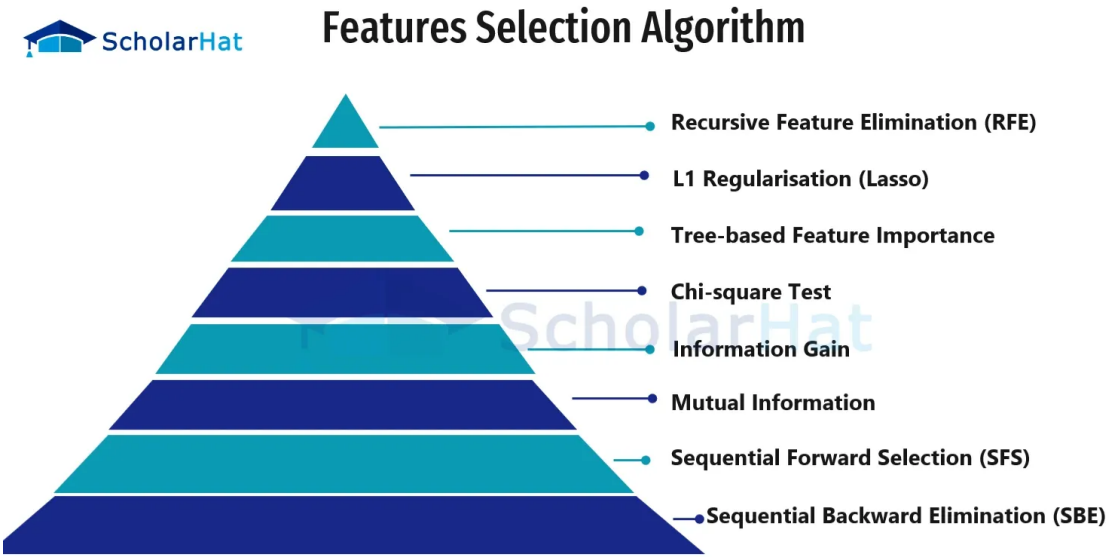
Feature Selection Techniques in Machine Learning

September 19, 2024

1 Feature Selection Techniques in Machine Learning

Découvrez les techniques de sélection des caractéristiques dans l'apprentissage automatique, y compris les méthodes basées sur les filtres, les enveloppes et les méthodes intégrées.

La sélection des caractéristiques est une technique utilisée pour réduire le nombre de variables d'entrée lors de l'élaboration d'un modèle prédictif. Elle est particulièrement importante dans les situations où les ensembles de données ont une dimension élevée. En sélectionnant un sous-ensemble de caractéristiques pertinentes, nous pouvons réduire le coût de calcul, éviter l'ajustement excessif et améliorer les capacités de généralisation du modèle.



1.1 Avantages de la sélection des caractéristiques

1. • **Amélioration des performances du modèle** : En réduisant le nombre de caractéristiques non pertinentes, nous réduisons le bruit dans les données, ce qui permet au modèle d'apprendre plus efficacement à partir des informations pertinentes.
2. • **Réduction de la suradaptation** : En éliminant le bruit et les données redondantes, la sélection des caractéristiques permet de réduire la complexité du modèle et le modèle est moins enclin à apprendre le bruit dans les données d'apprentissage, ce qui permet une meilleure généralisation à de nouvelles données.

3. • **Amélioration de l'interprétabilité** : La simplification du modèle en utilisant moins de caractéristiques facilite la compréhension du modèle et l'interprétation des résultats.

1.2 Méthodes de sélection des caractéristiques

Il existe plusieurs méthodes de sélection des caractéristiques, chacune ayant ses avantages et ses applications appropriées. Les principales catégories comprennent les méthodes de filtrage, les méthodes d'enveloppement et les méthodes intégrées.

- **Méthodes de filtrage** : Ces méthodes évaluent la pertinence des caractéristiques en examinant les propriétés intrinsèques des données. Des techniques telles que les coefficients de corrélation, les tests du Khi-deux et l'information mutuelle sont couramment utilisées.
- **Méthodes d'enveloppement** : Elles consistent à sélectionner les caractéristiques en fonction des performances du modèle. Les méthodes enveloppantes, telles que l'élimination récursive des caractéristiques (RFE) et la sélection avant ou arrière, testent de manière itérative différents sous-ensembles de caractéristiques et évaluent les performances du modèle.
- **Méthodes intégrées** : Ces méthodes effectuent une sélection des caractéristiques dans le cadre du processus d'apprentissage du modèle. Des techniques telles que Lasso (régularisation L1) et les algorithmes basés sur les arbres de décision (par exemple, Random Forests) effectuent intrinsèquement une sélection des caractéristiques.

1.3 Filter Methods

Les méthodes de filtrage évaluent la pertinence des caractéristiques en examinant leurs propriétés statistiques par rapport à la variable cible. Les techniques les plus courantes sont les suivantes :

- **Correlation Coefficient**: Mesure la relation linéaire entre chaque caractéristique et la variable cible.
- **Chi-Square Test**: Évalue l'indépendance entre les caractéristiques catégorielles et la variable cible.
- **Information mutuelle** : Quantifie la quantité d'information partagée entre chaque caractéristique et la variable cible.

Ces méthodes sont efficaces sur le plan informatique et sont particulièrement utiles pour les étapes de prétraitement avant l'application de modèles plus complexes.

1.4 Wrapper Methods

Les méthodes d'enveloppement utilisent un modèle prédictif pour évaluer l'importance des caractéristiques. Elles impliquent la sélection ou la suppression itérative de caractéristiques et l'évaluation des performances du modèle. Les techniques les plus courantes sont les suivantes : - **Forward Selection :** Commence avec aucune caractéristique et ajoute une caractéristique à la fois, en évaluant la performance du modèle à chaque étape. - **Backward Elimination :** Commence avec toutes les caractéristiques et enlève une caractéristique à la fois, en évaluant la performance du modèle à chaque étape. - **Recursive Feature Elimination (RFE):** Supprime de manière itérative les caractéristiques les moins importantes sur la base des coefficients du modèle ou de l'importance des caractéristiques.

Les méthodes d'enveloppement tendent à produire des modèles plus performants mais sont plus coûteuses en termes de calcul que les méthodes de filtrage.

1.5 Embedded Methods

Les méthodes intégrées effectuent la sélection des caractéristiques dans le cadre du processus d'apprentissage du modèle. Ces méthodes sont intégrées dans l'algorithme lui-même. Les techniques les plus courantes sont les suivantes : - **Lasso Regression:** Ajoute une pénalité au modèle s'il a trop de caractéristiques, réduisant ainsi certains coefficients de caractéristiques à zéro et sélectionnant ainsi un modèle plus simple. - **Decision Trees and Random Forests :** Des algorithmes tels que Random Forests et Gradient Boosting Trees effectuent automatiquement la sélection des caractéristiques en tenant compte de leur importance au cours du processus de construction du modèle.

