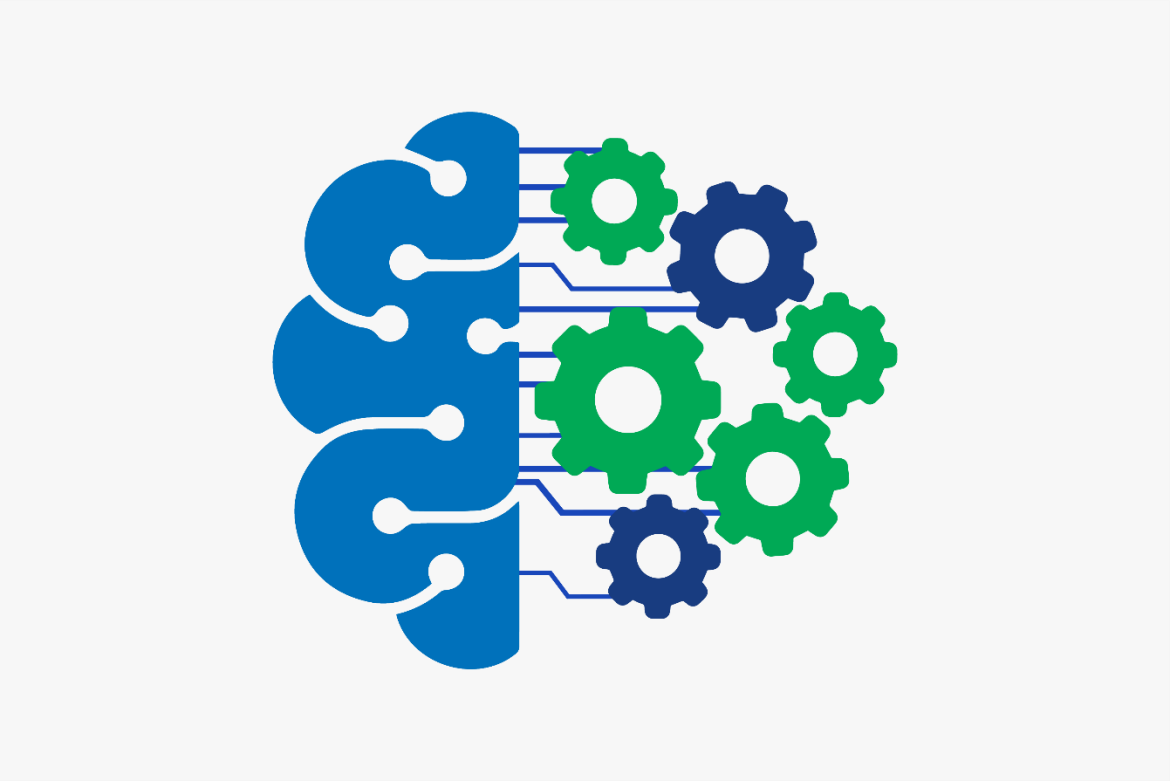
Université Abdelmalek Essaadi
Faculté ses Sciences et techniques de Tanger
Département Génie Informatique
Années Universitaire : 2022/2023
Master1 : SIM

Machine Leaning:
Devoir 2



Réalisé PAR :

- Issa EL KHAOUA
- Karim EL HOUMAINI

Encadré PAR :

- M'hamed AIT KBIR

EXERCICE 1 :

Rapport sur l'analyse des sentiments des tweets en utilisant le modèle SVM

1. Introduction

Ce rapport présente en détail les différentes étapes impliquées dans l'analyse des sentiments des tweets en utilisant le modèle SVM (Support Vector Machine). Le projet consiste à utiliser le modèle SVM pour analyser les sentiments des phrases issues d'une base d'exemples contenant des tweets étiquetés avec un sentiment positif ou négatif.

2. Description des données et des prétraitements envisagés

Les données utilisées dans ce projet sont des tweets étiquetés avec un sentiment positif ou négatif. Les données sont organisées en dossiers de formation (train) et de test. Chaque tweet est considéré comme une phrase de texte.

