TP Régression linéaire

Objectif du TP

L'objectif de ce TP est d'établir un modèle de régression linéaire multiple reliant, de manière optimisée, une variable quantitative Y à plusieurs variables quantitatives ou qualitatives X_1,…, X_p. Ce modèle permet par la suite la prédiction de Y en fonction de données X_k. Dans notre cas, le modèle linéaire cherche à prédire le montant des bénéfices réalisés par une startup ("profit") à partir des quatre variables indépendantes suivantes : dépenses de recherche et de développement ("R&D Spend"), d'administration ("Administration"), de marketing ("Marketing Spend") et l'état ("State") qui indique l'endroit où l'entreprise a commencé ses opérations initiales. Pour ce faire, il est nécessaire d'appliquer un prétraitement des données qui est une étape indispensable à tout projet de Machine Learning.

Prétraitement des données

Le prétraitement des données est une technique d'exploitation de données brutes qui consiste à transformer ces dernières en un format compréhensible. Sachant que les données du monde réel sont dans la plupart des cas incomplètes, incohérentes ou dépourvues de certaines tendances, le prétraitement de celles-ci est une méthode certaine pour résoudre ces problèmes ainsi que de nombreuses erreurs susceptibles d'exister dans ces données.

Le prétraitement des données pour l'apprentissage automatique consiste à suivre les étapes suivantes :

Etape 1 : Importation des bibliothèques

- Python n'a pas nativement des fonctions pour importer des données au format csv. Afin d'importer les données en Python la bibliothèque **pandas** est utilisée. Pandas est une bibliothèque open source, sous licence BSD, de Python qui permet la manipulation et l'analyse des données à travers des DataFrames qui correspondent à des tableaux de données d'individus qui représentent des observations (lignes) et de variables qui sont des attributs décrivant les individus (colonnes).
- **NumPy** est l'une des bibliothèques Python utilisée pour la prise en charge des tableaux multidimensionnels pour les opérations mathématiques et logiques. Les fonctions

fournies par NumPy peuvent être utilisées pour indexer, trier, remodeler et transmettre des images et des ondes sonores sous la forme d'une matrice de nombres réels en plusieurs dimensions.

- **Matplotlib** est le package de visualisation de données le plus basique en Python. Il prend en charge une grande variété de graphiques tels que des histogrammes, des graphiques à barres, des spectres de puissance, des graphiques d'erreur, etc. Il s'agit d'une bibliothèque graphique bidimensionnelle qui produit des graphiques clairs et concis qui sont essentiels pour l'analyse exploratoire des données. **matplotlib.pyplot** est une bibliothèque de traçage utilisée pour les graphiques 2D.
- Le module **warnings** gère les avertissements en python. Ces derniers sont des messages concernant des erreurs ou des anomalies qui ne sont pas suffisamment graves pour mériter d'interrompre le déroulement du programme.

Étape 2 : Importation de la dataset en question

En utilisant la bibliothèque Pandas, on importe la dataset sous format CSV. L'utilisation des fichiers de type CSV est dû à leur faible poids. La structure générale des fichiers CSV utilise des lignes comme observations et des colonnes comme attributs.

Étape 3 : Vérification des valeurs manquantes

Les valeurs manquantes en statistique sont relatives au cas lorsqu'une variable donnée n'a pas d'observations pour un certain individu. Lors de l'analyse statistique, les valeurs manquantes ne peuvent pas être ignorées. Cependant, si ces données ne sont pas traitées correctement, le chercheur peut tirer des conclusions inexactes sur les données.

Pour obtenir le nombre de valeurs manquantes dans chaque colonne, nous pouvons utiliser les ***méthodes*** pandas ***isnull*** ou ***notnull***. **La** méthode **isnull** renvoie une valeur booléenne **TRUE** lorsque la valeur nulle est présente tandis que la méthode **notnull** renvoie une valeur booléenne **TRUE**, s'il n'y a pas de valeur nulle pour chaque observation dans les données. On ajoute **.sum** pour déterminer le nombre des valeurs manquantes.

Étape 4 : Vérification des valeurs aberrantes

Les valeurs aberrantes peuvent être détectées en utilisant la visualisation, en mettant en œuvre des formules mathématiques sur l'ensemble de données ou en utilisant l'approche statistique. Le box-plot capture le résumé des données de manière efficace et efficiente avec seulement une

simple boîte et des moustaches. Il résume des exemples de données en utilisant les 25e, 50e et 75e centiles. On peut simplement obtenir des informations (quartiles, médiane et valeurs aberrantes) dans l'ensemble de données en regardant simplement sa boîte à moustaches.

Étape 5 : Vérification des valeurs catégorielles

Les modèles d'apprentissage automatique se basent sur des équations mathématiques, alors intuitivement la présence de données catégorielles entrainera un problème car on ne peut garder que des nombres dans les équations. Alors, ces données catégorielles doivent être codées en données numériques.

Etape 6 : Fractionnement de l'ensemble de données ;

Pour tous les modèles d'apprentissage automatique, qu'ils soient supervisés ou non supervisés, nous allons fractionner l'ensemble de données en deux :

- Un ensemble de données d'entrainement : Training.
- Un ensemble de données de test.

En général, on divise l'ensemble de données en un rapport 75/25% ou 80/20%, qui signifie que 75% des données sont des données de training et 25% pour les tests. Toutefois, ces répartitions peuvent varier en fonction de la forme et de la taille de la dataset.