Les méthodes intégrées sont efficaces et produisent souvent des sous-ensembles de caractéristiques robustes.

1.6 Applications concrètes de la sélection de caractéristiques

Les méthodes de sélection des caractéristiques sont très importantes dans divers domaines : santé, finance, marketing, etc :

1. Soins de santé : Dans le domaine de la santé, la sélection des caractéristiques est essentielle pour développer des modèles prédictifs permettant de diagnostiquer des maladies, de prédire les résultats pour les patients et de personnaliser les traitements. Par exemple, pour le diagnostic du cancer du sein, la sélection des caractéristiques les plus pertinentes à partir des données d'imagerie médicale et des dossiers des patients peut conduire à des modèles plus précis et plus faciles à interpréter, ce qui améliore en fin de compte les soins prodigués aux patients.
2. La finance : Dans le domaine de la finance, la sélection des caractéristiques permet de développer des modèles d'évaluation du crédit, de détection des fraudes et de prédiction des marchés boursiers. En identifiant les indicateurs financiers les plus pertinents, ces modèles peuvent faire des prédictions plus précises et fournir des informations précieuses pour la prise de décision.
3. Le marketing : En marketing, la sélection des caractéristiques est utilisée pour prédire le comportement des clients, segmenter les marchés et optimiser les campagnes de marketing. En sélectionnant les caractéristiques les plus pertinentes à partir des données des clients, telles que l'historique des achats et les informations démographiques, les entreprises peuvent créer des stratégies de marketing plus efficaces et plus ciblées.

1.7 Défis en matière de sélection des caractéristiques

La sélection des caractéristiques n'est pas sans poser de problèmes. Les problèmes les plus courants sont les suivants : - **Dimensionnalité élevée :** Dans des domaines tels que la génomique et l'exploration de textes, le nombre de caractéristiques peut être extrêmement élevé, ce qui complique la sélection des caractéristiques. - **Interactions des caractéristiques :** Certaines caractéristiques peuvent être sans intérêt individuellement, mais importantes lorsqu'elles sont combinées à d'autres caractéristiques. L'identification de ces interactions peut s'avérer complexe. - **Connaissance du domaine :** L'intégration de l'expertise du domaine dans le processus de sélection des caractéristiques peut être difficile, mais elle est souvent nécessaire pour obtenir des résultats optimaux. - **Surajustement :** Alors que la sélection des caractéristiques vise à réduire le surajustement, des méthodes inappropriées peuvent conduire à un surajustement des données d'apprentissage. - **Complexité informatique :** Les méthodes d'enrobage, en particulier, peuvent être coûteuses en temps

et en argent. - **Qualité des données :** Des données de mauvaise qualité peuvent induire en erreur les méthodes de sélection des caractéristiques, ce qui se traduit par des modèles sous-optimaux.

1.8 Conclusion

La sélection des caractéristiques est une étape importante du processus d'apprentissage automatique, car elle a un impact significatif sur les performances et l'interprétabilité des modèles. En sélectionnant soigneusement les caractéristiques les plus pertinentes, nous pouvons construire des modèles qui sont non seulement précis, mais aussi efficaces et plus faciles à comprendre. Que l'on utilise le filtre, le wrapper ou les méthodes intégrées, il est essentiel de prendre en compte le contexte et les caractéristiques spécifiques des données pour choisir la technique de sélection des caractéristiques la plus appropriée.

La sélection des caractéristiques est le processus qui consiste à choisir un sous-ensemble de caractéristiques pertinentes (variables, prédicteurs) à partir d'un ensemble de données, qui sont les plus importantes pour la construction d'un modèle. Cela permet d'améliorer les performances du modèle en accélérant la formation et en rendant le modèle plus facile à interpréter.

1.9 Pourquoi la sélection des caractéristiques ?

La sélection des caractéristiques est importante car elle améliore l'efficacité d'un modèle d'apprentissage automatique de plusieurs façons : 1. **Réduire la complexité temporelle et spatiale** La sélection des caractéristiques permet de minimiser la quantité de données que le modèle doit traiter, ce qui accélère l'apprentissage et réduit l'utilisation de la mémoire.

Exemple : Imaginez que vous essayez de prédire si un courrier électronique est du spam en utilisant 1 000 caractéristiques différentes. En ne sélectionnant que les 20 caractéristiques les plus importantes, comme des mots ou des phrases spécifiques, le modèle s'exécute plus rapidement et nécessite moins de mémoire

2. **Garbage In, Garbage Out** L'inclusion de caractéristiques non pertinentes peut entraîner de mauvaises performances du modèle, car la qualité des données d'entrée a une incidence directe sur les résultats. Exemple : Si vous ajoutez des données non pertinentes, comme la couleur préférée de l'expéditeur d'un courrier électronique, à un modèle de détection du spam, celui-ci risque d'être peu performant. Tout comme l'ajout de mauvais ingrédients dans

une recette donne un mauvais plat, des données non pertinentes conduisent à de mauvaises prédictions.