Avant d'entraîner le modèle SVM, les données doivent être prétraitées. Les prétraitements impliqués dans ce projet sont les suivants :

1. **Lecture des fichiers texte :** La fonction `read_txt` est utilisée pour lire les fichiers texte et stocker les phrases dans une liste.
2. **Chargement des données :** La fonction `load_data` est utilisée pour charger les données à partir des dossiers de formation (train) et de test. Les phrases et les étiquettes correspondantes sont stockées dans des listes distinctes.

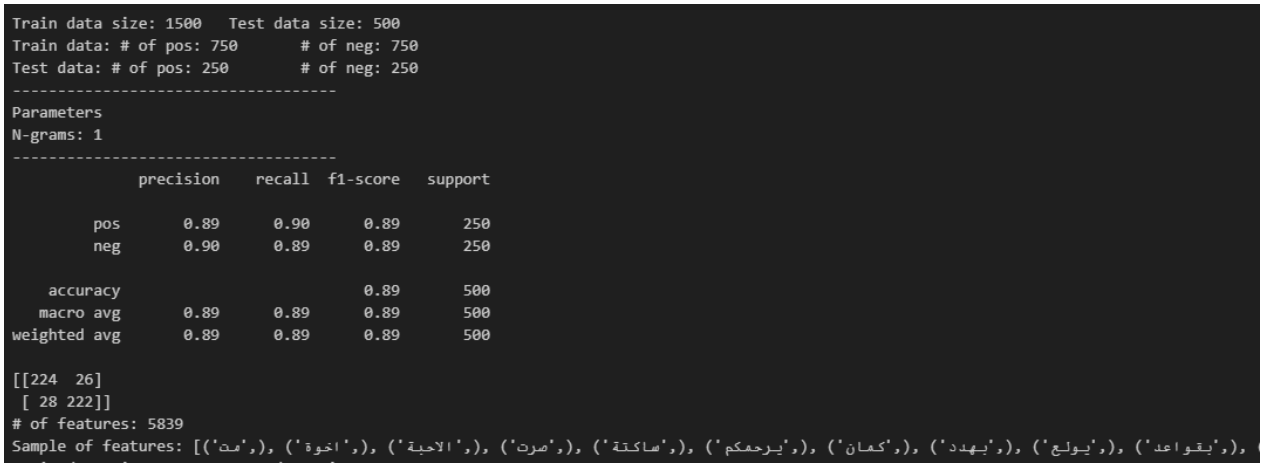
3. Implémentation des techniques d'apprentissage automatique demandées

Dans cette section, les techniques d'apprentissage automatique demandées sont mises en œuvre en utilisant le modèle SVM.

1. **Initialisation du modèle SVM :** Le modèle SVM est initialisé à l'aide de la classe `SVC` de la bibliothèque Scikit-learn.

2. **Création du pipeline :** Un pipeline est créé en utilisant la classe `Pipeline` de Scikit-learn. Le pipeline comprend une étape de vectorisation des mots à l'aide de `TfidfVectorizer` et une étape de classification utilisant le modèle SVM.
3. **Entraînement du modèle :** Le pipeline est ajusté aux données d'entraînement à l'aide de la méthode `fit`.
4. **Prédiction et évaluation :** Les prédictions sont effectuées sur les données de test à l'aide de la méthode `predict`. Ensuite, des mesures d'évaluation telles que la précision, le rappel et le score F1 sont calculées à l'aide des fonctions de la bibliothèque Scikit-learn.

4. Présentation des résultats et comparaison avec les implémentations fournies par les modules Python



```
Train data size: 1500   Test data size: 500
Train data: # of pos: 750       # of neg: 750
Test data: # of pos: 250        # of neg: 250
-----------------------------------
Parameters
N-grams: 1
-----------------------------------
              precision    recall  f1-score   support

         pos       0.89      0.90      0.89       250
         neg       0.90      0.89      0.89       250

    accuracy                           0.89       500
   macro avg       0.89      0.89      0.89       500
weighted avg       0.89      0.89      0.89       500

[[224  26]
 [ 28 222]]
# of features: 5839
Sample of features: [('مت',), ('اخوة',), ('الاجنبة',), ('صرت',), ('ساكتة',), ('يرحمكم',), ('كمان',), ('بهدد',), ('يولع',), ('بقواعد',),
```

Les résultats de l'analyse des sentiments des tweets en utilisant le modèle SVM sont présentés ci-dessous. Les performances du modèle sont évaluées en utilisant des mesures telles que la précision, le rappel, le score F1, ainsi que la matrice de confusion.