Manipulation

Réaliser la régression linéaire aux moindres carrés des données fournies dans le fichier « 50_Startups»

1. Lancer l'environnement de développement **JUPYTER** intégré dans **Anaconda**. Créer un nouveau projet FILE/ NEW/NOTEBOOK et enregistrer le sous le nom « atelier regression.ipynb ».
2. Importer les librairies pandas, numPy matplotlib.pyplot, Seaborn pour la visualisation des données et warnings en utilisant la commande **import** <nom de la bibliothèque> **as** <nom symbolique>. Ajouter warnings.filterwarnings('ignore') après l'importation de la bibliothèque warnings pour supprimer tous les avertissements.
3. En utilisant .**read_csv()** de la librairie « pandas », importer le fichier « 50_Startups» dans un objet que vous nommerez «dataset».
4. Pour vérifier l'importation, afficher :
    a) Les premières lignes du DataFrame (**.head**)
    b) Le nombre d'observations et de variables (**.shape**)

c) Le type des variables (**.info**)

d) Les statistiques descriptives (**.describe**)

5. Vérifier s'il y a des valeurs aberrantes (outliers) en utilisant les box-plots des quatre variables (**.boxplot**).
6. Afficher les observations pour les valeurs suivantes
   a) Profit= 14681,4
   b) R&D Spend=0
7. Déterminer le Quantile d'ordre 0.05 pour la variable « profit »
8. Tracer les nuages de points entre les variables « profit » et « Administration » et entre « profit » et « R&D Spend » (**.scatter**)
9. Afficher l'histogramme de la variable « profit » pour avoir une idée de la façon dont elles sont réparties (**.hist**)
10. Afin de voir les relations entre les variables d'un dataframe, on est souvent amené à visualiser des nuages de points pour chaque combinaison possible de variables quantitatives. Analyser ces données en utilisant la fonction **.pairplot** et commenter les résultats obtenus.
11. Avant de créer notre modèle, on doit vérifier quelles sont les variables qui ont une forte relation linéaire avec la variable « Profit ». Pour ce faire, il fallait déterminer la matrice de corrélation.

    Les coefficients de corrélation se situent dans l'intervalle [-1,1].
    – si le coefficient est proche de 1 c'est qu'il y a une forte corrélation positive
    – si le coefficient est proche de -1 c'est qu'il y a une forte corrélation négative
    – si le coefficient est proche de 0 en valeur absolue c'est qu'il y a une faible corrélation.

    Déterminer les coefficients de corrélation de Pearson (**.corr**) de chaque variable et afficher la matrice de corrélation sous forme de carte thermique (**.heatmap**). Interpréter les résultats obtenus.

12. Encoder la variable catégorielle « State » en faisant appel la classe *LabelEncoder***()** de la bibliothèque sklearn.preprocessing. Après avoir créé une instance de cette classe, on l'ajuste à l'aide de *fit_transform()* à la colonne « state» pour qu'elle revoie celle-ci sous format encodée. Afficher les cinq premières lignes de la matrice des variables explicatives après cet encodage. Que remarquez-vous ?

13. Maintenant, nous utilisons le OneHotEncoder pour transformer notre colonne «State» en données numériques. Pour cela, nous importons le **ColumnTransformer** et le **OneHotEncoder** respectivement des bibliothèques sklearn***.compose*** et ***sklearn.preprocessing***. Le ColumnTransformer permet à une colonne particulière du DataFrame d'être transformée séparément. Comparer ce type du codage par rapport au codage de la question précédente.

14. Une fois que notre ensemble de données est prêt, la prochaine tâche importante consiste à diviser notre ensemble de données en un ensemble d'apprentissage et un ensemble de test. Nous utilisons la fonction **train_test_split** pour fractionner nos données. Ici, nous donnons le "test_size = 0.2", ce qui indique que 20% des données sont l'ensemble de test.

15. Déterminez la régression linéaire grâce à **LinearRegression** de **sklearn.linear_model**. N'oubliez pas de faire fitter votre modèle à vos données d'entrée grâce à **.fit**. Donnez les coefficients de régression grâce à **.coef_**. Donnez l'ordonnée à l'origine grâce à .**intercept_.** Donner alors l'expression du modèle de la régression linéaire.

16. Utiliser le modèle obtenu pour prédire (**.predict**) les valeurs de nos données de test. Comparer alors les valeurs prédites aux valeurs réelles.

17. Déterminer les valeurs de MSE (carré moyen des erreurs), MAE (erreur absolue moyenne), RMSE (erreur quadratique moyenne) et le coefficient du détermination R^2. Interpréter.

18. La régularisation consiste à modifier la fonction de coût d'une régression. Le but est de changer le comportement de l'estimateur des moindres carrés afin de pénaliser les coefficients trop importants ou de réduire le nombre de variables explicatives au strict minimum. Le module ***sklearn.linear_model*** contient des modèles prenant en compte les principales formules de la régularisation tels que Ridge et Lasso. Appliquer ces deux méthodes et déterminer les valeurs de la question 17. Comparer les résultats obtenus.

19. Utiliser maintenant la bibliothèque **statsmodels** pour définir un modèle des moindres carrés ordinaires (OLS) avec la technique de sélection Backward. Pour cela, construire la matrice X en concaténant une colonne de 1 à la matrice des variables explicatives. Afficher les résultats de la régression OLS en faisant appel à la méthode .**summary().** Retirer le prédicteur qui correspond à p-value la plus élevée. Ajuster de nouveau le

modèle et répéter le processus jusqu'à ce que toutes les variables conservées auront des p-values inférieurs à 0.05. Interpréter le résultat obtenu.