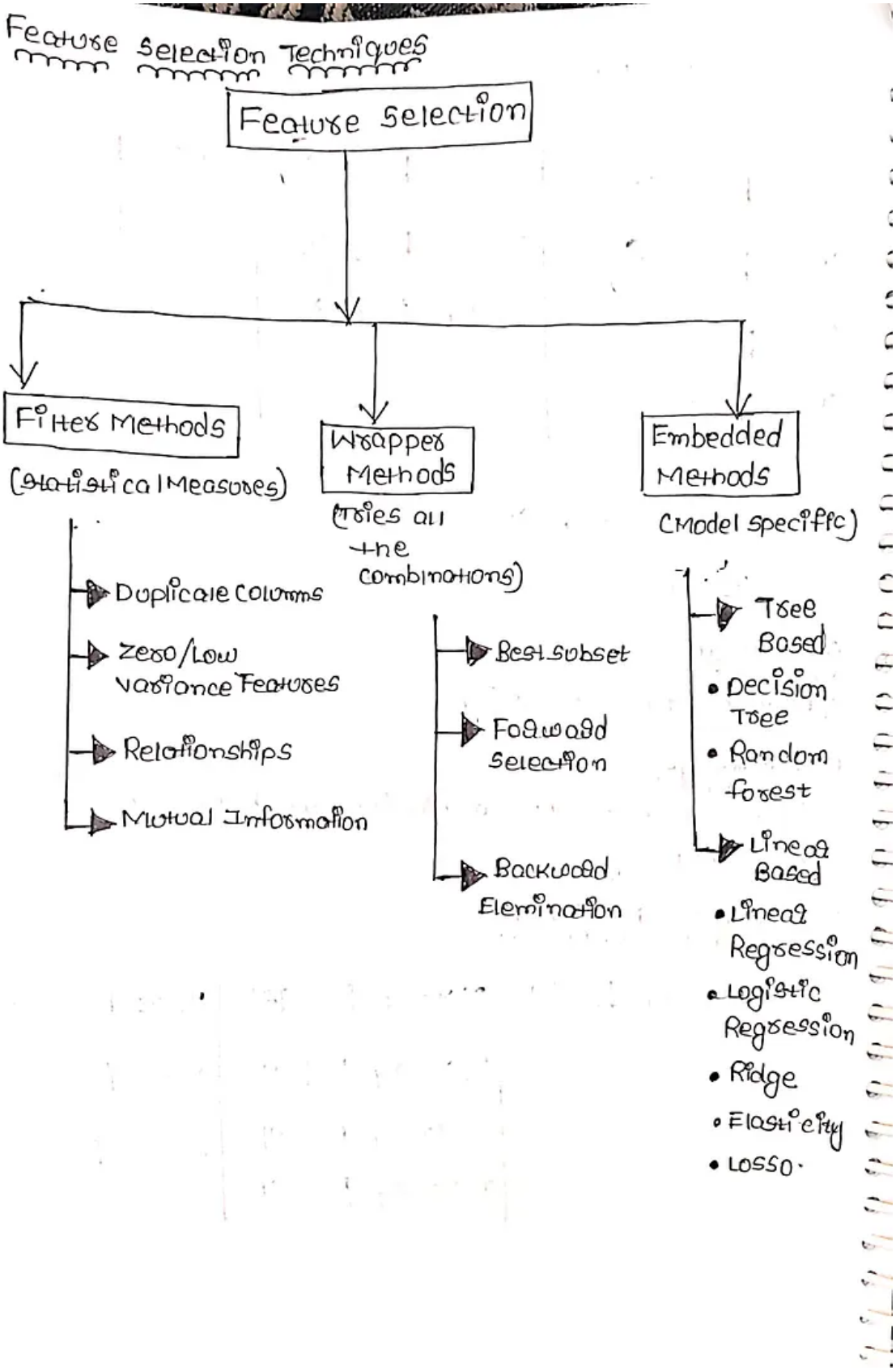
3. **Loi de parcimonie** Selon la loi de parcimonie, les modèles plus simples comportant moins de caractéristiques sont souvent plus efficaces et plus faciles à comprendre. Exemple : Lorsqu'il s'agit de prédire si un courriel est du spam, l'utilisation de quelques caractéristiques clés, telles que l'objet et des mots clés spécifiques, peut conduire à un modèle plus simple et plus précis. Plus le modèle est simple, plus il est performant.

4. **Caractéristiques non pertinentes** Certaines caractéristiques peuvent ne rien apporter de significatif aux performances du modèle, ce qui rend leur suppression indispensable. Exemple : La couleur de l'arrière-plan d'un courriel n'a aucune incidence sur la prédiction du spam. La suppression de ces caractéristiques non pertinentes permet au modèle de se concentrer sur ce qui importe réellement

5. **Dégradation des performances** L'inclusion de caractéristiques inutiles peut en fait réduire la précision et l'efficacité du modèle.

Exemple : L'ajout de caractéristiques inutiles, telles que le style de police de l'e-mail, risque d'embrouiller le modèle de détection du spam et de le rendre moins précis. Trop de données non pertinentes peuvent nuire aux performances.

6. **Multicollinéarité** Des caractéristiques fortement corrélées peuvent entraîner une multicollinéarité, ce qui rend les coefficients du modèle instables et difficiles à interpréter. Exemple : Si vous essayez de déterminer la valeur d'une maison, le fait d'avoir deux caractéristiques comme la "superficie" et le "nombre de pièces" qui sont étroitement liées peut perturber le modèle. Ce phénomène est similaire à la manière dont la multicolinéarité, causée par des caractéristiques fortement corrélées, peut déstabiliser les prédictions d'un modèle.


Feature Selection Techniques
Feature Selection
Filter Methods
(Statistical Measures)
Duplicate columns
Zero/Low variance Features
Relationships
Mutual Information
Wrapper Methods
(Tries all the combinations)
Best subset
Forward selection
Backward Elemination
Embedded Methods
(Model specific)
Tree Based
• Decision Tree
• Random forest
Linear Based
• Linear Regression
• Logistic Regression
• Ridge
• Elasticity
• Lasso.

Méthodes de filtrage : ces techniques évaluent l'importance des caractéristiques sur la base de leurs propriétés statistiques et sélectionnent les plus pertinentes avant d'entraîner le modèle. Elles utilisent des mesures telles que la corrélation, les tests du chi-carré ou l'information mutuelle pour classer les caractéristiques et supprimer celles qui n'atteignent pas un certain seuil. Les méthodes de filtrage sont rapides et indépendantes de l'algorithme d'apprentissage automatique utilisé. Colonnes en double : Dans les méthodes de filtrage, les colonnes en double font référence aux caractéristiques qui contiennent des informations identiques ou très similaires. Ces doublons n'apportent aucune valeur ajoutée et peuvent être supprimés pour simplifier l'ensemble de données et améliorer les performances du modèle.

[1]:
```
# Import required libraries
import pandas as pd

# Create a DataFrame with duplicate columns
df = pd.DataFrame({
    "Gender": ["M", "F", "M", "F", "M", "M", "F", "F", "M", "F"],
    "Experience": [2, 3, 5, 6, 7, 8, 9, 5, 4, 3],
    "gender": ["M", "F", "M", "F", "M", "M", "F", "F", "M", "F"],  # Duplicate␣
 ↪of "Gender"
    "exp": [2, 3, 5, 6, 7, 8, 9, 5, 4, 3],  # Duplicate of "Experience"
    "Salary": [25000, 30000, 40000, 45000, 50000, 65000, 80000, 40000, 35000,␣
 ↪30000]
})

# Display the original DataFrame
print("Original DataFrame:")
print(df)
print("*"*60)

# Identify duplicate columns
duplicate_columns = df.columns[df.T.duplicated()]
print(duplicate_columns)
print("*"*60)

# Drop the duplicate columns
columns=df.T[df.T.duplicated()].T
df.drop(columns,axis=1,inplace=True)

# Display the DataFrame after removing duplicate columns
print("\nDataFrame after removing duplicate columns:")
print(df)
```

```
Original DataFrame:
  Gender  Experience gender  exp  Salary
0      M           2      M    2   25000
1      F           3      F    3   30000
2      M           5      M    5   40000
3      F           6      F    6   45000
```

```
4      M           7      M    7   50000
5      M           8      M    8   65000
6      F           9      F    9   80000
7      F           5      F    5   40000
8      M           4      M    4   35000
9      F           3      F    3   30000
************************************************************
Index(['gender', 'exp'], dtype='object')
************************************************************

DataFrame after removing duplicate columns:
  Gender  Experience  Salary
0      M           2   25000
1      F           3   30000
2      M           5   40000
3      F           6   45000
4      M           7   50000
5      M           8   65000
6      F           9   80000
7      F           5   40000
8      M           4   35000
9      F           3   30000
```

2. **Caractéristiques à variance nulle ou faible** : Les caractéristiques qui varient peu ou pas du tout dans les données et qui n'aident pas à distinguer les résultats peuvent être supprimées pour simplifier le modèle.