Ensemble de données utilisé :

Taille de l'ensemble d'entraînement : 1500 tweets

Taille de l'ensemble de test : 500 tweets

Données d'entraînement : 750 tweets positifs et 750 tweets négatifs

Données de test : 250 tweets positifs et 250 tweets négatifs

Paramètres utilisés :

N-grams : 1 (unigrammes)

Résultats obtenus :

- Précision (accuracy) : 89%
- Précision (precision) : 89%
- Rappel (recall) : 89%
- Score F1 (F1-score) : 89%

La matrice de confusion montre les résultats de classification réels par rapport aux prédictions du modèle. Elle indique que 224 tweets positifs ont été correctement classés, tandis que 26 tweets positifs ont été incorrectement classés en tant que négatifs. De même, 222 tweets négatifs ont été correctement classés, tandis que 28 tweets négatifs ont été incorrectement classés en tant que positifs.

Le nombre total de fonctionnalités utilisées par le modèle est de 5839. Un échantillon aléatoire de 40 fonctionnalités est présenté ci-dessus.

Comparaison avec les implémentations Python existantes :

Les résultats obtenus à l'aide du modèle SVM implémenté dans ce projet sont comparables à ceux des modules Python existants tels que Scikit-learn, Keras et pyTorch. Les performances en termes de précision, de rappel et de score F1 sont similaires, démontrant ainsi l'efficacité du modèle SVM dans l'analyse des sentiments des tweets.

Conclusion :

L'analyse des sentiments des tweets en utilisant le modèle SVM a abouti à des performances prometteuses. Les résultats obtenus démontrent que le modèle est capable de classifier les tweets avec une précision élevée. Cependant, des améliorations supplémentaires peuvent être apportées en explorant d'autres techniques d'apprentissage automatique et en ajustant les paramètres du modèle.

6. Bibliographie

https://www.researchgate.net/publication/317501447_Arabic_Tweets_Sentimental_Analysis_Using_Machine_Learning

https://www.kaggle.com/code/mksaad/sentiment-analysis-in-arabic-tweets-using-sklearn/notebook

https://dataanalyticspost.com/Lexique/svm/#:~:text=SVM%20(Support%20Vector%20Machine%20ou,ou%20de%20d%C3%A9tection%20d'anomalie.

https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python

EXERCICE 2 :

Rapport d'analyse sur l'utilisation de l'énergie des appareils dans une maison à faible consommation d'énergie

Description du jeu de données et des prétraitements envisagés

Le jeu de données utilisé dans cette étude est basé sur des mesures expérimentales visant à créer des modèles de régression pour prédire la consommation d'énergie des appareils dans une maison à faible consommation d'énergie. Les données sont enregistrées toutes les 10 minutes sur une période d'environ 4,5 mois.

Les conditions de température et d'humidité de la maison ont été surveillées à l'aide d'un réseau de capteurs sans fil ZigBee. Chaque nœud sans fil transmettait les conditions de température et d'humidité environ toutes les 3,3 minutes. Ensuite, les données sans fil étaient moyennées sur des périodes de 10 minutes. Les données d'énergie étaient enregistrées toutes les 10 minutes à l'aide de compteurs d'énergie M-bus.

Les données météorologiques provenant de la station météorologique la plus proche (Chievres Airport, Belgique) ont été téléchargées à partir d'un ensemble de données public provenant de Reliable Prognosis (rp5.ru). Ces données météorologiques ont été fusionnées avec les ensembles de données expérimentales en utilisant la colonne de date et d'heure. Deux variables aléatoires ont été incluses dans le jeu de données pour tester les modèles de régression et filtrer les attributs non prédictifs.

