Projet Machine learning

Objectif

L'objectif principal était de développer un programme complet en utilisant les techniques d'apprentissage automatique et de traitement de données abordées dans le cours. Plus précisément, il s'agissait de monter un ensemble de données (dataset) sur les performances des élèves, de définir un objectif spécifique lié à la prédiction de leurs notes, et de mettre en place plusieurs pipelines (chaînes de traitement) pour atteindre cet objectif.

Dataset

Après une longue recherche, j'ai finalement trouvé un dataset pertinent pour mon projet de prédiction des notes des élèves. Ce dataset est disponible sur GitHub, au sein du référentiel **ZooidsCompositePhysicalizations**, dans le dossier **Zooid_Vis/bin/data/**. Le fichier en question, nommé "student-dataset.csv", contient les données nécessaires pour mon analyse. Grâce à cette précieuse ressource, accessible à l'adresse

https://github.com/ShapeLab/ZooidsCompositePhysicalizations/blob/master/Zooid_Vis/bin/data/student-dataset.csv

je peux désormais poursuivre mon projet de manière plus efficace en disposant d'un ensemble de données approprié.

Explication de code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import numpy as np
```

Dans cette section, nous importons les bibliothèques et modules nécessaires pour notre projet de prédiction des notes des élèves.

import pandas as pd :

Nous importons la bibliothèque Pandas, qui est largement utilisée pour le traitement et l'analyse des données. Pandas nous permettra de charger et manipuler facilement notre ensemble de données (dataset) sur les performances des élèves.

from sklearn.model_selection import train_test_split :

Nous importons la fonction train_test_split du module model_selection de la bibliothèque scikit-learn. Cette fonction nous aidera à diviser notre ensemble de données en deux parties : un ensemble d'entraînement (training set) et un ensemble de test (test set). L'ensemble d'entraînement sera utilisé pour entraîner notre modèle de machine learning, tandis que l'ensemble de test servira à évaluer les performances du modèle sur des données qu'il n'a pas encore vues.

from sklearn.linear_model import LinearRegression :

Nous importons la classe LinearRegression du module linear_model de scikit-learn. Cette classe implémente l'algorithme de régression linéaire, qui est un modèle de machine learning couramment utilisé pour résoudre des problèmes de prédiction de valeurs numériques, comme dans notre cas où nous cherchons à prédire les notes des élèves.

from sklearn.metrics import mean_squared_error :

Nous importons la fonction mean_squared_error du module metrics de scikit-learn. Cette fonction calcule l'erreur quadratique moyenne (Mean Squared Error, MSE) entre les valeurs prédites par notre modèle et les valeurs réelles. L'erreur quadratique moyenne est une métrique couramment utilisée pour évaluer les performances d'un modèle de régression.

Après avoir importé ces bibliothèques et modules, nous pourrons charger notre ensemble de données, le diviser en ensembles d'entraînement et de test, entraîner un modèle de régression linéaire sur l'ensemble d'entraînement, faire des prédictions sur l'ensemble de test, et évaluer les performances du modèle à l'aide de l'erreur quadratique moyenne.

```
data = pd.read_csv('/content/student-dataset.csv')
```

Cette ligne de code utilise la fonction read_csv de la bibliothèque Pandas pour charger les données de notre ensemble de données (dataset) à partir d'un fichier CSV nommé "student-dataset.csv". Le chemin /content/student-dataset.csv indique l'emplacement du fichier CSV. Pandas lira le contenu de ce fichier et créera un objet DataFrame data contenant les données.

```
data = data.drop(['id','name','nationality','latitude','longitude','gender','age','city','ethnic.group'], axis=1)
data.fillna(data.mean(), inplace=True)
```

Cette section du code effectue deux opérations de prétraitement sur notre ensemble de données :

data = data.drop(['id','name','nationality','latitude','longitude','gender','age','city','ethnic.group'], axis=1)

Cette ligne de code supprime (drop) plusieurs colonnes de notre DataFrame data. Les colonnes supprimées sont 'id', 'name', 'nationality', 'latitude', 'longitude', 'gender', 'age', 'city', et 'ethnic.group'. Ces colonnes peuvent être considérées comme non pertinentes ou inutiles pour notre tâche de prédiction des notes des élèves. Le paramètre axis=1 indique que nous supprimons des colonnes (au lieu de supprimer des lignes, ce qui correspondrait à axis=0).

data.fillna(data.mean(), inplace=True)

Cette ligne de code remplace (fill) les valeurs manquantes (NaN) dans notre DataFrame data par la moyenne de chaque colonne respective. La méthode fillna de Pandas est utilisée pour effectuer cette opération. Le paramètre inplace=True signifie que les modifications sont effectuées directement sur le DataFrame data existant, sans en créer une copie.

En résumé, cette partie du code charge notre ensemble de données à partir d'un fichier CSV, supprime les colonnes non pertinentes pour notre tâche de prédiction, et remplace les valeurs manquantes par la moyenne de chaque colonne. Ces étapes de prétraitement sont souvent nécessaires pour préparer les données avant de les utiliser pour entraîner un modèle de machine learning.

