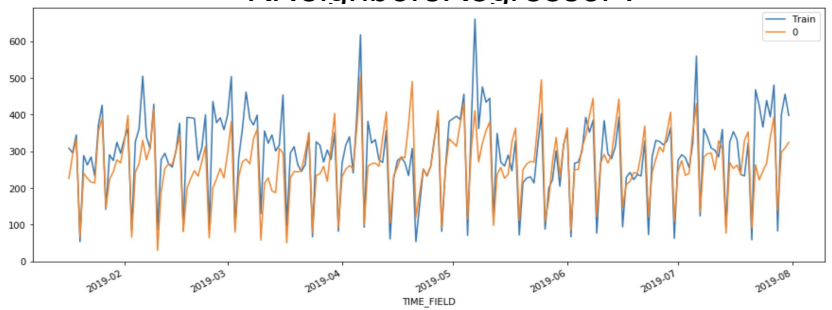
Linear Regression :



Random Forest :



KNeighborsRegressor :



Remarques :

- On remarque premièrement que les graphs de neurals networks et random forest sont très similaires et ont parfois des pics inattendus.
- La plupart des modèles suivent assez bien les tendances de vente de l’article.
- Les modèles de régressions linéaires et du “plus proche voisin” suivent très bien les tendances mais ont une valeur moyenne absolue toujours en dessous de la valeur réelle moyenne absolue.

Calcul du taux d’erreur

Exemple de résultats pour random forest

```
valeur prédite:  253.85574243259694   valeure réelle : 309.0   pourcentage d erreur : 17.85 %
valeur prédite:  312.6441456962617   valeure réelle : 296.0   pourcentage d erreur : 5.62 %
valeur prédite:  355.3768622090116   valeure réelle : 345.0   pourcentage d erreur : 3.01 %
valeur prédite:  72.90449009494628   valeure réelle : 54.0   pourcentage d erreur : 35.01 %
valeur prédite:  274.572544718421   valeure réelle : 289.0   pourcentage d erreur : 4.99 %
valeur prédite:  280.211922431508   valeure réelle : 263.0   pourcentage d erreur : 6.54 %
valeur prédite:  258.57712381228123   valeure réelle : 285.0   pourcentage d erreur : 9.27 %
valeur prédite:  256.3004380546023   valeure réelle : 233.0   pourcentage d erreur : 10.0 %
valeur prédite:  362.93407015930063   valeure réelle : 373.0   pourcentage d erreur : 2.7 %
valeur prédite:  407.70191942070744   valeure réelle : 426.0   pourcentage d erreur : 4.3 %
La moyenne de l erreur est de : 16.604845360824743 %
pourcentage d erreur supérieur à  50 % : 2.0618556701030926 %
pourcentage d erreur supérieur à  10 % : 61.855670103092784 %
La moyenne de l erreur sur la semaine est de : 7.8785185185185185 %
pourcentage d erreur supérieur à  50 % : 0.0 %
pourcentage d erreur supérieur à  10 % : 25.925925925925924 %
```

Remarques :

- La moyenne de l’erreur n’est pas si mauvaise avec une “accuracy” d’environ 84% ce qui n’était pas gagné en supprimant la colonne “FAM_Sales”
- Il y a tout de même plus de 60% des prévisions qui ont plus de 10% de différence avec la valeur réelle.
- Cependant on remarque très peu de “grosses erreurs” avec seulement 2% d’erreurs supérieur à 50%
- Enfin comme attendu, on remarque de meilleurs résultats lorsqu’on regroupe par semaine.