[3]:
```python
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import VarianceThreshold
import pandas as pd

# Load the Breast Cancer dataset
X, y = load_breast_cancer(return_X_y=True, as_frame=True)

# Display the shape of the original feature set
print("Original Shape: ", X.shape)
print("*" * 60)

# Initialize VarianceThreshold with a threshold of 0.03
# This will filter out features with variance below 0.03
vth = VarianceThreshold(threshold=0.03)

# Fit the VarianceThreshold and transform the data
# This removes features with low variance
X_filtered = vth.fit_transform(X)

# Display the shape of the filtered feature set
print("Filtered Shape: ", X_filtered.shape)
```

```python
# Display feature names after filtering
# Note: VarianceThreshold does not have get_feature_names_out() method
# The following line may not work and could be omitted or replaced with manual␣
 ↪feature names
print(vth.get_feature_names_out())
print("*" * 60)

# Create a DataFrame with the filtered features
# Feature names might not be available, so using generic names instead
X_filtered_df = pd.DataFrame(X_filtered, columns=vth.get_feature_names_out())

# Display the filtered DataFrame
print(X_filtered_df)
```

```
Original Shape:  (569, 30)
************************************************************
Filtered Shape:  (569, 13)
['mean radius' 'mean texture' 'mean perimeter' 'mean area' 'radius error'
 'texture error' 'perimeter error' 'area error' 'worst radius'
 'worst texture' 'worst perimeter' 'worst area' 'worst concavity']
************************************************************
     mean radius  mean texture  mean perimeter  mean area  radius error
0          17.99         10.38          122.80     1001.0        1.0950  \
1          20.57         17.77          132.90     1326.0        0.5435
2          19.69         21.25          130.00     1203.0        0.7456
3          11.42         20.38           77.58      386.1        0.4956
4          20.29         14.34          135.10     1297.0        0.7572
..           …             …               …          …             …
564        21.56         22.39          142.00     1479.0        1.1760
565        20.13         28.25          131.20     1261.0        0.7655
566        16.60         28.08          108.30      858.1        0.4564
567        20.60         29.33          140.10     1265.0        0.7260
568         7.76         24.54           47.92      181.0        0.3857

     texture error  perimeter error  area error  worst radius  worst texture
0           0.9053            8.589      153.40        25.380          17.33  \
1           0.7339            3.398       74.08        24.990          23.41
2           0.7869            4.585       94.03        23.570          25.53
3           1.1560            3.445       27.23        14.910          26.50
4           0.7813            5.438       94.44        22.540          16.67
..             …                …           …             …              …
564         1.2560            7.673      158.70        25.450          26.40
565         2.4630            5.203       99.04        23.690          38.25
566         1.0750            3.425       48.55        18.980          34.12
567         1.5950            5.772       86.22        25.740          39.42
```

```
568         1.4280            2.548       19.15          9.456          30.37

     worst perimeter  worst area  worst concavity
0              184.60      2019.0           0.7119
1              158.80      1956.0           0.2416
2              152.50      1709.0           0.4504
3               98.87       567.7           0.6869
4              152.20      1575.0           0.4000
..                ...         ...              ...
564            166.10      2027.0           0.4107
565            155.00      1731.0           0.3215
566            126.70      1124.0           0.3403
567            184.60      1821.0           0.9387
568             59.16       268.6           0.0000

[569 rows x 13 columns]
```

3 . **Relations** : les relations dans les méthodes de filtrage permettent d'identifier les caractéristiques les plus pertinentes en fonction de leur relation avec la variable cible. Différentes méthodes sont utilisées en fonction du type de caractéristiques impliquées (catégorielles ou numériques).

1. **Categorical vs. Categorical: Chi-Square Test**

Définition : Le test du chi carré mesure l'association entre deux variables catégorielles en comparant les fréquences observées et attendues dans un tableau de contingence. Il évalue si la distribution des variables catégorielles diffère de ce que l'on pourrait attendre du hasard.

Quand l'utiliser : Utilisez le test du chi carré lorsque la caractéristique et la variable cible sont toutes deux catégoriques.

1.10 Python Implementation :

[4]:
```python
import pandas as pd
from scipy.stats import chi2_contingency

# Sample dataset
data = {
    'Gender': ['Male', 'Female', 'Female', 'Male', 'Female', 'Male', 'Male',␣
 ↪'Female', 'Female', 'Male'],
    'Marital_Status': ['Married', 'Single', 'Married', 'Single', 'Married',␣
 ↪'Single', 'Married', 'Single', 'Single', 'Married'],
    'Purchased': ['No', 'Yes', 'Yes', 'No', 'Yes', 'No', 'Yes', 'No', 'Yes',␣
 ↪'No']
}

df = pd.DataFrame(data)
print("Sample Data:")
print(df)
```

```
# Create crosstab between Gender and Purchased
crosstab = pd.crosstab(df['Gender'], df['Purchased'])
print("\nCrosstab (Contingency Table):")
print(crosstab)

# Perform Chi-Square test
chi2, p, dof, expected = chi2_contingency(crosstab)

print("\nChi-Square Test Results:")
print(f"Chi-Square Statistic: {chi2}")
print(f"P-value: {p}")
print(f"Degrees of Freedom: {dof}")
print("Expected Frequencies:")
print(expected)

# Interpretation
if p < 0.05:
    print("\nConclusion: There is a significant relationship between Gender and␣
 ↪Purchased.")
else:
    print("\nConclusion: There is no significant relationship between Gender␣
 ↪and Purchased.")
```

```
Sample Data:
   Gender Marital_Status Purchased
0    Male        Married        No
1  Female         Single       Yes
2  Female        Married       Yes
3    Male         Single        No
4  Female        Married       Yes
5    Male         Single        No
6    Male        Married       Yes
7  Female         Single        No
8  Female         Single       Yes
9    Male        Married        No

Crosstab (Contingency Table):
Purchased  No  Yes
Gender
Female      1    4
Male        4    1

Chi-Square Test Results:
Chi-Square Statistic: 1.6
P-value: 0.20590321073206466
Degrees of Freedom: 1
Expected Frequencies:
```

```
[[2.5 2.5]
 [2.5 2.5]]
```

```
Conclusion: There is no significant relationship between Gender and Purchased.
```

2.**Categorical vs. Numerical: ANOVA (Analysis of Variance)**

Définition : L'ANOVA teste si les moyennes des différents groupes (catégories) pour une caractéristique numérique sont significativement différentes. Elle permet de déterminer si une caractéristique catégorielle influence de manière significative une variable cible numérique.

Quand l'utiliser : Utilisez l'ANOVA lorsque la caractéristique est catégorique et que la variable cible est numérique.