```
X = data[['english.grade', 'math.grade', 'sciences.grade', 'language.grade', 'portfol
io.rating', 'coverletter.rating', 'refletter.rating']]
y = data['math.grade']
```

X = data[['english.grade', 'math.grade', 'sciences.grade', 'language.grade', 'portfolio.rating', 'coverletter.rating', 'refletter.rating']] :

Cette ligne crée un nouveau DataFrame X qui contient uniquement les colonnes spécifiées entre crochets. Ces colonnes représentent les caractéristiques (features) que nous utiliserons pour prédire la variable cible. Dans ce cas, les caractéristiques sont les notes en anglais, mathématiques, sciences, langue, ainsi que les notes pour le portfolio, la lettre de motivation et la lettre de recommandation.

y = data['math.grade'] :

Cette ligne crée une nouvelle Series y qui contient les valeurs de la colonne 'math.grade' du DataFrame data. Cette colonne représente la variable cible que nous voulons prédire, à savoir les notes en mathématiques.

La séparation des données en caractéristiques (X) et variable cible (y) est une étape essentielle dans les problèmes de machine learning supervisé. Les caractéristiques (X) sont les données d'entrée que le modèle utilisera pour apprendre et faire des prédictions, tandis que la variable cible (y) est la variable que nous voulons prédire en fonction des caractéristiques.

Cette séparation permet ensuite de diviser les données en ensembles d'entraînement et de test, d'entraîner le modèle sur l'ensemble d'entraînement, et d'évaluer ses performances sur l'ensemble de test. Dans ce cas, nous cherchons à prédire les notes en mathématiques des élèves en utilisant leurs autres notes et évaluations comme caractéristiques.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

Voici ce que signifient les différents éléments :

X_train, X_test, y_train, y_test sont les variables qui contiendront les données divisées en ensembles d'entraînement et de test.

X et **y** sont les DataFrames/Series que nous avons définis précédemment, contenant respectivement les caractéristiques (features) et la variable cible (target variable).

test_size=0.2 signifie que 20% des données seront utilisées pour l'ensemble de test, et les 80% restants pour l'ensemble d'entraînement.

random_state=42 est un paramètre qui permet de fixer la graine aléatoire pour obtenir des résultats reproductibles. Cela garantit que les mêmes données seront divisées de la même manière à chaque exécution du code.

Après cette ligne de code, nous aurons quatre nouvelles variables :

X_train : un DataFrame contenant les caractéristiques (features) de l'ensemble d'entraînement.

X_test : un DataFrame contenant les caractéristiques (features) de l'ensemble de test.

y_train : une Series contenant les valeurs de la variable cible pour l'ensemble d'entraînement.

y_test : une Series contenant les valeurs de la variable cible pour l'ensemble de test.

Cette division en ensembles d'entraînement et de test est cruciale pour évaluer les performances d'un modèle de machine learning de manière fiable. Le modèle sera entraîné sur l'ensemble d'entraînement, puis évalué sur l'ensemble de test, qui contient des données que le modèle n'a jamais vues auparavant. Cela permet de simuler les performances du modèle sur de nouvelles données et d'éviter le sur-apprentissage (overfitting).

```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

regressor = LinearRegression() : Cette ligne crée un objet regressor qui est une instance de la classe LinearRegression de la bibliothèque scikit-learn. Cette classe implémente l'algorithme de régression linéaire, qui est un modèle de machine learning supervisé utilisé pour résoudre des problèmes de régression (prédiction de valeurs numériques).

regressor.fit(X_train, y_train) : Cette ligne entraîne le modèle de régression linéaire regressor sur les données d'entraînement. X_train contient les caractéristiques (features) de l'ensemble d'entraînement, et y_train contient les valeurs de la variable cible correspondantes.

Le processus d'entraînement consiste à ajuster les coefficients du modèle de régression linéaire pour minimiser l'erreur entre les prédictions du modèle et les valeurs réelles de la variable cible dans l'ensemble d'entraînement. Cela se fait généralement en utilisant une technique d'optimisation comme la descente de gradient.

Après cette étape d'entraînement, le modèle regressor a appris les relations entre les caractéristiques (features) et la variable cible à partir des données d'entraînement. Il peut maintenant être utilisé pour faire des prédictions sur de nouvelles données.

Dans le cadre de votre projet de prédiction des notes des élèves, le modèle de régression linéaire apprendra les relations entre les notes dans d'autres matières, les évaluations de portfolio, lettres de motivation, etc. (caractéristiques) et les notes en mathématiques (variable cible). Une fois entraîné, ce modèle pourra être utilisé pour prédire les notes en mathématiques d'élèves supplémentaires en fonction de leurs caractéristiques.

```
y_pred = regressor.predict(X_test)
```

Voici ce que signifie cette ligne :

y_pred est une nouvelle variable qui contiendra les prédictions du modèle pour la variable cible (dans votre cas, les notes en mathématiques) sur l'ensemble de test.

regressor.predict(X_test) appelle la méthode predict du modèle de régression linéaire regressor en lui passant X_test comme argument.