Amélioration des modèles :

Pour améliorer les modèles on a utilisé deux méthodes différentes :

1) Modifier manuellement les paramètres jusqu'à obtenir des meilleurs résultats en fonction de l'objectif souhaité (réduire l'erreur moyenne / éviter à tout prix les très grandes erreurs)
2) utilisé la méthode "GridSearchCV" de sklearn qui permet de trouver un meilleur modèle rapidement en spécifiant les paramètres à modifier et la méthode de calcul d'erreur souhaité : *(exemple pour neural networks)*

```
search_space = {'hidden_layer_sizes': [(200, ), (250, ), (64, 32, 16, ), (128, 64 , 32, ), (128,32,), (200,100,)],
                'alpha' : [0.001,0.005,0.02,0.025, 0.03] ,
                'learning_rate' : ['constant', 'invscaling', 'adaptive']}

model = GridSearchCV(estimator = neural_net , param_grid = search_space , cv = 4, verbose= 1, n_jobs = 3,
                     scoring= 'neg_mean_absolute_error' )
```

Comparaison des modèles :

Les modèles ont chacun une efficacité différente et sont plus ou moins intéressant en fonction de l'effet souhaité.
Dans le cas de l'erreur moyenne c'est le réseaux de neurones qui est le plus précis avec une accuracy à 86%. C'est aussi celui qui évite le plus les grosses marges d'erreurs avec seulement 2% des résultats qui ont un taux d'erreur supérieur à 50%.

Pour les erreurs inférieur à 10% c'est l'algorithme de random forest qui l'emporte.

Conclusion de la partie 1:

Les résultats obtenus avec les différents modèles ne sont pas extraordinaires avec des prévisions très peu précises (environ 60% d'erreurs lorsqu'on veut atteindre un taux d'erreur inférieur à 10% par rapport aux valeurs réelles) mais ils restent encourageant car ils suivent la tendance (croissance ou décroissance des courbes). Aussi on peut éviter les large erreurs avec 2% pour le réseaux de neurone par exemple.

Enfin les résultats sont utilisable si on se concentre sur la moyenne par semaine ou on atteint un taux d'erreur moyen de 93% et 0% d'erreur supérieur à 50%. (sur le mois on devrait obtenir encore plus de performance)

Ce qu’il reste à faire:

Pour améliorer nos résultats il y a plusieurs possibilités :

1) Éliminer les outliers (valeurs exceptionnelles) par exemple en décembre vers l’approche du 24 ou vers la fin de l’année on obtient des résultats très inattendu ce qui perturbe les modèles dans la phase d'entraînement et ce qui donne des mauvais résultats dans la phase de test.
2) Jouer sur le découpage des données entre la phase de test et la phase d’entraînement.
3) Jouer encore plus sur les paramètres des modèles pour les améliorer.
4) Procéder à une normalisation des données
5) Procéder à une augmentation des données (data augmentation) par des méthodes tel que le blanchiment des données par exemple.

Partie 2: Prévision avec l’aspect temporel

Néanmoins l'intérêt de cette étude reste réduit car on n’utilise pas l’aspect temporelle associé à notre dataset. En effet nous voulons pouvoir prédire la quantité d’article vendu à un jour j+1 en ayant les informations sur la vente de cette article les jours précédents.

Nous allons donc explorer quelque chose de nouveau qui est l’implémentation d’algorithme d’ia sur une time series.

Time series to supervised learning dataset

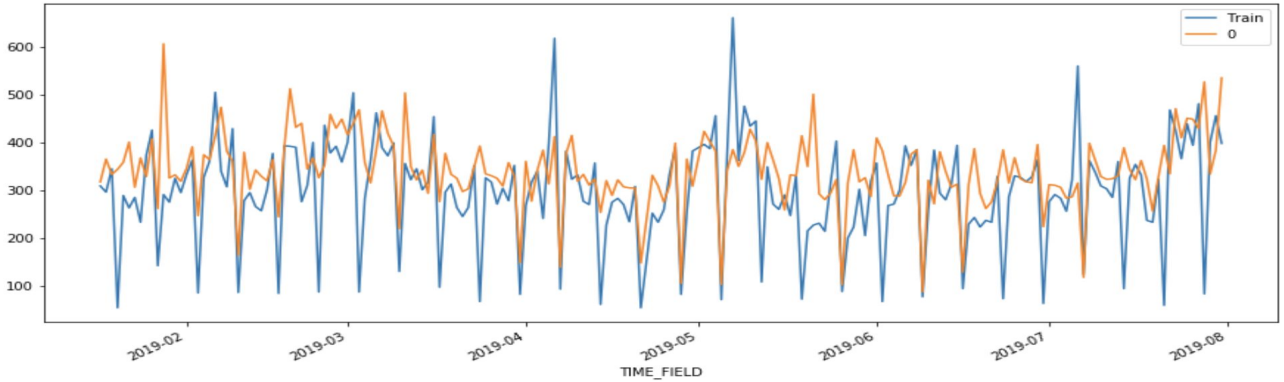
Pour cela nous devons tout d'abord adapter notre ensemble de données pour que le modèle d'ia puisse y être implémenter.
On va donc transformer notre time series en "supervised learning dataset". c'est à dire que chaque ligne de notre dataset contiendra les informations du jour précédent.

