INFO 905 - Machine Learning Prix des maisons

Votre rendu se fera à l'aide d'un notebook Jupyter.
L'objectif de ce TP est de prédire le prix des maisons sur un jeu de données californien.

Données et environnement

1. Installez Python et vérifiez que vous disposez d'une version supérieure à la version 3 en exécutant la commande suivante :

```
pip3 --version
```

   Cette étape est nécessaire pour s'assurer que vous avez la bonne version de Python pour exécuter les codes du TP.

2. Installez les modules nécessaires pour ce TP en exécutant la commande suivante :

```
pip3 install --upgrade jupyter matplotlib numpy pandas scipy scikit-learn six
```

   Ces modules sont utilisés pour différentes tâches de manipulation de données et d'apprentissage automatique (Machine Learning).

3. Récupérez l'archive *housing.csv* et décompressez-la. Ces données serviront de jeu de données pour le TP.

4. Lancez un notebook Jupyter en exécutant la commande suivante :

```
jupyter notebook monfichier.ipynb
```

   Cela lancera une interface Jupyter Notebook où vous pourrez écrire et exécuter votre code.

Exploration des données

Utilisez la librairie *Pandas* pour explorer les données du jeu de données. Voici quelques fonctions utiles :

- *read_ csv* : permet de charger les données à partir d'un fichier CSV.
- *head* : affiche les premières lignes du jeu de données.
- *info* : affiche des informations sur les variables, telles que leur type et le nombre de valeurs non nulles.
- *value_ counts* : compte le nombre d'occurrences de chaque valeur dans une variable.
- *describe* : fournit des statistiques descriptives sur les variables, telles que la moyenne, l'écart-type, etc.
- *count* : compte le nombre de valeurs non nulles dans chaque variable.
-
- *mean* : calcule la moyenne des valeurs dans une variable.
- *min* et *max* : retournent respectivement la valeur minimale et maximale d'une variable.

Utilisez ces fonctions pour répondre aux questions suivantes :

1. Combien y a-t-il de variables dans le jeu de données ?

2. Quel est leur type (numérique, catégoriel, etc.) ?
3. Y a-t-il des données manquantes dans le jeu de données ?

Ces informations vous permettront de mieux comprendre les données avec lesquelles vous travaillez.

Jeu de test/validation

Pour évaluer la performance de nos modèles de prédiction, nous allons diviser les données en deux jeux : un jeu de test et un jeu de validation. Cela nous permettra de vérifier si nos modèles peuvent généraliser leurs prédictions sur de nouvelles données.

Pour diviser les données, nous utiliserons la fonction *train_test_split* de la librairie scikit-learn. Cette fonction permet de séparer les données en deux en spécifiant une proportion pour le jeu de test (par exemple, 80% pour l'entraînement et 20% pour la validation). Elle renvoie quatre ensembles de données : les variables explicatives d'entraînement, les variables explicatives de validation, la variable à expliquer d'entraînement et la variable à expliquer de validation.

Visualisation des données et recherche de corrélations

La visualisation des données est une étape importante pour comprendre les relations entre les variables et identifier les éventuelles corrélations. La librairie Pandas offre plusieurs fonctions de visualisation qui peuvent être utilisées pour explorer les données.

Utilisez la fonction *plot* de Pandas pour proposer une visualisation des données. Cela peut être un histogramme, un diagramme en boîte, un graphique à barres, etc.

Ensuite, utilisez la fonction *corr* pour calculer les corrélations entre les variables. Identifiez la variable qui semble être la plus corrélée au prix des maisons. Une corrélation élevée indique une relation linéaire forte entre deux variables, ce qui peut être utile pour la prédiction.

Expliquez pourquoi cette variable est considérée comme la plus corrélée et comment cela peut influencer la prédiction du prix des maisons.

Préparation des données

Avant d'entraîner nos modèles, il est nécessaire de préparer les données. Cette étape comprend plusieurs sous-tâches, telles que la séparation des variables explicatives de la variable à expliquer, la gestion des données manquantes et la transformation des variables qualitatives.

