UNIVERSITE ABDELMALEK ESSAADI
FACULTE DES SCIENCES ET TECHNIQUES
DE TANGER
DEPARTEMENT GENIE INFORMATIQUE

Réseaux de neurones artificiels

ENCADRÉ PAR : M'hamed AIT KBIR

RÉALISÉ PAR : TLEMZI FATIMA
MOUDENNE LAMIAE
EL HRIKI NOUHAILA

Table de matières :

Table de figures :

Remerciement :

Au terme de ce projet, nous exprimons nos vifs remerciements à Monsieur M'HAMED AIT KBIR pour son encadrement, assistance, tout le temps précieux qu'il a octroyé et pour ses conseils qui sont bien utiles pour la rédaction de ce projet.

Introduction :

Le machine learning (ML) est un sous-ensemble de l'intelligence artificielle (IA) qui est axé sur la création de systèmes qui apprennent ou améliorent les performances en fonction des données qu'ils traitent.

L'intelligence artificielle est un terme large qui désigne des systèmes ou des machines imitant l'intelligence humaine. Le machine learning et l'IA sont souvent abordés ensemble, et les termes sont parfois utilisés de manière interchangeable, mais ils ne veulent pas dire la même chose. Une distinction importante est que, même si l'intégralité du machine learning repose sur l'intelligence artificielle, cette dernière ne se limite pas au machine learning.

Objectif :

L'objectif principal du projet est la réalisation de deux applications pour implémenter les solutions apportées par les modèles de réseaux de neurones artificiels aux deux problèmes en bas

Problématiques :

1. **Le premier problème :** Utilisation des réseaux multi-couches pour l'analyse des sentiments des phrases issues d'une base d'exemples qui contient des phrases étiquetées avec un sentiment positif ou négatif.

2. **Le deuxième problème :** Utilisation des réseaux RBF (Radial basis function) pour l'approximation de la consommation

énergétique d'une maison à partir d'un ensemble de données de prévision énergétique des appareils électroménagers.

Première application :

Description des données :

Ces données proviennent du fameux site IMDB dédié à l'évaluation des œuvres cinématographiques, elles contiennent presque 748 commentaires sur des films qui se divisent entre des points de vue positifs et d'autres négatifs. Le but c'est d'utiliser les réseaux de neurones multicouches pour entrainer notre modèle à travers un apprentissage supervisé.

Notre data se compose essentiellement des reviews qui sont rédigés en anglais qui représente nos inputs (sachant bien que ces données ont besoin de quelques prétraitements avant de commencer le travail avec l'algorithme choisi) et les sentiments qui sont exprimés avec des 1 pour les commentaires positifs et des 0 pour les commentaires négatifs et qui représentent nos outputs.

Prétraitements possibles :

La première étape réalisée dans cette partie c'est l'élimination des caractères spéciaux comme les chiffres, @, /, et #...

Ensuite stopwords sert à enlever les mots inutiles qui n'influent pas le sens (positif ou négatif) de la phrase.

Lemmatize(word) élimine les dérivés d'un mot pour ne pas avoir la répétition.

D'un côté on a construit deux tableaux qui devise notre dataset en deux parties ; une pour les mots positifs et l'autre pour les mots négatifs.

Pour chaque mot on a calculé la fréquence pour savoir combien de fois ce mot se répète dans notre dataset.

D'un autre côté on a construit une liste des listes qui contient soit les mots positifs dans chaque phrase ou les mots négatifs dans chaque phrase.

Après les données de notre dataset seront deviser en deux parties ; une pour l'entrainement et l'autre pour le test.

[0,1] signifie que la probabilité qui domine c'est que le mot est négatif.

[1,0] signifie que la probabilité qui domine c'est que le mot est positif.

(0 et 1 signifient le pourcentage d'être positif et négatif successivement).

La dernière partie c'est celle de l'initialisation et d'apprentissage, dans cette partie on fait l'entrainement avec 500 itérations jusqu'à avoir une valeur d'erreur minimale.

On ajout qu'il y a une comparaison entre les outputs de la fonction predict et les vraies valeurs. C'est généralement le même résultat.