TIME_FIELD	QUANTITY	STORE_SALES	FAM_SALES	TEMPERATURE	HUMIDITY_MAX_PERCENT	Margin_Unitaire	Unit_price	day	week_day	week	month
2016-01-02	392.0	2.522783e+06	2593.8399	25.0	90.0	0.404842	2.39	2.0	5.0	53.0	1.0
2016-01-03	79.0	4.720384e+05	486.4500	25.0	85.0	0.404842	2.39	3.0	6.0	53.0	1.0
2016-01-04	399.0	1.987010e+06	2409.1900	25.0	86.0	0.404842	2.39	4.0	0.0	1.0	1.0
2016-01-05	456.0	2.224192e+06	2542.1300	23.0	86.0	0.404842	2.39	5.0	1.0	1.0	1.0
2016-01-06	639.0	2.501014e+06	2747.0900	23.0	86.0	0.404842	2.39	6.0	2.0	1.0	1.0

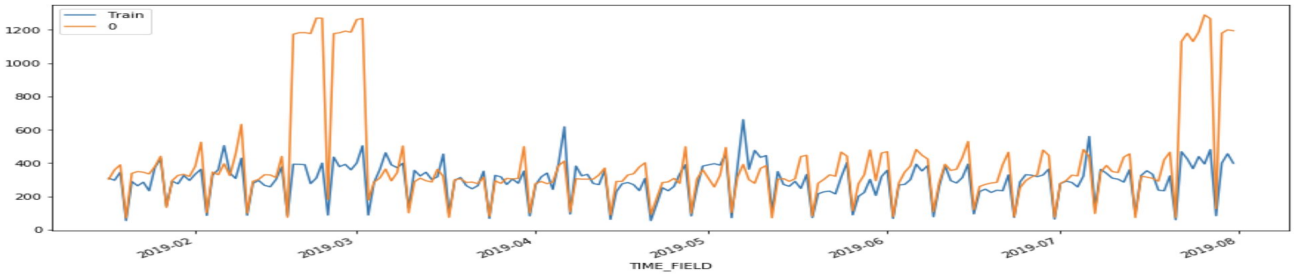
h	QUANTITY_day-1	STORE_SALES_day-1	TEMPERATURE_day-1	HUMIDITY_MAX_PERCENT_day-1	Margin_Unitaire_day-1	Unit_price_day-1	FAM_SALES-day-1
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
0	392.0	2.522783e+06	25.0	90.0	0.404842	2.39	2593.8399
0	79.0	4.720384e+05	25.0	85.0	0.404842	2.39	486.4500
0	399.0	1.987010e+06	25.0	86.0	0.404842	2.39	2409.1900
0	456.0	2.224192e+06	23.0	86.0	0.404842	2.39	2542.1300

Application et visualisation des prévisions :

Après avoir supprimé les lignes avec des valeurs manquantes on implémente les modèles étudié dans la partie 1.



Random Forest



Réseaux de neurone

Observations :

- On remarque un vrai décalage pour la prédiction avec random forest. Même si les courbes sont assez similaire, il semblerait que ce modèle reproduit approximativement la valeur du jour précédent, on s'attend donc à obtenir une moyenne de l'erreur sur la semaine très faible mais ce n'est pas un bon rapport de qualité car la prévision n'est pas bonne et reflète uniquement la valeur du jour précédent.
- Au contraire sur le graphe du réseau de neurone on observe une tendance vraiment suivit avec quelques pics d'erreurs inattendu à certains moment.

