Université Abdelmalek Essaadi
Ecole Nationale des Sciences Appliquées
Al Hoceima

Devoir Libre
« Apprentissage Statistique »

Réaliser par : KHALIL EL MAGHRAOUI
Génie informatique II

Table des matières

1- Question de cours

1-1. La définition du machine Learning

Deux définitions du Machine Learning sont proposées. Arthur Samuel l'a décrit comme :

"le domaine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmés.".

Il s'agit d'une définition plus ancienne et informelle.

Tom Mitchell donne une définition plus moderne :

« On dit qu'un programme informatique apprend de l'expérience E en ce qui concerne une certaine classe de tâches T et une mesure de performance P, si ses performances aux tâches en T, telles que mesurées par P, s'améliorent avec l'expérience E. ".

Exemple : Jouer aux dames :

- E : l'expérience de jouer à de nombreux jeux de dames
- T = la tâche de jouer aux dames.
- P = la probabilité que le programme remporte le prochain match.

En général, Le machine Learning constitue une manière de modéliser des phénomènes, dans le But de prendre des décisions stratégiques. Tout problème d'apprentissage Statistique peut être attribué à l'une des deux grandes classifications :

Apprentissage supervisé et apprentissage non supervisé.

1-2. Au niveau de la modélisation du Machine Learning, quel est la différence entre la phase de déploiement et la phase d'apprentissage ?

- **La phase de d'apprentissage** : C'est la première phase dans cette étape, nous formons notre modèle pour améliorer ses performances pour un meilleur résultat du problème.
  Nous utilisons des ensembles de données pour entraîner le modèle à l'aide de divers algorithmes d'apprentissage automatique. La formation d'un modèle est nécessaire pour qu'il puisse comprendre les différents modèles, règles et fonctionnalités.

- **La phase de déploiement** : C'est La dernière étape du cycle de vie de l'apprentissage statistiques est le déploiement, où nous déployons le modèle dans le système réel.
  Si le modèle préparé ci-dessus produit un résultat précis selon nos exigences avec une vitesse acceptable, nous déployons le modèle dans le système réel. Mais avant de déployer le projet,

nous vérifierons s'il améliore ses performances en utilisant les données disponibles ou non. La phase de déploiement est similaire à la rédaction du rapport final d'un projet.

1-3. Pourquoi doit-on effectuer une analyse d'erreur dans la modélisation du Machine Learning ?

L'analyse des erreurs dans le Machine Learning ne vise pas seulement à améliorer les performances de votre métrique cible, mais également à s'assurer qu'un modèle fonctionnant bien sur un ensemble de données d'entraînement et de validation statique est tout aussi performant en production. Cela implique de comprendre les limites de votre processus de formation - données, fonctionnalités ou modèle, et d'essayer de rendre ces aspects aussi robustes que possible. Malheureusement, il n'y a pas de cadre détaillé qui puisse être appliqué à n'importe quel problème de ML.

Les problèmes de vision par ordinateur peuvent nécessiter une analyse d'erreur différente de celle des problèmes de PNL. Mais c'est ce qui rend ce processus créatif !

1-4. Si les variables sont indépendantes alors : $f_Y(y|x) \geq f_Y(y)$ et $f_X(x|y) \geq f_X(x)$?

Non, si X et Y sont des variables indépendantes :

- $f_y(y|x) = f_y(y)$ et $f_x(x|y) = f_x(x)$

1-5. C'est quoi l'overfitting ?

Parmi les difficultés majeures des approches fréquentistes est le problème du surapprentissage

« Overfitting » qui est généralement contourné en utilisant une régularisation, un outil à la fois riche, flexible mais complexe à utiliser .

Autrement dit L'overfitting est caractérisé par une erreur de type variance très élevée, il est observé lorsqu'on utilise des modèles très complexes sur des problèmes simples mais bruités.