[5]:
```
import pandas as pd
from scipy.stats import f_oneway

# Sample dataset
data = {
    'Education_Level': ['High School', 'Bachelor', 'Master', 'High School',␣
 ↪'Bachelor', 'Master', 'Bachelor', 'Master', 'High School'],
    'Income': [40000, 55000, 70000, 42000, 58000, 72000, 60000, 75000, 41000]
}

df = pd.DataFrame(data)
print("Sample Data:")
print(df)

# Group the data by 'Education_Level'
groups = df.groupby('Education_Level')['Income'].apply(list)

# Perform ANOVA
f_statistic, p_value = f_oneway(*groups)

print("\nANOVA Test Results:")
print(f"F-Statistic: {f_statistic}")
print(f"P-value: {p_value}")

# Interpretation
if p_value < 0.05:
    print("\nConclusion: There is a significant difference in Income between␣
 ↪different Education Levels.")
else:
    print("\nConclusion: There is no significant difference in Income between␣
 ↪different Education Levels.")
```

```
Sample Data:
  Education_Level  Income
0     High School   40000
```

```
1        Bachelor   55000
2          Master   70000
3     High School   42000
4        Bachelor   58000
5          Master   72000
6        Bachelor   60000
7          Master   75000
8     High School   41000

ANOVA Test Results:
F-Statistic: 161.85365853658587
P-value: 6.0265393274654385e-06

Conclusion: There is a significant difference in Income between different
Education Levels.
```

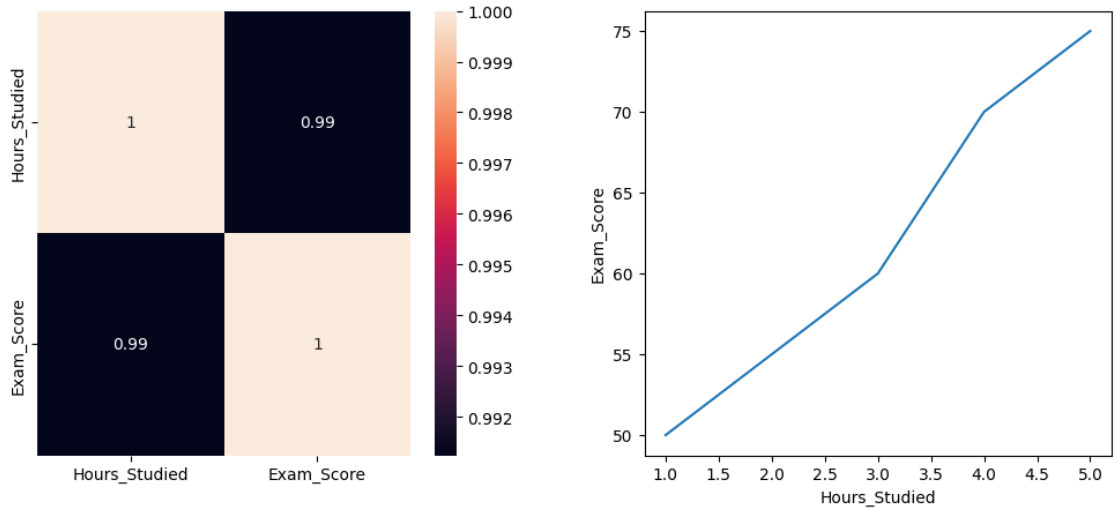
3. **Numerical vs. Numerical: Pearson Correlation**

Définition : La corrélation de Pearson mesure la relation linéaire entre deux variables numériques. Elle va de -1 (relation linéaire négative parfaite) à +1 (relation linéaire positive parfaite).

Quand l'utiliser : Utilisez la corrélation de Pearson lorsque la caractéristique et la variable cible sont toutes deux numériques

[6]:
```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Sample data
data = {'Hours_Studied': [1, 2, 3, 4, 5],
        'Exam_Score': [50, 55, 60, 70, 75]}
df = pd.DataFrame(data)
fig, axes = plt.subplots(1, 2, figsize=(12,5))
plt.subplots_adjust(hspace=0.3, wspace=0.3)

# Pearson Correlation
correlation = df.corr(method="pearson")
sns.heatmap(correlation,annot=True,ax=axes[0])
sns.lineplot(x=df["Hours_Studied"],y=df["Exam_Score"],ax=axes[1])
```

[6]:
```
<Axes: xlabel='Hours_Studied', ylabel='Exam_Score'>
```



2 4 .Mutual Information

Définition : L'information mutuelle (IM) mesure la quantité d'informations qu'une variable fournit sur une autre. Elle est non linéaire et non paramétrique, ce qui signifie qu'elle peut saisir n'importe quel type de dépendance entre les variables, et pas seulement les relations linéaires. MI élevé : indique que la connaissance de la valeur d'une variable donne des informations significatives sur l'autre.

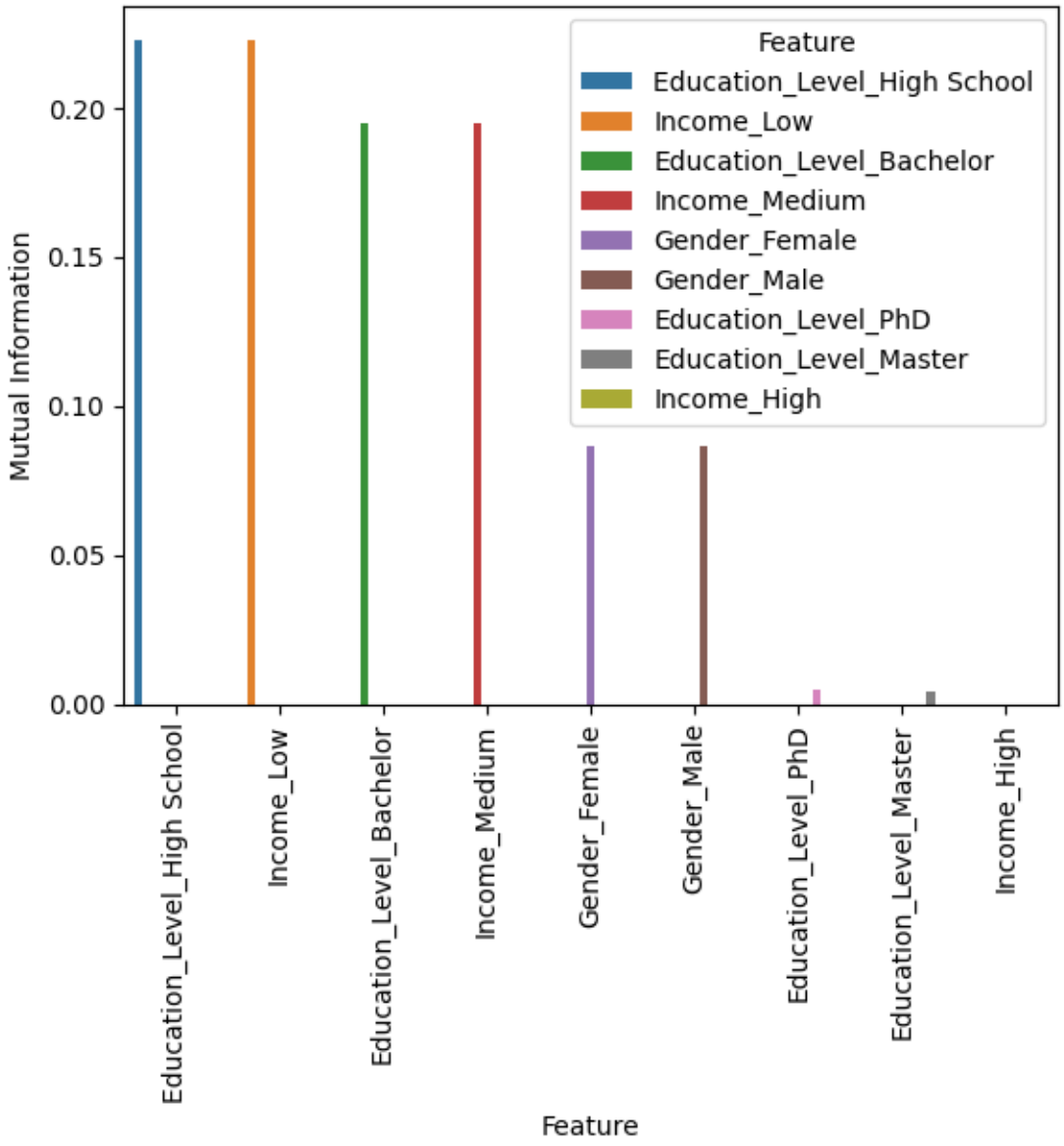
MI faible : indique que la connaissance de la valeur d'une variable ne fournit que peu d'informations sur l'autre.

