TP0 : introduction à scikit-learn et à l'apprentissage automatique

Objectifs

Pour ce TP nous allons apprendre à écrire notre premier modèle : une régression linéaire. Nous allons l'expérimenter sur un problème jouet afin de suivre pas à pas la construction et l'évaluation de notre modèle.

Dans le cas d'une régression linéraire on suppose qu'il existe une relation linéaire entre les données d'entrées et les valeurs à prédire.

Ce tp est insipré des travaux de Gavin Hackeling dans le livre *Mastering machine learning with scikit-learn*.

Il s'agit de prédire la relation entre le prix d'une pizza et sa taille. Comme spécifié précédemment, et comme nous avons choisi une régression linéaire, celà signifie que nous faisons l'hypothèse qu'il existe une relation linaire entre le prix d'une pizza et sa taille.

Exercice 1 : Création et analyse des données

Nous allons tout d'abord générer un ensemble de points pour l'ensemble d'apprentissage:

Exemple	Taille	Prix
1	6	7
2	8	9
3	10	13
4	14	17,5
5	18	18

Nous allons pour celà utiliser la bibliothèque `numpy` :

```
import numpy as np
```

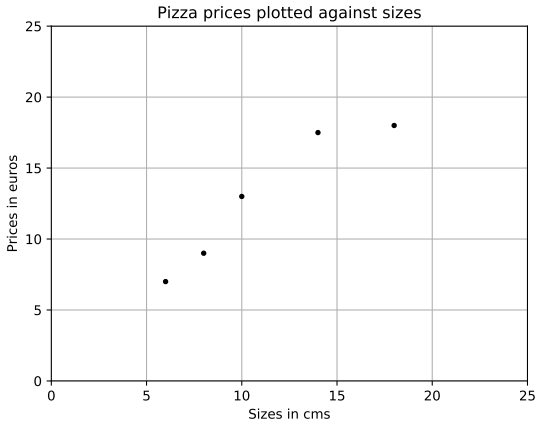
Il nous faut ensuite déclarer un tableau numpy contenant les données (tailles des pizzas) et déclarer un tableau contenant les prix correspondants.

```
X = np.array([[6], [8], [10], [14], [18]])
y = [7, 9, 13, 17.5, 18]
```

Nous souhaitons maintenant afficher les données dans un graphe afin d'examiner s'il existe une relation entre la taille et le prix d'une pizza. Pour cela nous allons utiliser `matplotlib` :

```
import matplotlib.pyplot as plt
```

Il faut maintenant définir le code matplotlib afin d'obtenir ce résultat :



Nous pouvons constater qu'il existe bien une relation, le prix d'une pizza augmente avec sa taille, ce qui est cohérent avec notre expertise sur le sujet.

Exercice 2: Regression linéaire

Nous allons utiliser la bibliothèque `sklearn` et en particulier le modèle de régression linéaire :

```
from sklearn.linear_model import LinearRegression
```

Utiliser maintenant les fonctions `fit` pour l'apprentissage et `predict` pour prédire la réponse du modèle sur un exemple. On obtient pour notre régression linéaire :

La régression linéaire permet d'apprendre un modèle de la forme :

$$y = \alpha x + \beta$$

Exercice 3: Evaluation du modèle

Il nous faut maintenant évaluer notre modèle. Pour celà nous avons besoin de définir une fonction de perte (également appelée une fonction de coût). La différence entre le prix réel des pizzas et le prix prédit par notre modèle est appelée *erreur résiduelle* ou *erreur d'apprentissage*. Lorsque nous évaluons notre modèle sur une base de test indépendante, l'erreur résultante est appelée *erreur de prédiction* ou *erreur de test*.

Nous pouvons créer le meilleur modèle possible en minimisant la somme de l'erreur résiduelle. Cette erreur est appellée *residual sum of squares (RSS)*.

La fonction de coût associée à cette erreur est définie comme suit :

$$RSS = \sum_{i=1}^{n} (y_i - f(x_i))^2,$$

avec y_i la valeur observée et $f(x_i)$ la valeur prédite.

Calculer l'erreur obtenue en python.

```
Residual sum of squares : 8.75
```

Exercice 4: Etude théorique de notre problème de pizza.

Nous souhaitons trouver les meilleurs paramètres α et β pour résoudre l'équation $y = \alpha x + \beta$ (minimiser la fonction de coût).

Pour calculer le paramètre α nous avons besoin de calculer la variance de x et la covariance de x et y. La variance mesure à quel point un échantillon de points est éparpillé. Une variance proche de 0 signifie que les données sont proches de leur moyenne. La variance est calculée comme suit :

$$var(x) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1},$$

avec $\bar{x}$ la moyenne des x, x_i l'exemple d'apprentissage, et n le nombre d'exemples.

Nous pouvons directement calculer la variance d'un échantillon à l'aide de la bibliothèque numpy avec la fonction `np.var(x, ddof=1)`.

La covariance mesure à quel point deux variables changent ensemble. Si les variables augmentent en même temps alors la covariance est positive, si l'une augmente et l'autre diminue alors la covariance est négative. S'il n'existe aucune relation entre les deux variables alors la covariance est nulle.

La covariance est calculée :

$$cov(x, y) = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{n-1},$$

avec x_i le ième exemple, $\bar{x}$ la moyenne des x, $\bar{y}$ la moyenne des prix, et y_i le prix du ième exemple.

Comme pour la variance, nous pouvons calculer la covariance directement à l'aide de numpy avec $np.cov(x.transpose(), y)[0][1]$.

Nous avons maintenant les valeurs de la variance et de la covariance nécessaires pour calculer le paramètre α comme suit :

$$\alpha = \frac{cov(x, y)}{var(x)}$$

Calculer ce paramètre.

Il nous faut maintenant calculer le paramètre $\beta = \bar{y} - \alpha\bar{x}$, avec $\bar{y}$ la moyenne des y, et $\bar{x}$ la moyenne des x (notre modèle doit passer par le centroid de coordonnées $(\bar{x}, \bar{y})$).

Calculer ce paramètre.

Comme vu en cours, afin d'éviter le surapprentissage, nous souhaitons maintenant évaluer notre modèle sur un ensemble de test indépendant.
Définissons cet ensemble comme suit :

Exemple	Taille	Prix
1	8	11
2	9	8.5
3	11	15
4	16	18
5	12	11

Nous allons utiliser une mesure classique appelée `R-squared` (coefficient de détermination). Nous avons besoin de notre RSS définie précédement ainsi que la somme totale des carrés :

$$SS_{tot} = \sum_{i=1}^{n} (y_i - \bar{y})^2,$$

avec y_i l'observation pour la ième variable et $\bar{y}$ la moyenne des observations.

Nous définissons simplement

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}.$$

Heureusement, là encore nous pouvons calculer directement R^2 avec sklearn, en utilisant la fonction score.

```
Rsquared = 0.66
```