Par exemple :

```
[124. 68.]
1
Sortie prédite :
[[0.80084482]
[0.19480131]])
```

Accuracy_score : Enfin, calculons la précision, c'est à dire la probabilité de classer les chiffres correctement.

On calcule cette précision sur l'échantillon de test, qui n'a pas été utilisé dans l'entraînement du réseau.

(ex : `0.81 `)

accuracy : il indique le pourcentage de bonnes prédictions. C'est un très bon indicateur parce qu'il est très simple à comprendre.

$$\text{Accuracy} = \frac{Vrai\ positif + Vrai\ négatif}{Total}$$

Algorithme d'apprentissage

Le perceptron multicouche (PMC)

Le perceptron multicouche (PMC) est un réseau composé de couches successives. Une couche est un ensemble de neurones n'ayant pas de connexion entre eux. Une couche d'entrée lit les signaux entrants, un neurone par entrée xj, une couche en sortie fournit la réponse du système. Selon les auteurs, la couche d'entrée qui n'introduit aucune modification n'est pas comptabilisée. Une ou plusieurs couches cachées participent au transfert. Dans un perceptron, un neurone d'une couche cachée est connecté en entrée à chacun des neurones de la couche précédente et en sortie à chaque neurone de la couche suivante.

Après avoir appliquer l'algorithme PMC, on a créé un constructeur de la classe MultiLayerPerceptron qui prend en paramètre les quatre couches contenant les nœuds ([2 5 5 2]).

La première couche contient deux nœuds : un nœud lié à la fréquence négative et un nœud lié à la fréquence positive.

La deuxième et la troisième couche contient cinq nœuds.

La dernière couche contient deux nœuds qui donnent finalement le pourcentage de bonnes prédictions.

Résultat :

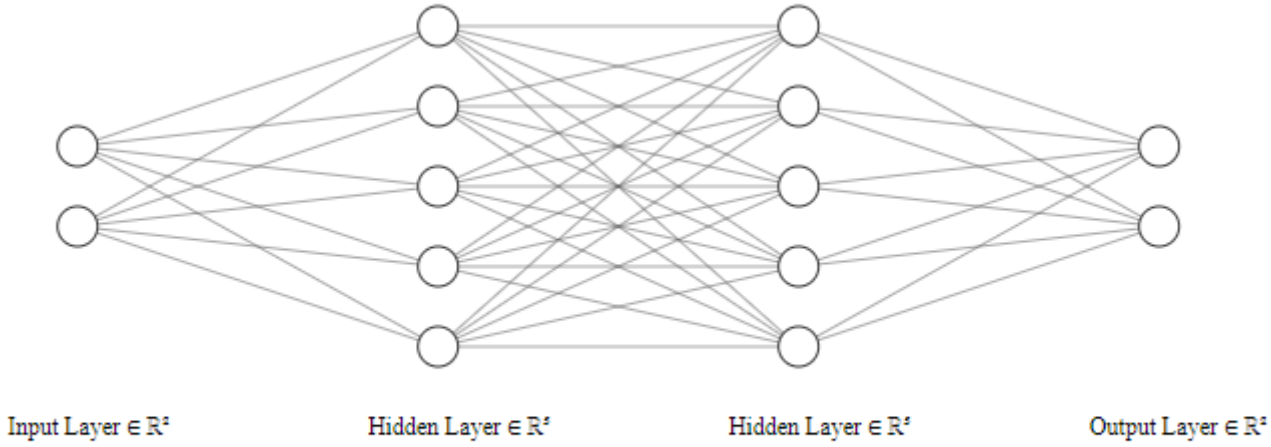


Figure1 : La structure du network

Deuxième application :

Description des données :