X_test contient les caractéristiques (features) de l'ensemble de test, c'est-à-dire les données sur lesquelles le modèle n'a pas été entraîné.

Ainsi, le modèle regressor utilise les relations qu'il a apprises entre les caractéristiques et la variable cible pendant la phase d'entraînement pour faire des prédictions sur les nouvelles données de l'ensemble de test X_test.

Dans le cadre de votre projet de prédiction des notes des élèves, le modèle utilisera les notes dans d'autres matières, les évaluations de portfolio, lettres de motivation, etc. (caractéristiques) de l'ensemble de test X_test pour prédire les notes en mathématiques correspondantes. Ces prédictions seront stockées dans la variable y_pred.

Cette étape est cruciale pour évaluer les performances du modèle sur des données qu'il n'a jamais vues auparavant, simulant ainsi son comportement sur de nouvelles données réelles. Les prédictions y_pred peuvent ensuite être comparées aux valeurs réelles de la variable cible dans l'ensemble de test pour calculer des métriques d'évaluation comme l'erreur quadratique moyenne (mean squared error) ou le coefficient de détermination (R^2).

```
print('Mean Absolute Error:',
metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test,
y_pred))
print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred)) :

Cette ligne calcule et affiche l'erreur absolue moyenne (Mean Absolute Error, MAE) entre les prédictions du modèle y_pred et les valeurs réelles de la variable cible dans l'ensemble de test y_test. La MAE est la moyenne des valeurs absolues des différences entre les prédictions et les valeurs réelles. Plus la MAE est petite, plus le modèle est précis.

print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred)) :

Cette ligne calcule et affiche l'erreur quadratique moyenne (Mean Squared Error, MSE) entre les prédictions du modèle y_pred et les valeurs réelles de la variable cible dans l'ensemble de test y_test. La MSE est la moyenne des carrés des différences entre les prédictions et les valeurs réelles. Plus la MSE est petite, plus le modèle est précis.

print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred))) :

Cette ligne calcule et affiche la racine carrée de l'erreur quadratique moyenne (Root Mean Squared Error, RMSE) entre les prédictions du modèle y_pred et les valeurs réelles de la variable cible dans l'ensemble de test y_test. La RMSE est la racine carrée de la MSE et a la même unité que la variable cible, ce qui la rend plus interprétable que la MSE.

Ces métriques sont couramment utilisées pour évaluer les performances d'un modèle de régression. Elles mesurent l'écart entre les prédictions du modèle et les valeurs réelles, ce qui permet de quantifier la précision du modèle

Dans le cadre de votre projet de prédiction des notes des élèves, ces métriques vous permettront d'évaluer à quel point les prédictions du modèle de régression linéaire pour les notes en mathématiques sont proches des notes réelles.

Une faible **MAE, MSE et RMSE** indiquent que le modèle est performant et capable de prédire avec précision les notes en mathématiques à partir des autres caractéristiques (notes dans d'autres matières, évaluations de portfolio, etc.). Inversement, des valeurs élevées de ces métriques signalent que le modèle n'est pas suffisamment précis et qu'il pourrait être nécessaire d'ajuster ou de changer d'approche.

```
def predict_end_semester_percentage(student_details):
    # student_details should be a dictionary with keys matching
the feature columns
    input_data = pd.DataFrame([student_details])
    prediction = regressor.predict(input_data)
    return prediction[0]
```

Voici une explication détaillée de cette fonction :

def predict_end_semester_percentage(student_details):

Cette ligne définit une nouvelle fonction nommée predict_end_semester_percentage qui prend un argument student_details.

input_data = pd.DataFrame([student_details]) :

Cette ligne crée un nouveau DataFrame Pandas input_data à partir du dictionnaire student_details. Le dictionnaire est encapsulé dans une liste [student_details] car la fonction pd.DataFrame attend un itérable (comme une liste) de dictionnaires.

prediction = regressor.predict(input_data) :

Cette ligne utilise le modèle de régression linéaire regressor (entraîné précédemment) pour faire une prédiction sur les données input_data. La méthode predict renvoie un objet NumPy contenant les prédictions pour chaque ligne du DataFrame input_data.

return prediction[0] :

Cette ligne renvoie le premier (et unique) élément de l'objet NumPy prediction, qui correspond à la prédiction pour le dictionnaire student_details.

Pour utiliser cette fonction, vous devez fournir un dictionnaire student_details contenant les valeurs des caractéristiques (features) pour un étudiant donné. Les clés de ce dictionnaire doivent correspondre aux noms des colonnes de caractéristiques utilisées pour entraîner le modèle (par exemple, 'english.grade', 'math.grade', 'sciences.grade', etc.).

La fonction crée alors un DataFrame Pandas à partir de ce dictionnaire, puis utilise le modèle de régression linéaire regressor pour faire une prédiction sur ces données. Le résultat de la prédiction est renvoyé. Cette fonction peut être utile pour prédire les performances futures d'un étudiant en fonction de ses résultats et évaluations précédents, ce qui peut aider à identifier les élèves qui pourraient avoir besoin d'un soutien supplémentaire ou, au contraire, ceux qui excellent.