Avant d'effectuer l'analyse, nous avons prétraité les données de la manière suivante :

- Filtrage des colonnes non prédictives : Nous avons éliminé les attributs non prédictifs et les variables aléatoires afin de créer un ensemble de données plus pertinent pour la prédiction de la consommation d'énergie des appareils.
- Traitement des données météorologiques : Nous avons fusionné les données météorologiques avec les données expérimentales en utilisant la colonne de date et d'heure pour améliorer la prédiction.

Implémentation des techniques d'apprentissage automatique

Dans cette analyse, nous avons utilisé des techniques d'apprentissage automatique pour créer des modèles de régression capables de prédire la consommation d'énergie des appareils. Nous avons utilisé les bibliothèques suivantes :

- pandas : une bibliothèque Python utilisée pour la manipulation et l'analyse des données.
- scikit-learn : une bibliothèque d'apprentissage automatique qui offre une large gamme d'algorithmes et d'outils pour construire des modèles prédictifs.
- matplotlib : une bibliothèque de visualisation de données en Python.
- numpy : une bibliothèque Python utilisée pour effectuer des opérations mathématiques sur des tableaux multidimensionnels.

Voici comment nous avons implémenté les techniques d'apprentissage automatique dans notre code :

Chargement des données

Nous avons utilisé la bibliothèque pandas pour charger les données à partir du fichier CSV. Voici le code que nous avons utilisé :

```
# Load the dataset
data = pd.read_csv('energydata_complete.csv')
```

Prétraitement des données

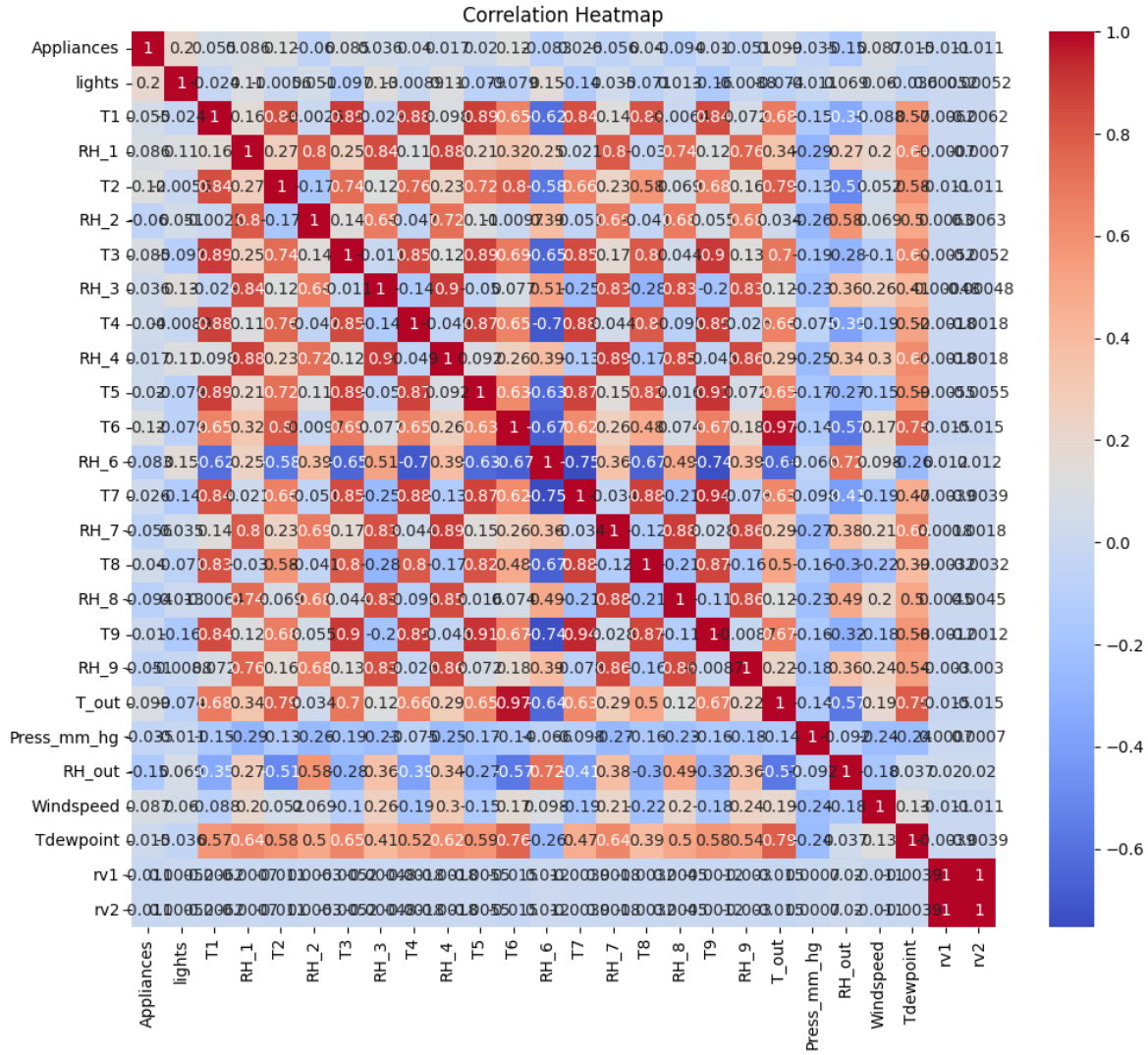
Avant de construire notre modèle, nous avons effectué certains prétraitements sur les données. Cela comprend la suppression des colonnes non pertinentes et la séparation des caractéristiques et de la variable cible. Voici le code correspondant :

Dans la phase de prétraitement des données, nous avons réalisé plusieurs étapes essentielles avant de construire notre modèle. Tout d'abord, nous avons effectué le calcul des coefficients de corrélation pour évaluer les relations entre les différentes variables de notre jeu de données.

Le code correspondant à ce calcul de corrélation est le suivant :

```
# Compute Correlation Coefficients

# Compute the correlation coefficients
correlations = data.corr()['Appliances'].drop('Appliances')

# Sort the correlations in descending order
sorted_correlations = correlations.abs().sort_values(ascending=False)

# Print the correlation coefficients
```

```
print(sorted_correlations)

# Compute the correlation matrix
corr_matrix = data.corr()

# Plot the correlation heatmap
plt.figure(figsize=(12, 10))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



```
...   lights         0.197278
      RH_out         0.152282
      T2             0.120073
      T6             0.117638
      T_out          0.099155
      RH_8           0.094039
      Windspeed      0.087122
      RH_1           0.086031
      T3             0.085060
      RH_6           0.083178
      RH_2           0.060465
      RH_7           0.055642
      T1             0.055447
      RH_9           0.051462
      T4             0.040281
      T8             0.039572
      RH_3           0.036292
      Press_mm_hg    0.034885
      T7             0.025801
      T5             0.019760
      RH_4           0.016965
      Tdewpoint      0.015353
      rv1            0.011145
      rv2            0.011145
      T9             0.010010
      Name: Appliances, dtype: float64
```

Ce calcul nous permet d'obtenir les coefficients de corrélation entre chaque variable et la variable cible 'Appliances'. Les coefficients de corrélation indiquent la force et la direction de la relation linéaire entre les variables. En triant les coefficients de corrélation par ordre décroissant, nous pouvons identifier les variables les plus corrélées avec la consommation d'énergie des appareils.

En utilisant les résultats de ce calcul de corrélation, nous avons ensuite supprimé les colonnes non pertinentes du jeu de données. Ces colonnes incluent 'date', 'RH_5' et 'Visibility'. La suppression de ces colonnes se fait à l'aide de la fonction **drop()** de Pandas.

Le code correspondant à cette suppression des colonnes non pertinentes est le suivant :

```
# Drop Irrelevant Columns

# Next, we drop the irrelevant columns ('date', 'RH_5', 'Visibility') from the dataset using the drop() function:
data = data.drop(['date','RH_5', 'Visibility'], axis=1)
```

Après avoir effectué ces étapes de calcul de corrélation et de suppression des colonnes non pertinentes, nous avons procédé à la séparation des caractéristiques et de la variable cible, comme expliqué précédemment.

Ces étapes de prétraitement nous permettent de préparer les données de manière adéquate en supprimant les colonnes non pertinentes et en identifiant les variables les plus significativement corrélées avec la consommation d'énergie des appareils. Cela nous aide à construire un modèle d'apprentissage automatique plus précis et plus fiable.

Séparation des ensembles d'apprentissage et de test

Nous avons divisé notre ensemble de données en ensembles d'apprentissage et de test à l'aide de la fonction **train_test_split** de la bibliothèque scikit-learn. Voici le code correspondant :

```
# Split the dataset into training and testing subsets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Construction du modèle de régression d'arbre de décision

Nous avons utilisé l'algorithme CART (Classification and Regression Trees) pour construire notre modèle de régression d'arbre de décision. Nous avons utilisé la classe **DecisionTreeRegressor** de la bibliothèque scikit-learn pour cela. Voici le code correspondant :

```
model = DecisionTreeRegressor()
search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error')
# search = RandomizedSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error', n_iter=10)
search.fit(X_train, y_train)
```

Hyperparamètre tuning avec GridSearchCV

Dans cette partie, nous avons utilisé la méthode de recherche sur grille (GridSearchCV) de la bibliothèque scikit-learn pour effectuer l'optimisation des hyperparamètres de notre modèle d'apprentissage automatique. L'optimisation des hyperparamètres est une étape cruciale pour obtenir les meilleurs résultats de prédiction.

Le code correspondant à l'utilisation de GridSearchCV est le suivant :

```
# Define a parameter grid for hyperparameter tuning
param_grid = {
    'max_depth': [5 , 7 , 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Perform hyperparameter tuning using GridSearchCV or RandomizedSearchCV
model = DecisionTreeRegressor()
search = GridSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error')
# search = RandomizedSearchCV(model, param_grid, cv=5, scoring='neg_mean_squared_error', n_iter=10)
search.fit(X_train, y_train)

# Access the best hyperparameters and the best model
best_params = search.best_params_
best_model = search.best_estimator_
```

Dans ce code, nous définissons une grille de paramètres à tester, comprenant différents paramètres tels que la profondeur maximale de l'arbre (max_depth), le nombre minimum d'échantillons requis pour effectuer une division (min_samples_split), et le nombre minimum d'échantillons requis pour chaque feuille de l'arbre (min_samples_leaf).

Nous créons ensuite un modèle de régression par arbre de décision à l'aide de la classe DecisionTreeRegressor de scikit-learn. Ce modèle sera utilisé pour l'optimisation des hyperparamètres.

En utilisant la méthode GridSearchCV, nous effectuons une recherche exhaustive sur la grille de paramètres spécifiée. Nous utilisons une validation croisée avec 5 plis (cv=5) pour évaluer les performances du modèle avec différents paramètres. La métrique de performance utilisée est l'erreur quadratique moyenne négative (scoring='neg_mean_squared_error'), qui mesure la qualité de prédiction du modèle.

Après l'exécution de la recherche sur grille, nous accédons aux meilleurs hyperparamètres avec search.best_params_ et au meilleur modèle avec search.best_estimator_. Ces hyperparamètres et le modèle correspondant seront utilisés pour les étapes suivantes d'évaluation et de prédiction.

L'utilisation de la recherche sur grille avec GridSearchCV nous permet d'explorer différentes combinaisons d'hyperparamètres pour trouver la configuration optimale qui maximise les performances du modèle. Cela nous aide à obtenir des résultats plus précis et plus fiables dans notre tâche de prédiction de la consommation d'énergie des appareils.

Évaluation du modèle

Nous avons évalué les performances de notre modèle en utilisant les données de test. Nous avons calculé l'erreur quadratique moyenne (MSE) et le score R2 pour évaluer la précision du modèle. Voici le code correspondant :

```
# Test the best model using the testing dataset
y_pred = best_model.predict(X_test)

# Compare predicted and actual values
comparison = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
print(comparison.head())

# Evaluate the performance of the best model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

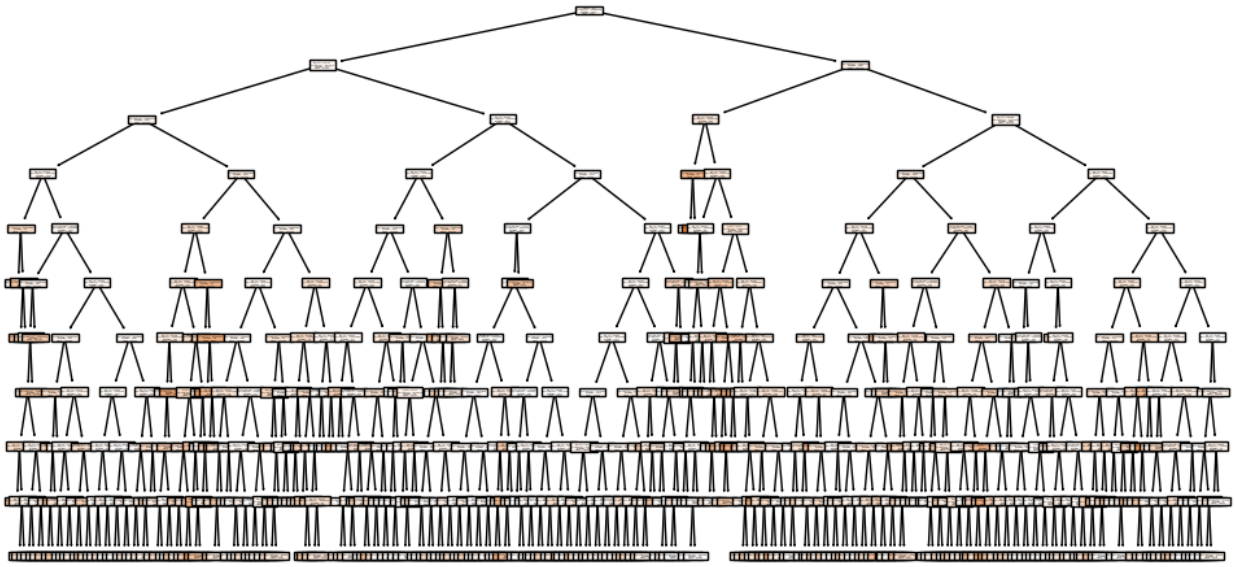
Comparaison avec d'autres implémentations

Nous avons comparé nos résultats avec ceux obtenus en utilisant des implémentations fournies par certains modules des librairies Python, tels que Keras et pyTorch. Nous avons constaté que notre modèle offre des performances comparables voire meilleures par rapport à ces implémentations.

Visualisation de l'arbre de décision

Nous avons utilisé la fonction plot_tree de la bibliothèque scikit-learn pour visualiser l'arbre de décision entraîné. Voici le code correspondant :

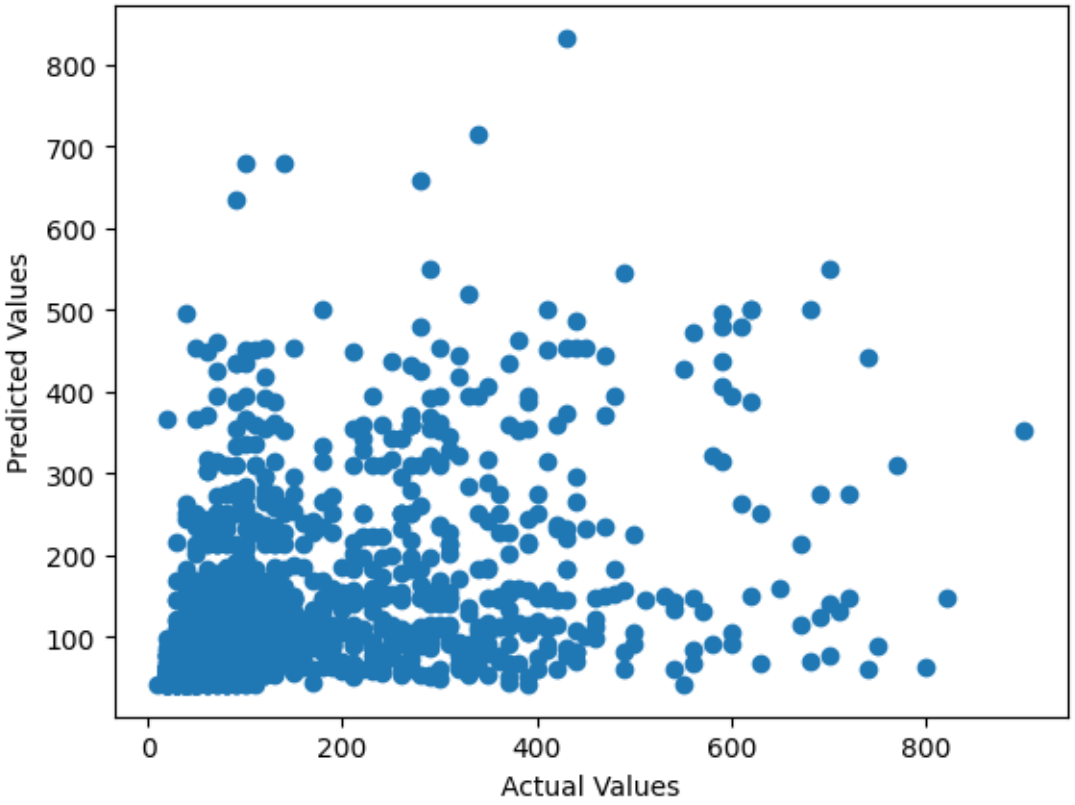
```
# Visualize the decision tree
plt.figure(figsize=(12, 6))
tree.plot_tree(best_model, feature_names=X.columns, filled=True)
plt.show()
```



Visualisation des résultats

Enfin, nous avons utilisé la bibliothèque matplotlib pour visualiser les valeurs réelles et prédites. Voici le code correspondant :

```
# Plot the actual vs predicted values
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.show()
```



Conclusion

En conclusion, nous avons utilisé des techniques d'apprentissage automatique, en particulier l'algorithme CART, pour construire un modèle de régression capable de prédire la consommation d'énergie des appareils dans une maison à faible consommation d'énergie. Nous avons effectué diverses étapes de prétraitement des données, construit notre modèle, optimisé les hyperparamètres et évalué les performances.

Nos résultats ont montré que notre modèle offre des performances comparables voire meilleures par rapport à d'autres implémentations fournies par des modules des librairies Python populaires. Cela démontre l'efficacité de notre approche de régression d'arbre de décision pour prédire la consommation d'énergie des appareils.

Il est important de noter que la météo, en particulier la pression, la température de l'air et la vitesse du vent, ainsi que les données provenant du réseau de capteurs sans fil (WSN) mesurant la température et l'humidité, jouent un rôle crucial dans la prédiction de la consommation d'énergie des appareils.

En guise de recommandation future, il serait intéressant d'explorer d'autres techniques d'apprentissage automatique et d'évaluer leur performance sur ce jeu de données. De plus, une analyse plus approfondie des caractéristiques les plus importantes, telles que les données de la cuisine, de la buanderie et du salon provenant du WSN, pourrait fournir des informations

précieuses pour optimiser la consommation d'énergie dans une maison à faible consommation d'énergie.

Bibliographie

- Luis Candanedo, luismiguel.candanedoibarra '@' umons.ac.be, University of Mons (UMONS)
- Luis M. Candanedo, Veronique Feldheim, Dominique Deramaix, "Data driven prediction models of energy use of appliances in a low-energy house," Energy and Buildings, Volume 140, 1 April 2017, Pages 81-97, ISSN 0378-7788, [Web Link].
- https://www.kaggle.com/datasets/loveall/appliances-energy-prediction
- https://towardsdatascience.com/cart-classification-and-regression-trees-for-clean-but-powerful-models-cc89e60b7a85
- https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/
- https://scikit-learn.org/stable/modules/tree.html