DataFrame :

	date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	...	T9	RH_9	T_out
0	2016-01-11 17:00:00	60	30	19.890000	47.596667	19.200000	44.790000	19.790000	44.730000	19.000000	...	17.033333	45.5300	6.600000
1	2016-01-11 17:10:00	60	30	19.890000	46.693333	19.200000	44.722500	19.790000	44.790000	19.000000	...	17.066667	45.5600	6.483333
2	2016-01-11 17:20:00	50	30	19.890000	46.300000	19.200000	44.626667	19.790000	44.933333	18.926667	...	17.000000	45.5000	6.366667
3	2016-01-11 17:30:00	50	40	19.890000	46.066667	19.200000	44.590000	19.790000	45.000000	18.890000	...	17.000000	45.4000	6.250000
4	2016-01-11 17:40:00	60	40	19.890000	46.333333	19.200000	44.530000	19.790000	45.000000	18.890000	...	17.000000	45.4000	6.133333
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
19730	2016-05-27 17:20:00	100	0	25.566667	46.560000	25.890000	42.025714	27.200000	41.163333	24.700000	...	23.200000	46.7900	22.733333
19731	2016-05-27 17:30:00	90	0	25.500000	46.500000	25.754000	42.080000	27.133333	41.223333	24.700000	...	23.200000	46.7900	22.600000
19732	2016-05-27 17:40:00	270	10	25.500000	46.596667	25.628571	42.768571	27.050000	41.690000	24.700000	...	23.200000	46.7900	22.466667
19733	2016-05-27 17:50:00	420	10	25.500000	46.990000	25.414000	43.036000	26.890000	41.290000	24.700000	...	23.200000	46.8175	22.333333
19734	2016-05-27 18:00:00	430	10	25.500000	46.600000	25.264286	42.971429	26.823333	41.156667	24.700000	...	23.200000	46.8450	22.200000

Press_mm_hg	RH_out	Windspeed	Visibility	Tdewpoint	rv1	rv2
733.5	92.000000	7.000000	63.000000	5.300000	13.275433	13.275433
733.6	92.000000	6.666667	59.166667	5.200000	18.606195	18.606195
733.7	92.000000	6.333333	55.333333	5.100000	28.642668	28.642668
733.8	92.000000	6.000000	51.500000	5.000000	45.410389	45.410389
733.9	92.000000	5.666667	47.666667	4.900000	10.084097	10.084097
...	...	...	...	...	...	...
755.2	55.666667	3.333333	23.666667	13.333333	43.096812	43.096812
755.2	56.000000	3.500000	24.500000	13.300000	49.282940	49.282940
755.2	56.333333	3.666667	25.333333	13.266667	29.199117	29.199117
755.2	56.666667	3.833333	26.166667	13.233333	6.322784	6.322784
755.2	57.000000	4.000000	27.000000	13.200000	34.118851	34.118851

Figure2 : DataFrame

L'ensemble de données est à 10 min pendant environ 4,5 mois. Les conditions de température et d'humidité de la maison ont été surveillées avec un réseau de capteurs sans fil ZigBee (c'est comme le Bluetooth pour l'IoT). Chaque nœud sans fil a transmis les conditions de température et d'humidité environ 3,3 min. Ensuite, les données sans fil ont été moyennées sur des périodes de 10 minutes. Les données énergétiques ont été enregistrées toutes les 10 minutes avec des compteurs d'énergie m-bus. La météo de la station météorologique de l'aéroport la plus proche (aéroport de Chievres, Belgique) a été téléchargée à partir d'un ensemble de données publiques de Reliable Prognosis (rp5.ru) et fusionnée avec les ensembles de données expérimentaux à l'aide de la colonne date et heure. Deux variables aléatoires ont été incluses dans l'ensemble de données pour tester les modèles de régression et pour filtrer les attributs non prédictifs (paramètres).

Informations sur les attributs :

- Date heure année-mois-jour heure:minute:second.
- Appareils, consommation d'énergie en Wh.
- Lumières, consommation d'énergie des luminaires de la maison en Wh.
- T1, Température dans la cuisine, en Celsius.
- RH_1, Humidité dans la cuisine, en %.
- T2, Température dans le salon, en Celsius.
- RH_2, Humidité dans le salon, en %.
- T3, Température dans la buanderie.
- RH_3, Humidité dans la buanderie, en %.
- T4, Température dans le bureau, en Celsius.
- RH_4, Humidité dans le bureau, en %.
- T5, Température dans la salle de bain, en Celsius.
- RH_5, Humidité dans la salle de bain, en %.
- T6, Température à l'extérieur du bâtiment (côté nord), en Celsius.
- RH_6, Humidité à l'extérieur du bâtiment (côté nord), en %.
- T7, Température salle de repassage, en Celsius.
- RH_7, Humidité salle de repassage, en %.

