TP4

Classification supervisée avec un problème concret

Objectifs du TP

Apprendre à résoudre un problème de classification supervisée en suivant toutes les étapes :

1. Collecte et préparation des données.
2. Analyse exploratoire des données (EDA).
3. Création et entraînement d'un modèle de classification.
4. Évaluation du modèle.
5. Implémentation et visualisation des résultats.

Les étudiants doivent travailler en groupe de 2 à 3 et rendre un compte rendu à déposer sur Classroom. Le compte rendu devrait lister le code dans un notebook google colab et ce en plus d'un rapport d'analyse avec graphiques et interprétations.

Sujet :

On souhaite prédire si un étudiant réussira à un examen en fonction de ses caractéristiques.

- ***Plateforme*** : Google Colab
- **Bibliothèques** Python: *pandas, numpy, matplotlib, seaborn, scikit-learn.*

I. Collecte et préparation des données

Dataset proposé (Utilisez le fichier .csv en attaché) :

```
Heures_etude,Participation_classe,Presence,Resultat
2,5,80,0
3,7,85,0
4,8,70,1
6,9,90,1
1,4,60,0
8,10,95,1
5,6,75,1
2,6,65,0
7,8,80,1
3,5,50,0
```

II. Importation et exploration initiale

```
# Importation des bibliothèques
import pandas as pd
import numpy as np

# Chargement des données (remplacer 'chemin_vers_fichier' par le chemin réel)
url = "https://raw.githubusercontent.com/username/dataset_path.csv"  # Si vous hébergez le fichier
df = pd.read_csv(url)

# Aperçu des données
print(df.head())

# Vérification des valeurs manquantes
print(df.isnull().sum())

# Statistiques descriptives
print(df.describe())
```

III. Analyse exploratoire des données (EDA)

Visualisation

```
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution des résultats
sns.countplot(data=df, x='Resultat')
plt.title('Distribution des résultats (0 = échec, 1 = réussite)')
plt.show()

# Relation entre les heures d'étude et le résultat
sns.scatterplot(data=df, x='Heures_etude', y='Presence', hue='Resultat',
palette='coolwarm')
plt.title('Relation entre Présence et Heures d\'étude')
plt.show()

# Matrice de corrélation
corr = df.corr()
sns.heatmap(corr, annot=True, cmap='Blues')
plt.title('Matrice de corrélation')
plt.show()
```

IV. Préparation des données

Normalisation et séparation des ensembles

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Séparation des caractéristiques (X) et de la cible (y)
X = df[['Heures_etude', 'Participation_classe', 'Presence']]
y = df['Resultat']

# Séparation en ensemble d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Normalisation des données
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

V. Modélisation

Création et entraînement des modèles

```
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier

# Modèles
models = {
    "Logistic Regression": LogisticRegression(),
    "K-Nearest Neighbors": KNeighborsClassifier(n_neighbors=3),
    "Decision Tree": DecisionTreeClassifier(max_depth=5)
}

# Entraînement des modèles
for name, model in models.items():
    model.fit(X_train, y_train)
    print(f"{name} entraîné.")
```

VI. Évaluation

Evaluation des performances

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,
ConfusionMatrixDisplay

for name, model in models.items():
    y_pred = model.predict(X_test)
    print(f"\nModèle : {name}")
    print("Accuracy :", accuracy_score(y_test, y_pred))
    print("Rapport de classification :\n", classification_report(y_test, y_pred))

    # Matrice de confusion
    cm = confusion_matrix(y_test, y_pred)
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=model.classes_)
    disp.plot(cmap='Blues')
    plt.title(f"Matrice de confusion - {name}")
    plt.show()
```

VII. Implémentation et test final

Interface interactive avec Streamlit (facultatif)

Pour tester les prédictions avec une interface, utiliser **Streamlit** :

```
# Installer Streamlit dans Google Colab
!pip install streamlit

# Interface simple
import streamlit as st

st.title("Prédiction de réussite d'examen")
hours = st.number_input("Heures d'étude :", 0, 10)
participation = st.number_input("Participation en classe (1-10) :", 1, 10)
presence = st.number_input("Présence (%) :", 0, 100)

if st.button("Prédire"):
    input  data = scaler.transform([[hours, participation, presence]])
    prediction = models["Logistic Regression"].predict(input_data)[0]
    result = "Réussite" if prediction == 1 else "Échec"
    st.write(f"Résultat prédit : {result}")
```

Conclusion :

……..

……..

……..

……..

……..

……..

……..

……..

……..

……..