Calcul du taux d'erreur : *(exemple du réseau de neurone)*

```
valeur prédite:  251.27047401975108   valeure réelle : 309.0   pourcentage d erreur : 18.68 %
valeur prédite:  253.66229566193908   valeure réelle : 296.0   pourcentage d erreur : 14.3 %
valeur prédite:  309.58456958805584   valeure réelle : 345.0   pourcentage d erreur : 10.27 %
valeur prédite:  351.9279648524822   valeure réelle : 54.0   pourcentage d erreur : 551.72 %
valeur prédite:  74.43836127446743   valeure réelle : 289.0   pourcentage d erreur : 74.24 %
valeur prédite:  274.7661468396251   valeure réelle : 263.0   pourcentage d erreur : 4.47 %
valeur prédite:  276.1342675955526   valeure réelle : 285.0   pourcentage d erreur : 3.11 %
valeur prédite:  259.21874508516305   valeure réelle : 233.0   pourcentage d erreur : 11.25 %
valeur prédite:  254.22868248964335   valeure réelle : 373.0   pourcentage d erreur : 31.84 %
valeur prédite:  363.2899180313831   valeure réelle : 426.0   pourcentage d erreur : 14.72 %
La moyenne de l erreur est de : 81.18768041237114 %
pourcentage d erreur supérieur à  50 % : 28.865979381443296 %
pourcentage d erreur supérieur à  10 % : 78.35051546391753 %
La moyenne de l erreur sur la semaine est de : 6.973333333333333 %
pourcentage d erreur supérieur à  50 % : 0.0 %
pourcentage d erreur supérieur à  10 % : 14.814814814814813 %
```

On a un taux d'erreur global plutôt mauvais mais très impacté par les quelques pics très forts observé dans le graphe de prévision. Mis à part ces "pics d'erreurs", la qualité de la prévision reste très peu pertinente au niveau journalier avec presque 80% des prévisions qui ont un taux d'erreur supérieur à 10%.
Cependant on reste assez satisfait de la moyenne d'erreur de la prévision hebdomadaire.

Conclusion partie 2 et pistes d’améliorations :

La prévision temporelle réalisée n’est pas optimal mais on a un début de modèle intéressant car on obtient tout de même une bonne prévision des tendances. Pour améliorer notre modèle on a évidemment les stratégies d’amélioration imaginées en partie 1 mais on a aussi des pistes plus adéquates aux “time series” comme :

- L’ajout de variables temporelles (moyennes sur le mois ou la semaine)
- Certaines dates sont manquantes (exemple jour fériés / travaux), il faudrait pouvoir les compléter pour supprimer les discontinuités
- Enfin il existe des manières automatiques de transformer des time series en “supervised learning dataset” sur sklearn. (à voir si c’est efficace)

Partie 4: Application Flask

Le modèle d'ia que l'on a enregistré au préalable si on ne veut pas avoir à réentrainer notre ia (chronophage)

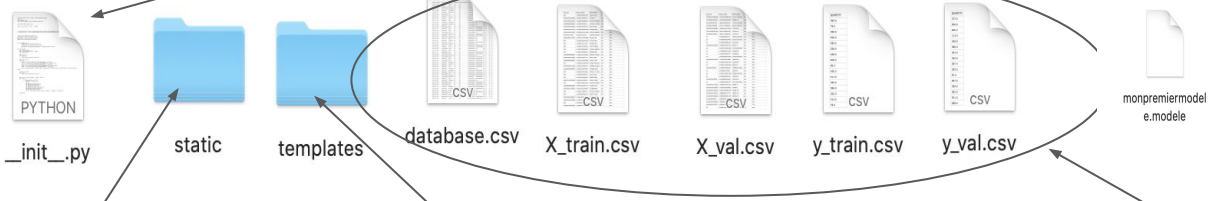
Notre Application Flask s'organise en 3 parties :

1) Une page d'accueil où est présenté l'utilité du projet
2) Une page de présentation de notre dataset
3) Une page de test d'un des modèles d'ia utilisé

Il s'organise de la façon suivante :

Le programme de lancement de l'application

Dans un dossier App :