- T8, Température chambre ado 2, en Celsius.
- RH_8, Humidité chambre ado 2, en %.
- T9, Température chambre parents, en Celsius.
- RH_9, Humidité dans la chambre des parents, en %.
- To, Température extérieure (depuis la station météo de Chievres), en Celsius.
- Pression (depuis la station météo de Chievres), en mm Hg.
- RH_out, Humidité extérieure (depuis la station météo de Chievres), en %.
- Vitesse du vent (depuis la station météo de Chievres), en m/s.
- Visibilité (depuis la station météo de Chievres), en km energy_target.
- Point de rosée (depuis la station météo de Chievres), °C.
- rv1, Variable aléatoire 1, non dimensionnelle.
- rv2, Variable aléatoire 2, non dimensionnelle.

Prétraitements possibles :

Les prétraitements qu'on va effectuer concerne l'élimination des données bruitées et puisque on a plusieurs variables qui sont inutiles à notre étude, on va utiliser la corrélation pour se débarrasser des attributs qui ne vont servir à rien, avant de passer à entrainer le modèle avec l'algorithme d'apprentissage choisi pour cet exercice.

On utilise le diagramme de corrélation pour sélectionner les meilleures caractéristiques pour le modèle. Voici notre carte des chaleurs des variables :

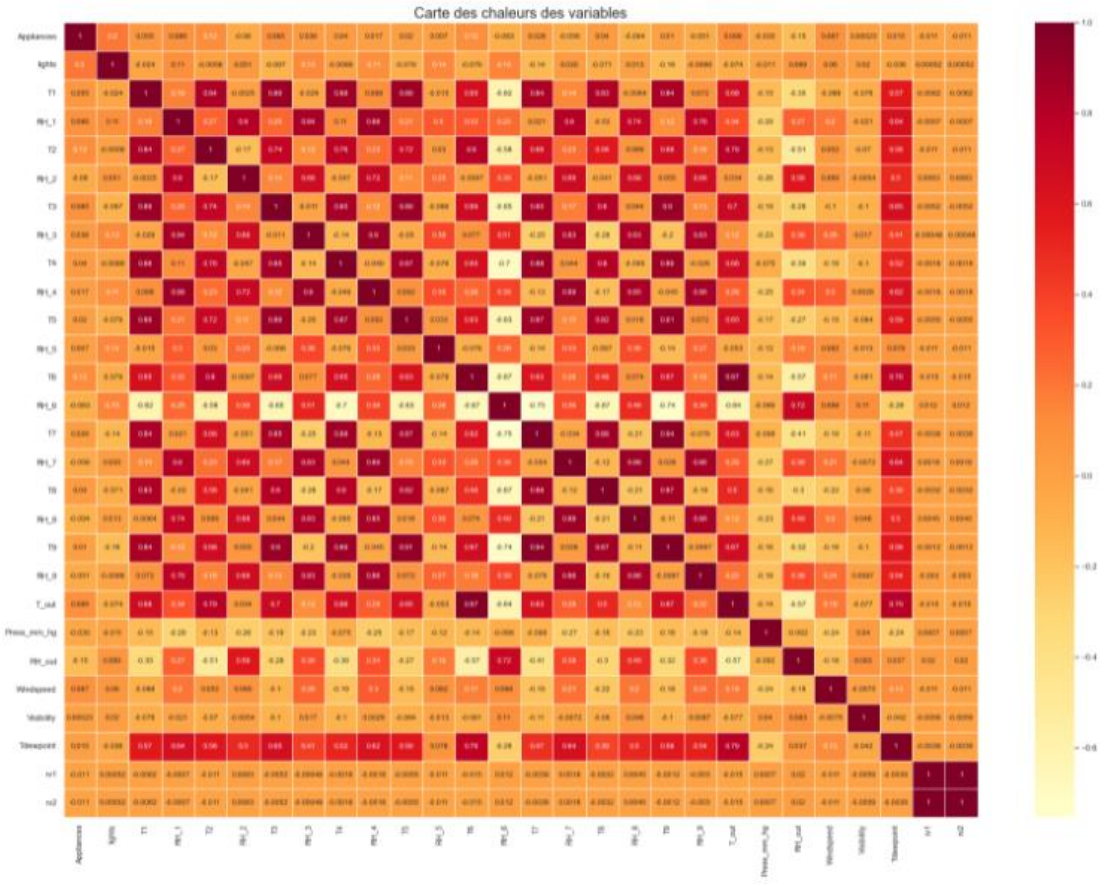


Figure 3 : Carte des chaleurs des variables avant la corrélation.