. C'est pourquoi il est conseillé de mesurer la précision à la fois sur les données d'apprentissage et sur les données de validation

1-6. Quel est le rôle du réseau Bayésien ?

Le réseau bayésien est une technologie informatique clé pour faire face aux événements probabilistes et pour résoudre un problème incertain, L'intérêt de cette approche est donc fort quand on peut tenir compte d'expériences passées parfaitement similaires. Elle est donc utilisée dans plusieurs domaines comme par exemple la détection de spams : la connaissance préalable des spams permet d'associer une probabilité correspondant au nombre de fois où un type de mot apparaît. Cette probabilité, obtenue grâce aux expériences passées, permet de considérer un mot comme étant typique d'un spam.

1-7. Qu'est-ce que la matrice de transition dans les chaines de Markov ?

Matrice de transition décrivant les probabilités de transitions particulières et un état initial (ou distribution initiale) dans l'espace d'états. Par convention, nous supposons que tous les états et transitions possibles ont été inclus dans la définition du processus, il y a donc toujours un état suivant et le processus ne se termine pas.

Donne par la formule ci-dessous :

Theorem 8.6: Let $\{X_0, X_1, X_2, \ldots\}$ be a Markov chain with $N \times N$ transition matrix P. Then the t-step transition probabilities are given by the matrix P^t. That is,

$$\mathbb{P}(X_t = j \mid X_0 = i) = \left(P^t\right)_{ij}.$$

It also follows that

$$\mathbb{P}(X_{n+t} = j \mid X_n = i) = \left(P^t\right)_{ij} \text{ for any } n. \qquad \square$$

1-8. Quelle est l'utilité d'utiliser les « HIDDEN Markovian Mode » ?

Un modèle de Markov caché (HMM) est un modèle statistique qui peut être utilisé pour décrire l'évolution d'événements observables qui dépendent de facteurs internes, qui ne sont pas directement observables. Nous appelons l'événement observé un «symbole» et le facteur invisible sous-jacent à l'observation un «état». Un HMM se compose de deux processus stochastiques, à savoir, un processus invisible d'états cachés et un processus visible de symboles observables. Les états cachés forment une chaîne de Markov et la distribution de probabilité du symbole observé dépend de l'état sous-jacent. Pour cette raison, un HMM est également appelé un processus stochastique doublement intégré

Autrement dit C'est un modèle statistique dans lequel le système modélisé est supposé être un processus markovien de paramètres inconnus, on ne peut pas observer directement la séquence d'états: les états sont cachés. Chaque état émet des "observations" qui, elles, sont observables. On ne travaille pas donc sur la séquence d'états, mais sur la séquence d'observations générées par les états. HMM sont massivement utilisés notamment en reconnaissance de formes, en intelligence artificielle ou encore en traitement automatique du langage naturel.