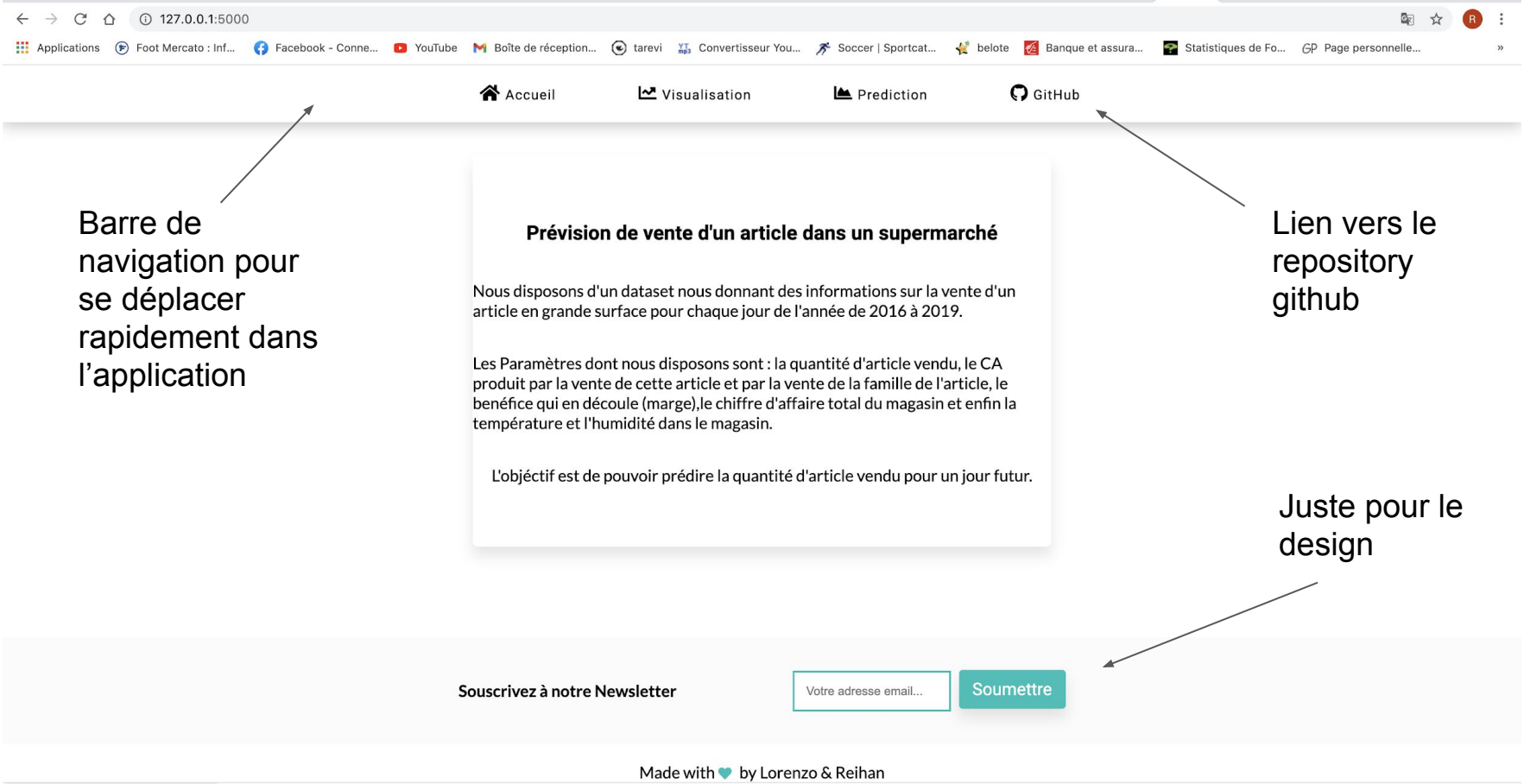


Ici les images et les styles css

dans ce dossier toutes les pages html utilisées

Les dossiers csv utilisé pour lancer notre algorithme

Page d’accueil :