1. Séparez les variables explicatives de la variable à expliquer en utilisant les fonctions *drop* et *copy*. Cela permettra de créer deux ensembles distincts : un ensemble contenant les variables explicatives et un autre contenant la variable à expliquer.
2. Remédiez aux données manquantes en choisissant l'une des méthodes suivantes :
   - Supprimer les lignes correspondantes à des données manquantes en utilisant la fonction *dropna*.
   - Supprimer la variable entière si la majorité de ses données sont manquantes en utilisant la fonction *drop*.
   - Remplacer les valeurs manquantes par une valeur par défaut en utilisant la fonction *fillna*.

   Choisissez la méthode qui convient le mieux à vos données et justifiez votre choix.
3. Gérez les variables qualitatives en utilisant les classes *LabelEncoder* et *OneHotEncoder*. Ces classes permettent de transformer les variables qualitatives en variables numériques, ce qui est nécessaire pour certains algorithmes de Machine Learning. Utilisez la méthode *fit_transform* pour appliquer cette transformation sur vos données.

4. Recalibrez les variables pour qu'elles soient à la même échelle. Cela est généralement nécessaire car certains algorithmes de Machine Learning fonctionnent mieux lorsque les variables sont dans la même plage de valeurs. Deux méthodes couramment utilisées sont le *min-max scaling* et la *normalization*. Le *min-max scaling* met toutes les valeurs des variables dans un intervalle prédéfini, généralement entre 0 et 1. La *normalization* ajuste les valeurs des variables pour qu'elles aient une moyenne de 0 et un écart-type de 1. Expliquez ces deux méthodes et choisissez celle qui convient le mieux à vos données.

Sélection et entraînement de modèle

Maintenant que les données sont préparées, nous pouvons sélectionner et entraîner nos modèles de prédiction. Dans ce TP, nous utiliserons trois modèles différents : la régression linéaire, l'arbre de décision et les forêts aléatoires.

Régression linéaire

La régression linéaire est un modèle simple qui cherche à établir une relation linéaire entre les variables explicatives et la variable à expliquer. Utilisez la classe *LinearRegression* de la librairie scikit-learn pour créer un modèle de régression linéaire. Entraînez le modèle en utilisant la fonction *fit*, puis faites des prédictions sur le jeu de validation en utilisant la fonction *predict*. Enfin, évaluez la performance du modèle en calculant l'erreur quadratique moyenne (MSE) à l'aide de la fonction *mean_ squared_ error*.

Analysez les résultats obtenus et discutez de la pertinence de cette prédiction.

Arbre de décision

Les arbres de décision sont des modèles basés sur des règles de décision. Ils utilisent une série de questions sur les variables explicatives pour prendre des décisions et prédire la variable à expliquer. Utilisez la classe *DecisionTreeRegressor* de la librairie scikit-learn pour créer un modèle d'arbre de décision. Entraînez le modèle, faites des prédictions et évaluez sa performance de la même manière que pour la régression linéaire.

Analysez les résultats obtenus et comparez-les à ceux de la régression linéaire.

Forêts Aléatoires

Les forêts aléatoires sont une extension des arbres de décision. Elles combinent plusieurs arbres de décision pour améliorer les prédictions. Utilisez la classe *RandomForestRegressor* de la librairie scikit-learn pour créer un modèle de forêts aléatoires. Entraînez le modèle, faites des prédictions et évaluez sa performance de la même manière que pour les modèles précédents.

Analysez les résultats obtenus et comparez-les à ceux des autres modèles.

Recherche d'un modèle

Pour sélectionner le meilleur modèle, nous pouvons utiliser les classes *GridSearchCV* et *RandomizedSearchCV* de la librairie scikit-learn. Ces classes permettent de rechercher les meilleurs hyperparamètres d'un modèle en effectuant une recherche exhaustive ou aléatoire dans l'espace des hyperparamètres.

Utilisez ces classes pour rechercher le meilleur modèle parmi les trois modèles que vous avez utilisés. Analysez les résultats obtenus et discutez de la performance du modèle sélectionné.