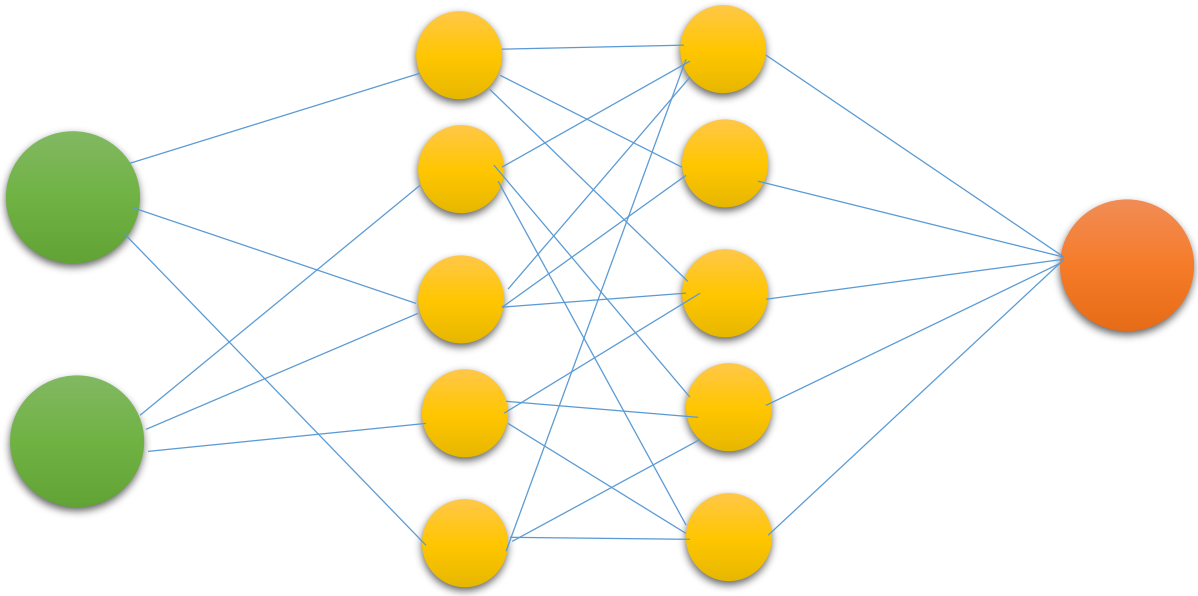
1-9. Donnez une explication brève du Reinforcement Learning Algorithme

Reinforcement Learning Algorithme est un paradigme d'apprentissage qui consiste à apprendre à contrôler un système afin de maximiser une mesure de performance numérique qui exprime un objectif à long terme. Ce qui distingue l'apprentissage par renforcement de l'apprentissage supervisé, c'est que seule une rétroaction partielle est donnée à l'apprenant sur les prédictions de l'apprenant.

En outre, les prévisions peuvent avoir des effets à long terme en influençant l'état futur du système contrôlé. Ainsi, le temps joue un rôle particulier. L'objectif de l'apprentissage par renforcement est de développer des algorithmes d'apprentissage efficaces, ainsi que de comprendre les avantages et les limites des algorithmes.

L'apprentissage par renforcement est d'un grand intérêt en raison du grand nombre d'applications pratiques qu'il peut être utilisé pour traiter, allant des problèmes d'intelligence artificielle à la recherche opérationnelle ou à l'ingénierie de contrôle. Dans ce livre, nous nous concentrons sur ces algorithmes d'apprentissage par renforcement qui s'appuient sur la puissante théorie de la programmation dynamique. Nous donnons un catalogue assez complet de problèmes d'apprentissage, décrivons les idées de base, notons un grand nombre d'algorithmes de pointe, suivis d'une discussion sur leurs propriétés théoriques et leurs limites.

1-10. Donnez un schéma explicatif qui illustre le fonctionnement des réseaux de neurones



Entrees

Neurones

Hidden layers

Neurone de sorite

2-Algorithme de régression « sous python »

Problématique :

Pour définir la problématique et ses données d'entraînement, on doit poser cette question : Étant donné les caractéristiques de la ferme, combien devrais-je normalement la vendre ?

Imaginons pour l'instant que la seule caractéristique dont nous disposons est la surface de la ferme.

Notre data set « ferme.csv » est un ensemble de N = 545 observations de surface et des prix associés : (x,y)=(surface, prix) .

2-1. Illustrer cette association (surface, prix) sous python (jupyter)

On a importé :

- numpy: **import numpy as np.**
- pandas: **import pandas as pd.**
- matplotlib: **import matplotlib.pyplot as plt.**

Et d'autres bibliothèques pour le reste du code :

machine learning DL

```
Realise par : Khalil el maghraoui
```

import libraries

```
In [1]: import numpy as np

In [15]: import seaborn as sns

In [70]: from sklearn import linear_model

In [71]: import pandas as pd

In [72]: import matplotlib.pyplot as plt

In [73]: from scipy.stats import skew, norm
         from sklearn.neighbors import KNeighborsRegressor
```

2-2. Ensuite lire les données depuis le fichier ferme.csv

D'abord on utilise la command pwd pour savoir le PATH de mon fichier.ipynb et placer trainning data dans le même répertoire .

Donc en peut charger notre data(ferme.csv) correctement dans notre fichier.ipynb ,lire le data A l'aide de **read_csv() ,** Et après les afficher dans un tableau

```
In [74]: pwd
Out[74]: 'C:\\Users\\KURAPIKA.dll\\Desktop\\GI2\\S1\\Apprentissage statistique'

In [75]: train = pd.read_csv("ferme.csv")

In [76]: train
```

Out[76]:

	prix	surface
0	1330	37
1	1400	32
2	904	26
3	955	3
4	2545	7
...	...	...
540	1490	48
541	2020	58
542	2050	7
543	1220	42
544	1610	44

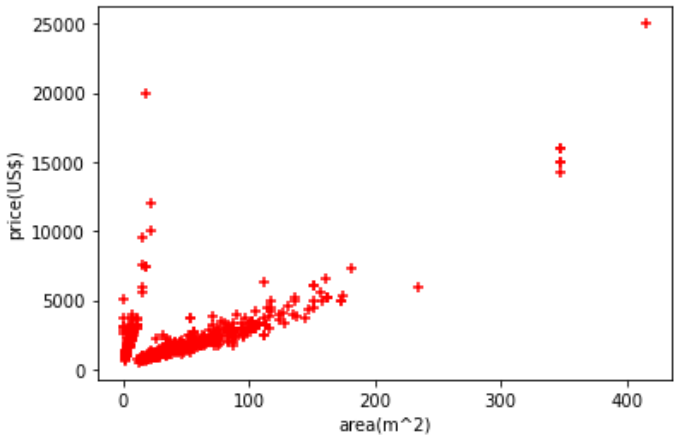
545 rows × 2 columns

2-3. Interprétez et expliquez le graphe.

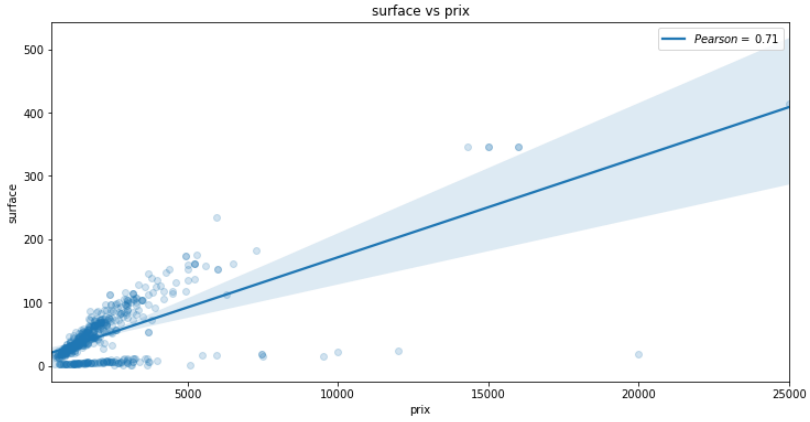
Donc on va afficher notre data avant l'entrainement, d'après on peut se dire que La relation entre la variable d'entrée et la variable de sortie est linéaire. La régression linéaire fonctionne mieux lorsque la relation est linéaire. Donc le prix dépend de manière linéaire de la surface du ferme. On peut donc émettre une hypothèse de modélisation qui est que le phénomène possède la forme d'une droite. Aussi, on peut voir que lorsque la surface devient un peu trop grande, les données semblent devenir moins modélisables

In [77]:
```
# afficher les données dans un graphe afin d'examiner s'il existe une relation
#entre la surface et le prix d'une ferme
%matplotlib inline
plt.xlabel('area(m^2)')
plt.ylabel('price(US$)')
plt.scatter(train.surface, train.prix, color='red', marker='+')
```

Out[77]:
```
<matplotlib.collections.PathCollection at 0x184b2534ee0>
```



In [78]:
```
pearson_g = 0.71
plt.figure(figsize=(12,6))
sns.regplot(data=train, x = 'prix', y = 'surface', scatter_kws={'alpha':0.2})
plt.title('surface vs prix', fontsize = 12)
plt.legend(['$Pearson=$ {:.2f}'.format(pearson_g)], loc = 'best')
plt.show()
```



2-4. Trouvez les valeurs de theta sous python

La régression linéaire estime les coefficients de la droite. Cela signifie que nous devons calculer thêta 0 et thêta 1 pour trouver la meilleure ligne pour ajuster les données. Cette ligne est la meilleure estimation de l'émission des points de données inconnus. Voyons comment nous pouvons trouver cette ligne

Finalement On trouve donc la valeur numérique de θ pour nos données :

prix

In [81]:
```
# Trouvez les valeurs de theta sous python
X = np.matrix([np.ones(train.shape[0]), train['surface'].values]).T
y = np.matrix(train['prix']).T

# On effectue le calcul exact du paramètre theta
theta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)

print(theta)
```

```
[[828.36674356]
 [ 29.83343047]]
```

2-5. Représentation de la la droite de régression graphiquement avec l'association (surface, prix)

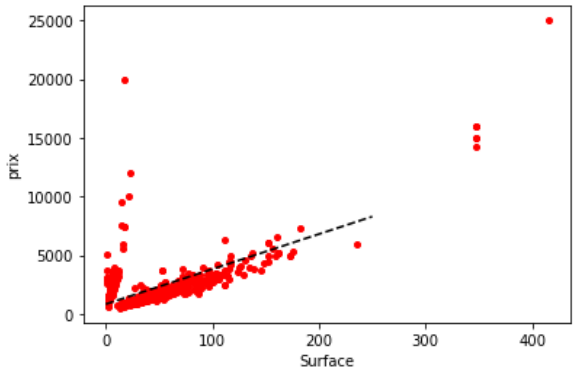
Le modèle final de la droite de régression sera donc : prix= ($\theta^*_1 \times$ surface) + θ^*_0

In [82]:
```
plt.xlabel('Surface')
plt.ylabel('prix')

plt.plot(train['surface'],train['prix'], 'ro', markersize=4)

# On affiche la droite entre 0 et 250
plt.plot([0,250], [theta.item(0),theta.item(0) + 250 * theta.item(1)], linestyle='--', c='#000000')

plt.show()
```



2-6. Représentez graphiquement encore une fois cette régression linéaire mais cette fois ci en utilisant le package scikit-learn

On vient ici de décomposer l'entraînement de la régression linéaire. Ce dernier est déjà implémentée dans le package **scikit-learn**

```
In [96]: from sklearn import linear_model
         reg = linear_model.LinearRegression()

In [97]: reg.fit(X, y)
         reg.predict([[11,22]])

Out[97]: array([[1484.70221392]])
```

Predict from CSV File

```
In [98]: test_data = pd.read_csv("ferme _csv_test.csv")
         test_data.head()
```

Out[98]:

	prix	surface
0	1330	37
1	1400	32
2	904	26
3	955	3
4	2545	7

```
In [99]: test_data_results = reg.predict(test_data)
         test_data['price']=test_data_results

In [100]: test_data_results

Out[100]: array([[ 1932.20367099],
                 [ 1783.03651864],
                 [ 1604.03593581],
                 [  917.86703497],
                 [ 1037.20075686],
                 [ 1544.36907487],
                 [ 2051.53739288],
                 [ 2827.20658512],
                 [ 2707.87286324],
                 [ 1037.20075686],
                 [ 3244.87461172],
                 [11180.56711703],
                 [ 1604.03593581],
                 [ 1902.37024052],
                 [ 1395.20192251],
```