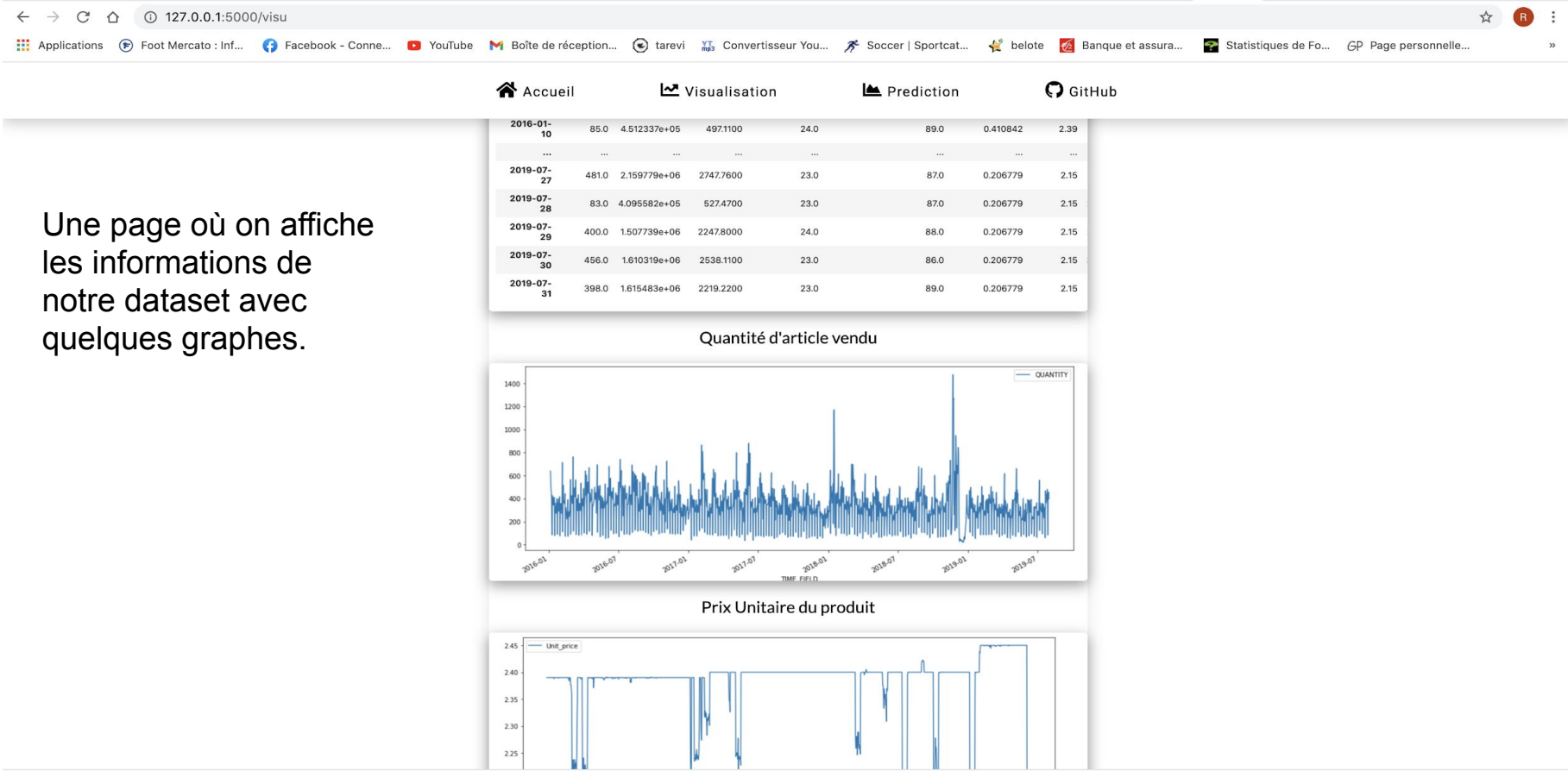
Barre de navigation pour se déplacer rapidement dans l’application

