Introduction

L'identification et la réduction de la consommation d'énergie sont l'une des nombreuses voies nécessaires pour lutter contre le changement climatique. Une grande source de consommation d'énergie va dans le chauffage, le refroidissement et l'alimentation des bâtiments.

Le but de notre projet est de construire un modèle qui peut être utilisé pour **prédire avec précision la consommation d'énergie des bâtiments en se basant sur le cas d'utilisation, la taille, l'année de construction et les informations météorologiques locales**.

L'utilisation prévue peut être comparée à l'utilisation réelle pour comprendre la valeur ajoutée des améliorations énergétiques spécifiques, créant ainsi une meilleure compréhension de la façon dont ces améliorations énergétiques affectent la consommation d'énergie.

Comprendre les données

Pour le développement de ce modèle, l'équipe de ASHRAE fournit une Dataset contenant quatre types de données.

La première, Building Data, contient des informations concernant 1449 appartements, y inclus la localisation, la surface, le nombre d'étage et le cas d'utilisation.

La deuxième, Weather Data, fournit des informations météorologiques. Elle inclue 16 sites géographiques et leurs températures de l'air, leurs points de rosée, la masse de nuage, la précipitation, la pression au niveau de la mer ainsi que la direction et la vitesse du vent.

La troisième, Train Data, contient des relevés horaires de consommation des bâtiments pendant un an.

Et la quatrième, Test Data, contient les expectations de la consommation pour une durée de deux ans.

Les observations des deux dernières bases de données peuvent avoir une ou plusieurs mesures de consommation d'énergie, principalement des mesures de consommation d'électricité, d'eau chaude et froide et de vapeur.

Prétraitement des données

La visualisation de données nous permet de constater que les champs **year_built** et **Floor_count** contiennent plusieurs valeurs manquantes. Et comme le pourcentage de ces observations par rapport au nombre totale des observations dépasse 50%, on a préféré de supprimer ces deux champs.

```
[20] #Pourcentage des valeurs manquantes
     (BuildM_DF.isnull().sum()/ BuildM_DF.shape[0])*100
```

```
site_id          0.000000
building_id      0.000000
primary_use      0.000000
square_feet      0.000000
year_built      53.416149
floor_count     75.500345
dtype: float64
```

Aussi, les taux des valeurs manquantes des deux colonnes **Cloud_coverage** et **precip_depth_1_hr** sont respectivement de l'ordre de 49% et de 35%.

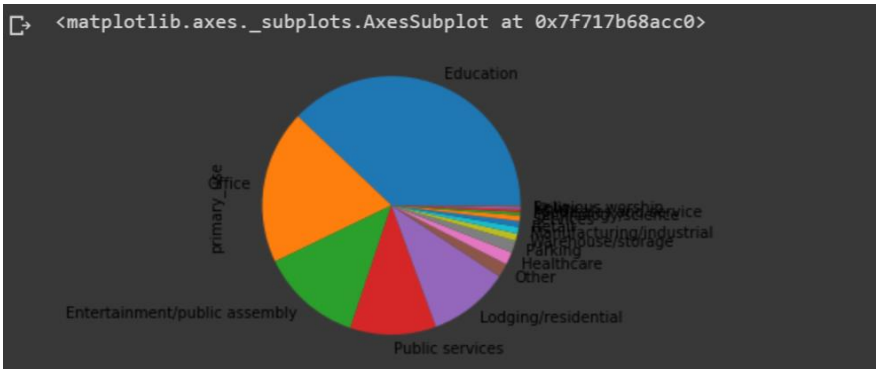
Ces valeurs sont remplacés par la moyennes des autres mesures de la meme colonne.

```
[26] (Weather_train_DF.isnull().sum()/Weather_train_DF.shape[0]) *100
```

```
site_id              0.000000
timestamp            0.000000
air_temperature      0.039350
cloud_coverage      49.489529
dew_temperature      0.080845
precip_depth_1_hr   35.979052
sea_level_pressure   7.596603
wind_direction       4.484414
wind_speed           0.217496
dtype: float64
```

La visualisation de la colonne **primary_use** montre une répartition not équitable des observations. Pour cela, on a regroupé les catégories les moins fréquents dans une autre catégorie.

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f717b68acc0>
```



Et comme la matrice de corrélation montre que **dew_temperature** est hautement corrélé avec **air_temperature** et que **wind_direction** est en forte corrélation avec **wind_speed**, on a choisi de négliger **dew_temperature** et **wind_direction**.

Les dates du champ **timestimp** sont décomposées pour donner naissance aux 3 nouveaux champs : **Month**, **Week** et **Day**.

De plus, pour éviter les erreurs de mémoire, on a modifié les types de données 64 bits à la plus petite taille possible en Numpy qui peut toujours représenter les informations.

```
[46] reduce_memory_usage(Train_DF)

     Mem. usage decreased to 652.57 Mb (71.7% reduction)
```

Phase d’apprentissage

Un réseau de neurones classique permet de gérer les cas où les expériences sont indépendantes les unes des autres. Lorsque les expériences sont sous la forme de séquences temporelles, une nouvelle structure a été inventée : les réseaux de neurones récurrents. Cette nouvelle structure introduit un mécanisme de mémoire des entrées précédentes qui persiste dans les états internes du réseau et peut ainsi impacter toutes ses sorties futures. Le réseau de neurones Long Short Term Memory (LSTM) est l’un des plus connu.

Le LSTM a été inventé pour résoudre le problème du vanishing and exploding gradient rencontré dans un réseau de neurones récurrent classique.

Comme n'importe quel neurone, les neurones LSTM sont généralement utilisés en couches. Dans notre modèle on a fait recours à 3 couches de neurones LSTM.

```
[ ] model3.summary()

Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_14 (LSTM)               (None, None, 256)         276480
_________________________________________________________________
dropout_11 (Dropout)         (None, None, 256)         0
_________________________________________________________________
batch_normalization_11 (Batc (None, None, 256)         1024
_________________________________________________________________
lstm_15 (LSTM)               (None, 128)               197120
_________________________________________________________________
batch_normalization_12 (Batc (None, 128)               512
_________________________________________________________________
dropout_12 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 129
=================================================================
Total params: 475,265
Trainable params: 474,497
Non-trainable params: 768
_________________________________________________________________
```

Evaluation

Pour évaluer le modèle, Kaggle propose comme métrique la racine de l'erreur quadratique moyenne (RMSE). En Statistique, l'erreur quadratique d'un estimateur est une mesure caractérisant la 'précision' de cet estimateur. Elle permet de comparer la performance de deux estimateurs.

Score RMSE