[10]:
```
import pandas as pd
from sklearn.feature_selection import mutual_info_classif
from sklearn.preprocessing import OneHotEncoder

# Sample dataset
data = {
    'Gender': ['Male', 'Female', 'Female', 'Male', 'Female', 'Male', 'Male',␣
 ↪'Female', 'Female', 'Male'],
    'Education_Level': ['High School', 'Bachelor', 'Master', 'PhD', 'High␣
 ↪School', 'Bachelor', 'Master', 'PhD', 'Bachelor', 'Master'],
    'Income': ['Low', 'Medium', 'High', 'High', 'Low', 'Medium', 'High',␣
 ↪'High', 'Medium', 'High'],
    'Target': [0, 1, 1, 0, 0, 1, 1, 1, 1, 0]
}

df = pd.DataFrame(data)
print("Sample Data:")
print(df)
```

```
# OneHotEncode categorical features
encoder = OneHotEncoder(sparse_output=False)
encoded_features = encoder.fit_transform(df[['Gender', 'Education_Level',␣
 ↪'Income']])

# Convert the encoded features into a DataFrame
encoded_df = pd.DataFrame(encoded_features, columns=encoder.
 ↪get_feature_names_out())

# Calculate mutual information
mi_scores = mutual_info_classif(encoded_df, df['Target'],␣
 ↪discrete_features=True)

# Create a DataFrame to display the MI scores
mi_df = pd.DataFrame({'Feature': encoded_df.columns, 'Mutual Information':␣
 ↪mi_scores})
mi_df = mi_df.sort_values(by='Mutual Information', ascending=False)
print("\nMutual Information Scores:")
```

```
Sample Data:
   Gender Education_Level  Income  Target
0    Male     High School     Low       0
1  Female        Bachelor  Medium       1
2  Female          Master    High       1
3    Male             PhD    High       0
4  Female     High School     Low       0
5    Male        Bachelor  Medium       1
6    Male          Master    High       1
7  Female             PhD    High       1
8  Female        Bachelor  Medium       1
9    Male          Master    High       0

Mutual Information Scores:
```

[12]:
```
sns.barplot(x=mi_df["Feature"],y=mi_df["Mutual␣
 ↪Information"],hue=mi_df["Feature"])
plt.xticks(rotation=90)
plt.show()
```



[9]:
```
print(mi_df)
```

```
                     Feature  Mutual Information
3  Education_Level_High School            0.223144
7                   Income_Low            0.223144
2    Education_Level_Bachelor            0.194976
8                Income_Medium            0.194976
0                Gender_Female            0.086305
1                  Gender_Male            0.086305
5          Education_Level_PhD            0.005132
4       Education_Level_Master            0.004022
```

```
6                  Income_High            0.000000
```

3 5. SelectKBest :

SelectKBest est une méthode de sélection de caractéristiques dans scikit-learn qui sélectionne les k meilleures caractéristiques sur la base d'une fonction de notation. Elle fait partie de la famille des méthodes de filtrage, ce qui signifie qu'elle évalue chaque caractéristique indépendamment du modèle.

Comment ça marche 1 - Choisir une fonction de notation :SelectKBest nécessite une fonction de notation pour classer les caractéristiques. Les options courantes sont les suivantes :

- f_classif: ANOVA F-value between label/feature for classification tasks.
- chi2: Chi-square test for independence.
- mutual_info_classif: Mutual information for classification

2. Sélection des meilleures caractéristiques : Sur la base de la fonction de notation, il classe toutes les caractéristiques et sélectionne les k meilleures caractéristiques.

[14]:
```python
import pandas as pd
import numpy as np
from sklearn.datasets import load_iris
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# 1. Load Data
data = load_iris()
X = data.data
y = data.target

# Convert to DataFrame for better visualization
df = pd.DataFrame(X, columns=data.feature_names)
df['target'] = y


# 2. SelectKBest
# We use the ANOVA F-value as the scoring function for classification tasks.
selector = SelectKBest(score_func=f_classif, k=3)  # Select the top 3features
X_new = selector.fit_transform(X, y)

# Display the scores and selected features
scores = selector.scores_
selected_features = np.array(data.feature_names)[selector.get_support()]
print("\nFeature Scores:\n", scores)
print("\nSelected Features:\n", selected_features)

# 3. Train-Test Split
```

```
X_train, X_test, y_train, y_test = train_test_split(X_new, y, test_size=0.2,␣
 ↪random_state=42)

# 4. Train a Model
model = RandomForestClassifier()
model.fit(X_train, y_train)

# 5. Evaluate the Model
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("\nModel Accuracy with Selected Features:", accuracy)
```

```
Feature Scores:
 [ 119.26450218   49.16004009 1180.16118225  960.0071468 ]

Selected Features:
 ['sepal length (cm)' 'petal length (cm)' 'petal width (cm)']

Model Accuracy with Selected Features: 1.0
```

4 Avantages des méthodes de filtrage

1. Simplicité et rapidité : Explication : Les méthodes de filtrage sont faciles à mettre en œuvre et efficaces sur le plan informatique, car elles utilisent des tests statistiques sans impliquer un apprentissage complexe du modèle.
2. Indépendance du modèle : explication : Ils évaluent les caractéristiques sur la base de mesures statistiques uniquement, ce qui les rend applicables à différents modèles d'apprentissage automatique.
3. Risque réduit de surajustement : Explication : Parce qu'elles ne s'appuient pas sur les performances spécifiques d'un modèle, les méthodes de filtrage sont moins susceptibles de s'adapter de manière excessive aux données d'apprentissage.