Pour la sélection des fonctionnalités, nous mettons en œuvre l'élimination à l'aide d'une corrélation soutenue par l'intuition métier.

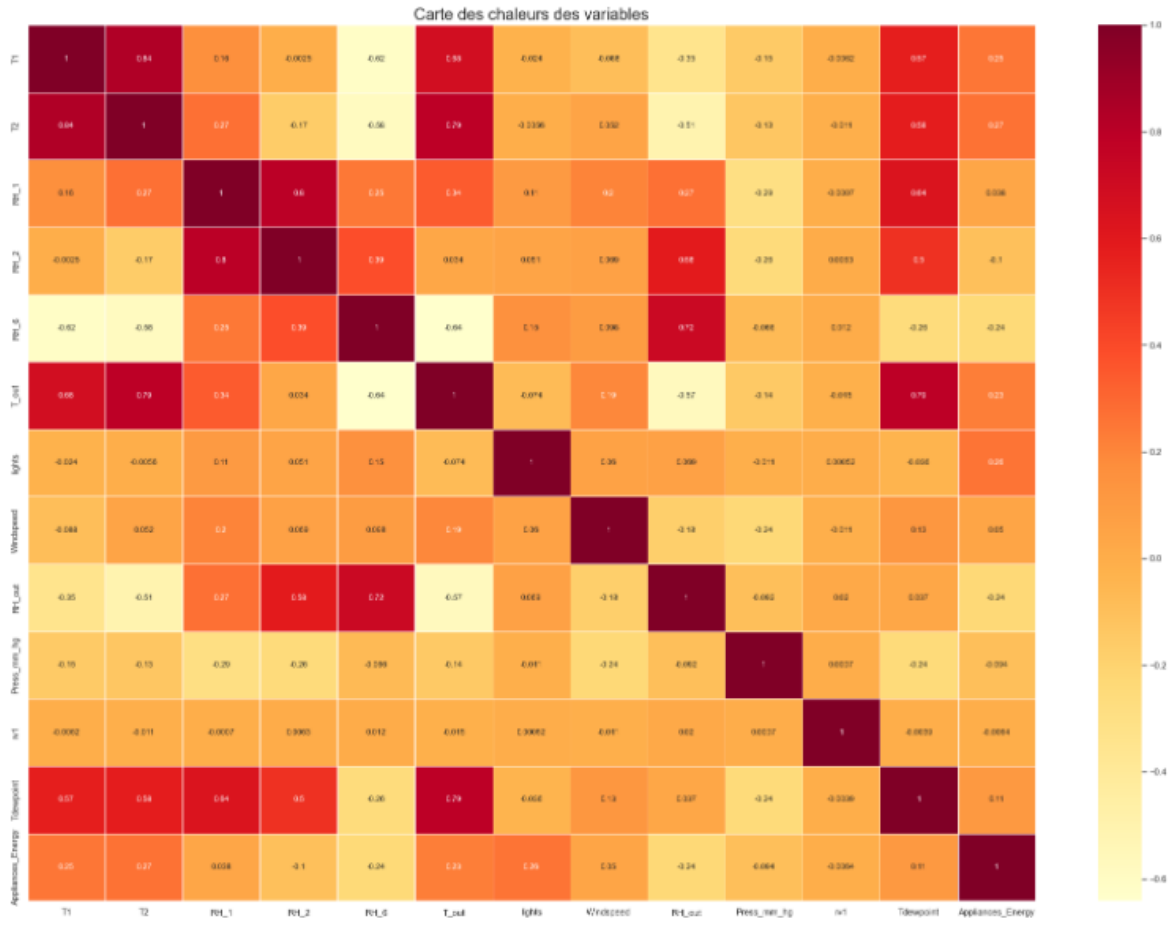


Figure 4 : Carte des chaleurs des variables après la corrélation.

Vous trouverez ci-dessous les 10 caractéristiques finales ainsi que la variable cible "Appliances" que nous avons sélectionnées pour l'analyse ultérieure.

```
df_final = df_Energy[['T1',
                      'T2',
                      'RH_1',
                      'RH_2',
                      'RH_6',
                      'T_out',
                      'lights',
                      'Windspeed',
                      'RH_out',
                      'Appliances',
                      'Press_mm_hg',
                      'rv1',
                      'Tdewpoint']]

df_final['Appliances_Energy'] = np.where(df_final['Appliances']>= 60, 1, 0)
df_final.drop(columns=['Appliances'],axis=1,inplace=True)
```

Figure 5 : Les caractéristiques dont on va concentrer notre étude.

Algorithme d'apprentissage

Méthode 1 : Réseaux de neurones à base de noyaux

Support Vector Machines (SVM) :

Supposons que nous ayons un ensemble de points appartenant à deux classes distinctes. Nous voulons séparer ces deux classes d'une manière qui nous permet d'attribuer correctement tous les nouveaux points futurs à une classe ou à l'autre

L'algorithme SVM tente de trouver un hyperplan qui sépare ces deux classes avec la marge la plus élevée possible. Si les classes sont entièrement linéairement séparables, une marge ferme peut être utilisée. Sinon, cela nécessite une marge souple.

Un noyau est une fonction qui prend le problème non linéaire d'origine et le transforme en un problème linéaire dans l'espace de dimension supérieure.

Radial basis function kernel :

Dans l'apprentissage automatique, le noyau de fonction de base radiale, ou noyau RBF, est une fonction de noyau populaire utilisée dans divers algorithmes d'apprentissage noyaux. En particulier, il est couramment utilisé dans la classification des machines à vecteurs de support.

RBF est le noyau par défaut utilisé dans l'algorithme de classification SVM de sklearn et peut être décrit avec la formule suivante :

$$K(x, x') = e^{-\gamma ||x - x'||^2}$$

Où gamma peut être réglé manuellement et doit être> 0. La valeur par défaut du gamma dans l'algorithme de classification SVM de sklearn est :

$$\gamma = \frac{1}{n\,features * \sigma^2}$$

Brièvement :

$||x - x'||^2$ is the squared Euclidean distance between two feature vectors (2 points).

Gamma is a scalar that defines how much influence a single training example (point) has.

(C'est la distance euclidienne carrée entre deux vecteurs d'attributs)

Nous utiliserons les bibliothèques suivantes :

```
from sklearn.model_selection import train_test_split,KFold  #biblio destinée à l'apprentissage automatique
from sklearn import preprocessing
import seaborn as sns  #biblio de visualisation incroyable pour le traçage de graphiques statistiques en Python
from sklearn.metrics import confusion_matrix,classification_report
%matplotlib inline
```

Figure 6 : Les bibliothèques utilisées

Ensuite, nous obtenons les données de prévision énergétique des appareils électroménagers

D'après le fichier csv :

```
df_Energy=pd.read_csv(r'C:\Users\Home\Downloads\energydata_complete.csv')
df_Energy
```

Figure 7 : Le fichier csv

Fonction SVM :

```
def runModelSVM(k,xTrain,yTrain,xTest,yTest):
    from sklearn.svm import SVC

    svc_clf = SVC(kernel=k)
    svc_clf.fit(xTrain,yTrain)
    y_pred=svc_clf.predict(xTest)

    print(' Kernel: ',k)
    print('Train score: {:.4f} %'.format(svc_clf.score(xTrain, yTrain)*100))
    print('Test score: {:.4f} %'.format(svc_clf.score(xTest, yTest)*100))
    print('')
    print('Classification Report:')
    print(classification_report(yTest,y_pred))
    print('Confusion Matrix:')
    print(confusion_matrix(yTest,y_pred))
```

Figure 8 : La fonction SVM

Et on appelle la fonction avec :

```
runModelSVM('rbf',xTrain,yTrain,xTest,yTest)
```

Result :

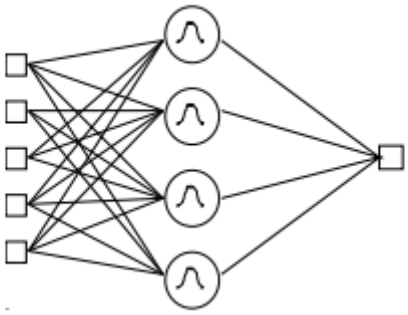
```
********** Result for Train-Test Split **********

 Kernel:  rbf
Train score: 67.9238 %
Test score: 68.1135 %

Classification Report:
              precision    recall  f1-score   support

           0       0.65      0.30      0.41      2192
           1       0.69      0.91      0.78      3729

    accuracy                           0.68      5921
   macro avg       0.67      0.60      0.59      5921
weighted avg       0.67      0.68      0.64      5921

Confusion Matrix:
[[ 651 1541]
 [ 347 3382]]
********** Result for Cross-Validation **********

 Kernel:  rbf
Train score: 67.8921 %
Test score: 73.2894 %

Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00       527
           1       0.73      1.00      0.85      1446

    accuracy                           0.73      1973
   macro avg       0.37      0.50      0.42      1973
weighted avg       0.54      0.73      0.62      1973

Confusion Matrix:
[[   0  527]
 [   0 1446]]
```

Figure 9 : Résultat du Train-Test Split

Méthode 2 : Réseau RBF from scratch

Le réseau RBF (Radial Basis Functions) fait partie des réseaux de neurones supervisés. Il est constitué de trois couches : une couche d'entrée qui retransmet les entrées sans distorsion, une couche

cachée qui contient les neurones RBF qui sont généralement des gaussiennes et une couche de sortie dont les neurones sont généralement animés par une fonction d'activation linéaire. Chaque couche est complètement connectée à la suivante et il n'y a pas de connexions à l'intérieur d'une même couche



RBF Neural Network Basé sur K-means Algorithm :

Nous utiliserons les bibliothèques suivantes :

```
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.preprocessing import scale
from sklearn import datasets
```

Figure 10 : Les bibliothèques utilisées

K-means Algorithm :

Le réseau de neurones à fonction de base radiale (RBF) est un réseau de neurones artificiels à rétroaction avec une forte capacité d'approximation. Un algorithme K-means basé sur le paramètre de densité a été introduit pour déterminer le centre de regroupement visant à améliorer le taux d'apprentissage du RBF :

On cherche à prédire la valeur de Energy_Appliance qui est soit 0 soit 1 :

0 pour Appliance <60 et 1 pour Appliance >60

Donc on a le nombre de classe = 2

```
nbreClasses = 2
def rbf(x, c, s):
    distance=np.linalg.norm(np.array(x)-np.array(c))
    return 1/np.exp(-distance/(2*s**2))

def kmeans(X, k, itersmax=100):

    clusters=X[np.random.choice(range(len(X)), k, replace=False)]
    converged=False

    iter=0
    while(not converged)and(iter<itersmax):

        cluster_list=[[]for i in range(len(clusters))]
        for x in X:
            distances=[]
            for c in clusters:
                distances.append(np.linalg.norm(np.array(x)-np.array(c)))
            cluster_list[int(np.argmin(distances))].append(x)

        cluster_list=list((filter(None, cluster_list)))

        prevClusters=clusters.copy()
        clusters=[]

        for j in range(len(cluster_list)):
            clusters.append(np.mean(cluster_list[j], axis=0))

        diff=np.abs(np.sum(prevClusters)-np.sum(clusters))
        print('Test KMEANS :', diff)
        converged=(diff==0)

        iter+=1

    stds=[np.std(x) for x in cluster_list]
    return np.array(clusters), stds
```

