Régression linéaire multiple

1er exemple : Prévision de ventes

Nous étudierons les variations des ventes liées aux campagnes publicitaires dans différents médias.
Jeu de données : « publi.csv »
On indexera le dataFrame data sur la colonne 0

- Afficher les 5 premières lignes
- Quelle est la dimension du dataFrame ?
- Quelles sont les caractères étudiés.
- Quelle est la variable de réponse ?

Visualisation avec seaborn

On utilisera la méthode pairplot pour afficher différents nuages de points :
Les arguments de pairplot sont :

- le dataframe
- x_vars : [……caractères….]
- y_vars= reponse
- kind = scatter ou reg
- size : hauteur d'une unité de la grille
- aspect = multiplier par size il permet de définir la largeur

Régression linéaire

Nous allons définir une équation $y = \beta_0 + \beta_1 \times TV + \beta_2 \times Radio + \beta_3 \times Journeaux$

Nous devons fournir à Scikit-learn une matrice X d'entrée (les caractères) et une matrice de sortie y (la reponse)

- Matrice d'entrée :
  caract_cols = ['TV','Radio','Journeaux']
  X = data[caract_cols]
- Matrice de sortie : y = data['ventes']

Définition des jeux de test et d'entrainement

Importer le module « train_test_split » de la bibliothèque « sklearn.cross_validation »

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 1)

Application du modèle de régression

- Importer le module « LinearRegression « de la bibliotheque « sklearn.linear_model »
  LinearRegression.fit(X_train, y_train)
- Affichage des coefficients et interpretation :
  Afficher les coefficients « linearRegression.intercept_ » et « linearRegression_coef_ »
- Associer les coefficents aux caractères étudiés :
  zip(caract_cols, linearRegression_coef_)

  L'équation obtenue est donc : $y = \cdots$

Prévisions :

Appliquer le modèle au jeu X_test :

```
y_pred = linearRegression(X_test)
```

Evaluation du modèle

Trois indicateurs permettent d'évaluer l'erreur de prédiction commise :

- MAE : Mean Absolute Error : erreur absolue moyenne $\frac{1}{n}\sum|y_i - \bar{y}|$
- MSE : Mean Squared error : erreur quadratique moyenne $\frac{1}{n}\sum(y_i - \bar{y})^2$
- RMSE : Root Mean Squared Error : erreur type $\sqrt{\frac{1}{n}\sum(y_i - \bar{y})^2}$

C'est la RMSE qui sera le plus souvent utilisée car elle est de la même unité que la variable étudiée et qu'elle pénalise les grands écarts.

```
print(np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

La valeur obtenue doit être la plus faible possible et le choix des variables pertinentes sera fait dans cet objectif.

Sélection des caractères

Recalculer la RMSE en perturbant le modèle par un choix judicieux des variables :

caract_cols= ['TV', 'Radio']
caract_cols= ['TV', 'journeaux']
caract_cols= ['Radio', 'journeaux']

Quel choix vous semble être le plus judicieux ?

Prévision de location de vélos

1. Import des données.
   Nous allons étudier la location de vélos dans une ville qui nous fournit deux ans d'enregistrement de locations heure par heure.
   Concours Kaggle : https://www.kaggle.com/c/bike-sharing-demand/data

   Le fichier de données : « bikeshare.csv »
   Il peut être utile de se poser les questions suivantes :
   - Que représente chaque observation ?
   - Quelle est la variable de réponse (définie par Kaggle) ?
   - Combien de propriétés a-t-on ?

2. Visualisation.
   Importer matplotlib
   Représenter le nuage de points correspondant aux données (fixer la transparence alpha à 0.2)

3. Régression linéaire simple :
   - Effectuer une régression linéaire permettant de prévoir le nombre de locations en fonction de la température et afficher les coefficients.
   - Faire une prédiction à 25 degrés.

4. Régression linéaire multiple.
   Considérons maintenant que les locations dépendent également de la saison, du climat et de l'humidité ambiante.
   - Représenter les nuages de points correspondant à chaque association.
   - Créer la matrice de corrélation dataframe.corr()
     Noter les fortes corrélations.
   - Utiliser seaborn pour une meilleure expérience de visualisation :
     seaborn.heatmap(dataFrame.corr())

5. Remarquez-vous d'autres relations ?
6. Utilisation du train test rmse
   Appliquer un train/test split aux données (fixer le random_state pour réutiliser les jeux de données)
   Appliquer le modèle de régression linéaire sur les données train
   Faire une prédiction sur les données X_test et calculer l'erreur type (RMSE) : racine carrée du carré moyen des erreurs ou erreur quadratique moyenne)
   Essayer plusieurs corrélations en tentant de minimiser cette erreur.