Lien vers le repository github

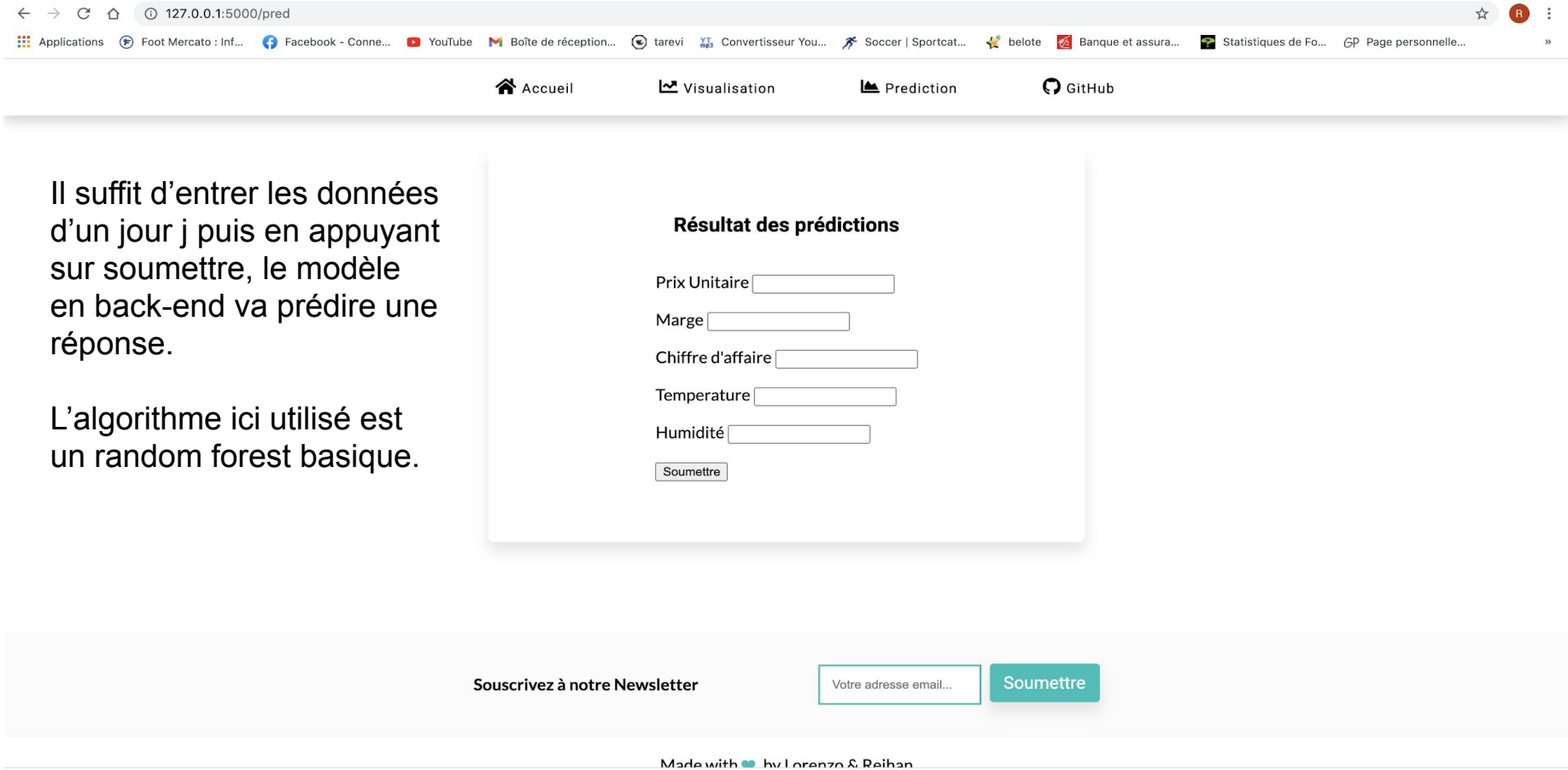
Juste pour le design

Page de visualisation des données

Une page où on affiche les informations de notre dataset avec quelques graphes.



Page de prédiction



Il suffit d'entrer les données d'un jour j puis en appuyant sur soumettre, le modèle en back-end va prédire une réponse.

L'algorithme ici utilisé est un random forest basique.