RBF Algorithm :

```
class RBFNet(object):

    def __init__(self, k=2, lr=0.01, epochs=100, rbf=rbf, withstds=True):
        self.k=k
        self.lr=lr
        self.epochs=epochs
        self.rbf=rbf
        self.withstds=withstds

        self.w=np.random.randn(nbreClasses,self.k)
        self.b=np.random.randn(nbreClasses)

    def rbf_states(self, X, clusters, stds):
        rbfs=[]
        for x in X:
            rbfs.append([rbf(x, c, s)for(c, s)in zip(clusters, stds)])
        return np.array(rbfs)

    def fit(self, X, y):
        self.centers,self.stds=kmeans(X,self.k, itersmax=100)

        if not self.withstds:
            dMax=np.max([np.linalg.norm(np.array(c1)-np.array(c2)) for c1 in self.centers for c2 in self.centers])
            self.stds=np.repeat(dMax/np.sqrt(2*self.k),self.k)

        epochsDisp=self.epochs//10
```

```
        for epoch in range(self.epochs):
            globalLoss=0;
            for i in range(X.shape[0]):
                a=np.array([rbf(X[i], c, s) for c, s in zip(self.centers,self.stds)])
                F=self.w.dot(a.T)+self.b
                loss=(y[i]-F)**2
                globalLoss=globalLoss+np.linalg.norm(loss)
                error=-(y[i]-F).flatten()
                self.w=self.w-self.lr*error.reshape(nbreClasses,1).dot(a.reshape(1,self.k))
                self.b=self.b-self.lr*error

            if epoch%epochsDisp== 0 :
                print(str(epoch),'RBF training : mean error :{0:.2f}'.format(globalLoss/X.shape[0]))

    def predict(self, X):
        y_pred=[]
        for i in range(X.shape[0]):
            a=np.array([rbf(X[i], c, s)for c, s in zip(self.centers,self.stds)])
            F=self.w.dot(a.T)+self.b
            y_pred.append(F)
        return np.array(y_pred)
```

Figure 11 : Algorithme RBF

Appel à la fonction d'Apprentissage avec le réseau RBF

```
rbfnet=RBFNet(lr=0.01, k=30, epochs=200, withstds='true')

n12 = np.squeeze(np.asarray(xTrain))
n13 = np.squeeze(np.asarray(yTrain))


rbfnet.fit(n12, n13)
```

Figure 12 : La fonction d'apprentissage

Résultat d'apprentissage :

```
0 RBF training : mean error :0.40
20 RBF training : mean error :0.39
40 RBF training : mean error :0.39
60 RBF training : mean error :0.39
80 RBF training : mean error :0.39
100 RBF training : mean error :0.39
120 RBF training : mean error :0.39
140 RBF training : mean error :0.39
160 RBF training : mean error :0.39
180 RBF training : mean error :0.39
```

Figure 13 : Résultat d'apprentissage

On essaye de prédire avec les exemples de test xTest :

```
n14 = np.squeeze(np.asarray(xTest))
predictions=rbfnet.predict(n14)
```

Résultat de Prédiction :

```
Evaluation :
Exemples Test : (5921, 12)
---------------------------------- :
--------------------   Prediction   ------------------
[0 0 0 ... 1 1 0]

--------------------   true   ------------------
[1 0 1 ... 0 0 1]
              precision    recall  f1-score   support

           0       0.27      0.38      0.32      2192
           1       0.53      0.41      0.47      3729

    accuracy                           0.40      5921
   macro avg       0.40      0.40      0.39      5921
weighted avg       0.44      0.40      0.41      5921
```

Figure 14 : Résultat de prédiction

Conclusion :

Dans ce projet, on a pu réaliser deux applications, une pour l'analyse des sentiments des phrases et l'autre pour l'approximation de la consommation énergétique d'une maison ce qui nous a permis de développer beaucoup de connaissances sur l'apprentissage automatique et profond, un des champs les plus importants de l'intelligence artificielle les réseaux de neurones.

Références :

https://archive.ics.uci.edu/ml/datasets/Appliances+energy+prediction

https://archive.ics.uci.edu/ml/datasets/Sentiment+Labelled+Sentences

https://www.lovelyanalytics.com/2020/05/26/accuracy-recall-precision/

http://eric.univ-lyon2.fr/~ricco/cours/slides/reseaux_neurones_perceptron.pdf