5 Inconvénients des méthodes de filtrage

1. Prise en compte limitée des interactions :

- Explication : Les méthodes de filtrage évaluent la pertinence des caractéristiques individuellement, ce qui signifie qu'elles ne vérifient que la relation d'une caractéristique avec la variable cible à la fois. Cette approche peut passer à côté d'interactions complexes entre les caractéristiques et peut prendre beaucoup de temps pour les ensembles de données très complexes ou comportant de nombreuses caractéristiques.

2. Sélection possible de caractéristiques redondantes :

- Explication : Les méthodes de filtrage peuvent choisir des caractéristiques qui sont individuellement pertinentes mais redondantes ou fortement corrélées entre elles, ce qui peut entraîner

des inefficacités dans le modèle.

6 Méthodes d'enveloppement

Les méthodes d'enveloppement dans la sélection des caractéristiques impliquent l'utilisation d'un modèle d'apprentissage automatique pour évaluer les performances de différents sous-ensembles de caractéristiques. Elles "enveloppent" le processus de sélection des caractéristiques autour du modèle, en évaluant les performances de différentes combinaisons de caractéristiques en entraînant et en évaluant le modèle plusieurs fois.

1. Meilleur sous-ensemble : la sélection du meilleur sous-ensemble passe en revue toutes les combinaisons de caractéristiques possibles pour trouver celle qui donne les meilleurs résultats pour le modèle.

[16]:
```
!pip install mlxtend
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting mlxtend
  Downloading mlxtend-0.23.1-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in ./.local/lib/python3.10/site-
packages (from mlxtend) (1.11.3)
Requirement already satisfied: numpy>=1.16.2 in ./.local/lib/python3.10/site-
packages (from mlxtend) (1.23.5)
Requirement already satisfied: pandas>=0.24.2 in ./.local/lib/python3.10/site-
packages (from mlxtend) (2.0.1)
Requirement already satisfied: scikit-learn>=1.0.2 in
./.local/lib/python3.10/site-packages (from mlxtend) (1.2.2)
Requirement already satisfied: matplotlib>=3.0.0 in
./.local/lib/python3.10/site-packages (from mlxtend) (3.7.1)
Requirement already satisfied: joblib>=0.13.2 in ./.local/lib/python3.10/site-
packages (from mlxtend) (1.1.1)
Requirement already satisfied: contourpy>=1.0.1 in ./.local/lib/python3.10/site-
packages (from matplotlib>=3.0.0->mlxtend) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/lib/python3/dist-packages
(from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in /usr/lib/python3/dist-
packages (from matplotlib>=3.0.0->mlxtend) (4.29.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/lib/python3/dist-
packages (from matplotlib>=3.0.0->mlxtend) (1.3.2)
Requirement already satisfied: packaging>=20.0 in ./.local/lib/python3.10/site-
packages (from matplotlib>=3.0.0->mlxtend) (23.1)
Requirement already satisfied: pillow>=6.2.0 in ./.local/lib/python3.10/site-
packages (from matplotlib>=3.0.0->mlxtend) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/lib/python3/dist-
packages (from matplotlib>=3.0.0->mlxtend) (2.4.7)
Requirement already satisfied: python-dateutil>=2.7 in
./.local/lib/python3.10/site-packages (from matplotlib>=3.0.0->mlxtend)
(2.9.0.post0)
```

```
Requirement already satisfied: pytz>=2020.1 in ./.local/lib/python3.10/site-
packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in ./.local/lib/python3.10/site-
packages (from pandas>=0.24.2->mlxtend) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in
./.local/lib/python3.10/site-packages (from scikit-learn>=1.0.2->mlxtend)
(3.5.0)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from
python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
Downloading mlxtend-0.23.1-py3-none-any.whl (1.4 MB)

1.4/1.4 MB 36.1 MB/s eta 0:00:00
Installing collected packages: mlxtend
Successfully installed mlxtend-0.23.1
```

[17]:
```python
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from mlxtend.feature_selection import ExhaustiveFeatureSelector
from sklearn.metrics import accuracy_score

# Load the iris dataset
data = load_iris()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

# Initialize the model
model = LogisticRegression(max_iter=200)

# Initialize the Exhaustive Feature Selector
efs = ExhaustiveFeatureSelector(
    estimator=model,
    min_features=1,
    max_features=3,
    scoring='accuracy',
    cv=5,
    n_jobs=-1
)

# Perform the feature selection
efs = efs.fit(X_train, y_train)
```

```
# Get the best feature subset and its score
best_features = efs.best_feature_names_
best_score = efs.best_score_

print("Best Feature Subset:", best_features)
print("Best Accuracy Score:", best_score)
```

```
Features: 14/14
```

```
Best Feature Subset: ('petal length (cm)', 'petal width (cm)')
Best Accuracy Score: 0.9619047619047618
```

7 Advantages and Disadvantages of Best Subset :

Avantages : sélection du meilleur modèle : garantit que le sous-ensemble de caractéristiques le plus efficace est choisi, ce qui peut conduire à la meilleure performance possible du modèle.

Inconvénients : complexité temporelle élevée : Le processus peut être coûteux en temps de calcul, en particulier avec un grand nombre de caractéristiques, car il évalue tous les sous-ensembles possibles de caractéristiques.

2. **Forward Selection** : Forward Selection commence avec aucune caractéristique et ajoute itérativement des caractéristiques au modèle. À chaque étape, elle ajoute la caractéristique qui améliore le plus les performances du modèle. Ce processus se poursuit jusqu'à ce que l'ajout de nouvelles fonctionnalités n'améliore plus le modèle.

Étapes de la sélection des caractéristiques en amont 1. Entraîner n modèles en utilisant chaque caractéristique (n) individuellement et vérifier la performance 2. Choisir la variable qui donne la meilleure performance 3. Répéter le processus en ajoutant une variable à la fois 4. La variable produisant la plus forte amélioration est conservée 5. Répéter l'ensemble du processus jusqu'à ce qu'il n'y ait plus d'amélioration significative de la performance du modèle.

3. **Backward Elimination**: L'élimination à rebours commence par toutes les caractéristiques et supprime de manière itérative la caractéristique la moins significative. À chaque étape, elle supprime la caractéristique qui a l'impact le plus faible sur les performances du modèle. Ce processus se poursuit jusqu'à ce que l'élimination d'autres caractéristiques dégrade les performances du modèle.

Étapes de l'élimination à rebours Former un modèle initial : 1. Commencez par former un modèle en utilisant toutes les caractéristiques disponibles. 2. Évaluez l'importance des caractéristiques : Évaluer l'importance de chaque caractéristique à l'aide de mesures de performance (par exemple, précision, score F1). 3. Identifier la caractéristique la moins importante : déterminer la caractéristique qui a le moins d'impact sur les performances du modèle : Déterminez la caractéristique qui a le moins d'impact sur les performances du modèle. 4. Supprimez la caractéristique la moins importante : Retirez cette caractéristique de l'ensemble de données et entraînez à nouveau le modèle. 5. Répétez le processus : Continuez à supprimer la caractéristique la moins importante, une à la fois, et entraînez à nouveau le modèle jusqu'à ce que la suppression d'autres caractéristiques n'entraîne aucune amélioration significative ou aggrave les performances du modèle.

[20]:
```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SequentialFeatureSelector
from sklearn.metrics import mean_squared_error

# Sample data
data = pd.DataFrame({
    'X1': [1, 2, 3, 4, 5],
    'X2': [2, 3, 4, 5, 6],
    'X3': [5, 6, 7, 8, 9],
    'Y': [1, 2, 1, 2, 1]
})

# Splitting the data
X = data[['X1', 'X2', 'X3']]
y = data['Y']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)

# Forward Selection
forward_selector = SequentialFeatureSelector(
    estimator=LinearRegression(),
    n_features_to_select='auto',  # Can specify number of features or use 'auto'
    direction='forward',
    scoring='neg_mean_squared_error',
    cv=4  # Cross-validation
)

# Fit the model
forward_selector.fit(X_train, y_train)
print("Forward Selection - Selected features:", X_train.
 ↪columns[forward_selector.get_support()].tolist())

# Backward Elimination
backward_selector = SequentialFeatureSelector(
    estimator=LinearRegression(),
    n_features_to_select='auto',  # Can specify number of features or use 'auto'
    direction='backward',
    scoring='neg_mean_squared_error',
    cv=4  # Cross-validation
)

# Fit the model
backward_selector.fit(X_train, y_train)
print("Backward Elimination - Selected features:", X_train.
 ↪columns[backward_selector.get_support()].tolist())
```

```
Forward Selection - Selected features: ['X3']
Backward Elimination - Selected features: ['X1', 'X2']
```

Avantages et inconvénients de la sélection en amont et de l'élimination en aval - **Avantage :** Réduit la complexité temporelle par rapport à la sélection du meilleur sous-ensemble en ajoutant progressivement des caractéristiques, ce qui évite d'avoir à évaluer toutes les combinaisons possibles. - **Inconvénient :** Certaines combinaisons qui pourraient potentiellement produire la meilleure mesure, telle que la précision ou le score F1, peuvent ne pas être prises en compte parce qu'elles ne considèrent qu'une seule caractéristique à la fois.

Embedded Methods

Les méthodes intégrées de sélection des caractéristiques intègrent la sélection des caractéristiques dans le processus d'apprentissage du modèle. Contrairement aux méthodes de filtrage, qui évaluent les caractéristiques indépendamment du modèle, et aux méthodes d'enveloppement, qui évaluent des sous-ensembles de caractéristiques sur la base des performances du modèle, les méthodes intégrées effectuent la sélection des caractéristiques dans le cadre du processus d'apprentissage du modèle. Il en résulte souvent un processus de sélection des caractéristiques plus efficace.

Decision Trees and Random Forests: - Concept: Tree-based models like Decision Trees and Random Forests provide feature importance scores based on how much each feature contributes to reducing impurity (e.g., Gini impurity or entropy for classification, variance for regression) - Implementation: Feature importances can be extracted from tree-based models to select the most important features.

[21]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
import pandas as pd

# Load data
X, y = load_iris(return_X_y=True)

# Apply Random Forest
rf = RandomForestClassifier()
rf.fit(X, y)

# Get feature importances
feature_importances = rf.feature_importances_
feature_names = load_iris().feature_names
importance_df = pd.DataFrame({'Feature': feature_names, 'Importance':␣
 ↪feature_importances})
print(importance_df.sort_values(by='Importance', ascending=False))
```

```
             Feature  Importance
2  petal length (cm)    0.426531
3   petal width (cm)    0.424178
0  sepal length (cm)    0.122388
1   sepal width (cm)    0.026903
```

Advantages

2. Computational Efficiency: Often more efficient than exhaustive methods, as they do not evaluate all possible feature combinations.

Difference between Filter ,Wrapper and Embedded Methods

Filter Methods	Wrapper Methods	Embedded Methods
These methods select features based on their statistical relationship with the target variable, independent of any machine learning algorithm	These methods evaluate feature subsets by training and testing a specific machine learning model to find the best-performing combination.	These methods perform feature selection as part of the model training process, usually through regularization techniques.
How It Works: It uses statistical techniques to filter out less relevant features before model training.	**How It Works**: It tries different combinations of features, trains the model on each, and picks the combination that gives the best model performance.	**How It Works**: The model itself decides which features are most important during the training process, often by penalizing less important ones
Advantages: **1. Fast and Simple**: Quick and easy to apply, as it doesn't involve any model training. **2. Scalability**: Suitable for large datasets with many features.	**Advantages**: **1.High Accuracy**: Since it evaluates features based on actual model performance, it often yields more accurate results. **2. Customizable**: Tailored to the specific algorithm you're using.	**Advantages**: **1. Efficiency**: Combines the benefits of both Filter and Wrapper methods, offering a balance of speed and accuracy. **2. In-built Feature Selection**: Automatically selects features while building the model, saving time.
Disadvantages: **1. Less Accurate**: Since it doesn't consider the specific machine learning algorithm, it might miss important feature interactions. **2.Independence Assumption**: Often assumes features are independent, which might not hold true in all cases.	**Disadvantages**: **1. Time-Consuming**: Involves training multiple models, which can be computationally expensive and slow, especially for large datasets. **2. Risk of Overfitting**: Due to multiple model evaluations, there's a higher chance of overfitting.	**Disadvantages**: **1.Algorithm-Specific**: The feature selection process is tied to the specific model, so it might not generalize well across different models. **2. Complexity**: Can be harder to interpret and requires a good understanding of the underlying algorithm.
Examples: Correlation, Chi-Square test, ANOVA, Information gain, etc.	**Examples**: Forward Selection, Backward Elimination, Best Subset etc.	**Examples**: LASSO (Least Absolute Shrinkage and Selection Operator), Elastic Net, Ridge Regression, etc.

Conclusion Dans ce blog, nous avons exploré différentes méthodes de sélection des caractéristiques, y compris les méthodes de filtrage et les méthodes d'enveloppement, en soulignant leurs avantages et inconvénients uniques. Les méthodes de filtrage, telles que le chi carré et l'ANOVA, offrent une efficacité de calcul mais peuvent négliger les interactions entre les caractéristiques. Les méthodes d'enveloppement telles que la sélection en amont et l'élimination en aval fournissent une évaluation plus détaillée, mais peuvent être très gourmandes en ressources informatiques. En implémentant ces méthodes en Python, nous avons démontré des approches pratiques pour optimiser la sélection des caractéristiques, en améliorant la performance et l'efficacité du modèle.

[ ]: