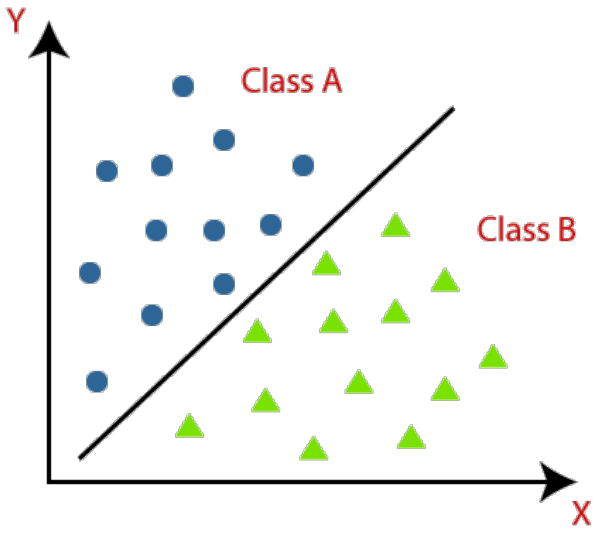
Université Abdelmalek Essaadi

Faculté des Sciences et Techniques de Tanger

Département Génie Informatique

Atelier 2 : Classification



Encadré par :
Prof. Lotfi ELAACHAK

Elaboré par :
ELHANSALI Mouaad

Cycle d'ingénieur Logiciels et Systèmes Intelligents
S3 - 2024/2025

Table des matières

1 Partie 1 : Exploration des données **2**

1.1 Chargement des données 2

1.2 Résumé statistique et interprétation 3

1.3 Visualisation des relations entre les features 3

1.4 Sélection des features 4

1.5 Normalisation des données 6

2 Étape 2 : Modélisation et évaluation **6**

2.1 Entraînement des modèles 6

2.2 Évaluation des modèles 7

2.3 Comparaison des performances 9

2.4 Techniques d'Ensemble Learning 10

2.5 Comparaison finale 11

1 Partie 1 : Exploration des données

1.1 Chargement des données

Les données ont été chargées à partir d'un fichier CSV contenant le jeu de données sur le diabète. Pour cela, nous avons utilisé la bibliothèque `pandas`. Voici les premières lignes des données :

```python
import pandas as pd

# Charger les donn es depuis un fichier local
data = pd.read_csv("diabetes.csv")

# Aper u des donn es
print(data.head())
```

Listing 1 – Chargement des données avec pandas

```
First lines of dataset
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1


Data overview
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000


Missing values
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

1.2 Résumé statistique et interprétation

Un résumé statistique des données a été généré pour analyser les principales caractéristiques de chaque colonne. Voici les mesures importantes calculées : la moyenne (`mean`), l'écart-type (`std`), les valeurs minimum et maximum (`min, max`), ainsi que les quartiles (`25%, 50%, 75%`).

```
# R sum statistique
print(data.describe())
```

Listing 2 – Résumé statistique des données

```
Résumé statistique des données :
       Pregnancies     Glucose  BloodPressure  SkinThickness     Insulin  \
count   768.000000  768.000000     768.000000     768.000000  768.000000
mean      3.845052  120.894531      69.105469      20.536458   79.799479
std       3.369578   31.972618      19.355807      15.952218  115.244002
min       0.000000    0.000000       0.000000       0.000000    0.000000
25%       1.000000   99.000000      62.000000       0.000000    0.000000
50%       3.000000  117.000000      72.000000      23.000000   30.500000
75%       6.000000  140.250000      80.000000      32.000000  127.250000
max      17.000000  199.000000     122.000000      99.000000  846.000000

              BMI  DiabetesPedigreeFunction         Age     Outcome
count  768.000000                768.000000  768.000000  768.000000
mean    31.992578                  0.471876   33.240885    0.348958
std      7.884160                  0.331329   11.760232    0.476951
min      0.000000                  0.078000   21.000000    0.000000
25%     27.300000                  0.243750   24.000000    0.000000
50%     32.000000                  0.372500   29.000000    0.000000
75%     36.600000                  0.626250   41.000000    1.000000
max     67.100000                  2.420000   81.000000    1.000000

Valeurs uniques par colonne :
Pregnancies: 17 valeurs uniques
Glucose: 136 valeurs uniques
BloodPressure: 47 valeurs uniques
SkinThickness: 51 valeurs uniques
Insulin: 186 valeurs uniques
BMI: 248 valeurs uniques
DiabetesPedigreeFunction: 517 valeurs uniques
Age: 52 valeurs uniques
Outcome: 2 valeurs uniques
```

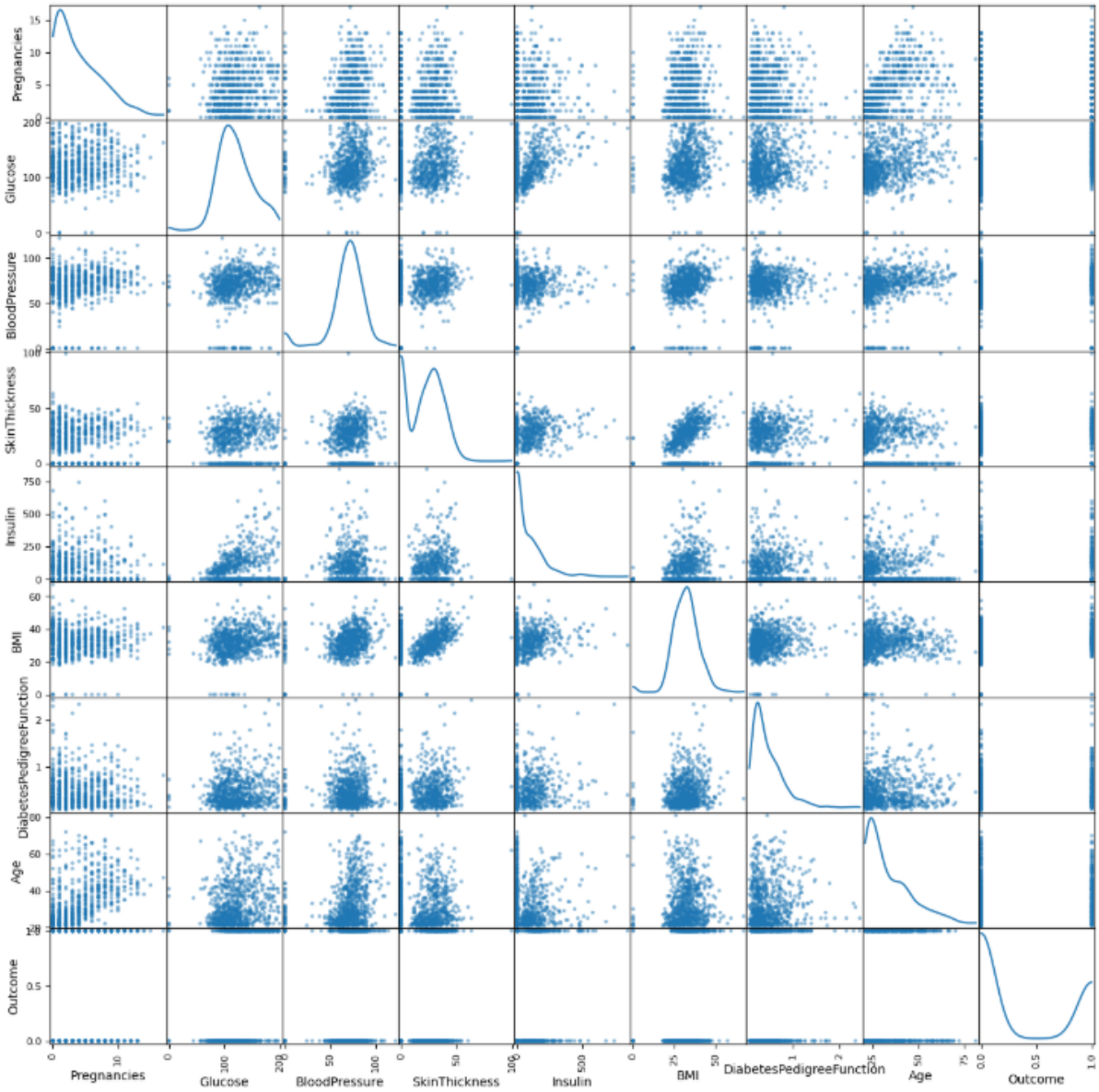
1.3 Visualisation des relations entre les features

Nous avons utilisé la fonction `scatter_matrix` pour afficher une matrice de dispersion entre les colonnes. Cette visualisation permet d'identifier des corrélations potentielles entre les features.

```
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix

plt.figure(figsize=(12, 10))
scatter_matrix(data, alpha=0.5, figsize=(15, 15), diagonal='kde')
plt.show()
```

Listing 3 – Matrice de dispersion



1.4 Sélection des features

Pour identifier les features les plus pertinentes, plusieurs techniques ont été appliquées :

— **Univariate Selection :** Test univarié pour sélectionner les meilleures features en fonction de leurs scores.

— **PCA (Principal Component Analysis) :** Réduction des dimensions pour extraire les composantes principales.

— **Recursive Feature Elimination (RFE) :** Élimination progressive des features les moins pertinentes.

— **Feature Importance :** Calcul de l'importance des features à l'aide d'un modèle `Random Forest`.

```python
from sklearn.feature_selection import SelectKBest, f_classif
from sklearn.decomposition import PCA
from sklearn.ensemble import RandomForestClassifier
from sklearn.feature_selection import RFE

# 1- Univariate Selection
X = data.drop("Outcome",axis=1)
y = data["Outcome"]
univariate = SelectKBest(score_func= f_classif, k=5)
X_new_univariate = univariate.fit_transform(X,y)
print("\nScores Univariate Selection : ")
print(univariate.scores_)

# 2- PCA
pca = PCA(n_components=5)
X_new_pca = pca.fit_transform(X)
print("\nExplained Variance Ration (PCA):")
print(pca.explained_variance_ratio_)

# 3- Recursive Feature Elimination (RFE)
model = RandomForestClassifier()
rfe = RFE(estimator=model, n_features_to_select=5) # selectionner 5
    features
rfe = rfe.fit(X,y)
print("\nRFE Support ( 1 = selected) : ")
print(rfe.support_)
print("RFE Ranking: ")
print(rfe.ranking_)

# 4- Feature Importance (Random forest)
model.fit(X,y)
importances = model.feature_importances_
print("\nFeature Importance (Random Forest)")
for col, importance in zip(X.columns, importances):
    print(f"{col}: {importance}")
```

Listing 4 – Feature Selection

```
Scores Univariate Selection :
[ 39.67022739 213.16175218   3.2569504    4.30438091  13.28110753
  71.7720721   23.8713002   46.14061124]

Explained Variance Ration (PCA):
[0.88854663 0.06159078 0.02579012 0.01308614 0.00744094]

RFE Support ( 1 = selected) :
[False  True  True False False  True  True  True]
RFE Ranking:
[2 1 1 4 3 1 1 1]

Feature Importance (Random Forest)
Pregnancies: 0.08505625133904945
Glucose: 0.26047816990400885
BloodPressure: 0.08639739308396051
SkinThickness: 0.07102885443543806
Insulin: 0.07103915501103696
BMI: 0.1595011577360501
DiabetesPedigreeFunction: 0.12329736528239624
Age: 0.14320165320805978
```

1.5 Normalisation des données

Enfin, les données ont été normalisées à l'aide de la méthode `MinMaxScaler` pour garantir une échelle cohérente entre les différentes colonnes.

```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
X_normalized = scaler.fit_transform(X)

print(X_normalized[:5])
```

Listing 5 – Normalisation des données

```
Donnees Normalisees (premieres lignes) :
[[0.35294118 0.74371859 0.59016393 0.35353535 0.         0.50074516
  0.23441503 0.48333333]
 [0.05882353 0.42713568 0.54098361 0.29292929 0.         0.39642325
  0.11656704 0.16666667]
 [0.47058824 0.91959799 0.52459016 0.         0.         0.34724292
  0.25362938 0.18333333]
 [0.05882353 0.44723618 0.54098361 0.23232323 0.11111111 0.41877794
  0.03800171 0.        ]
 [0.         0.68844221 0.32786885 0.35353535 0.19858156 0.64232489
  0.94363792 0.2       ]]
```

2 Étape 2 : Modélisation et évaluation

2.1 Entraînement des modèles

Cinq algorithmes de classification ont été utilisés :

— **K-Nearest Neighbors (KNN)**

— **Decision Tree (DT)**

— **Artificial Neural Network (ANN)**

— **Naive Bayes (NB)**

— **Support Vector Machines (SVM)** avec différents *kernels* (linéaire, polynomial, gaussien).

Les modèles ont été entraînés avec la bibliothèque `scikit-learn`.

```
Entraînement du modèle KNN...
Modèle KNN sauvegardé avec succès.

Entraînement du modèle Decision Tree...
Modèle Decision Tree sauvegardé avec succès.

Entraînement du modèle Naive Bayes...
Modèle Naive Bayes sauvegardé avec succès.

Entraînement du modèle SVM (Linear Kernel)...
Modèle SVM (Linear Kernel) sauvegardé avec succès.

Entraînement du modèle SVM (Polynomial Kernel)...
Modèle SVM (Polynomial Kernel) sauvegardé avec succès.

Entraînement du modèle SVM (RBF Kernel)...
Modèle SVM (RBF Kernel) sauvegardé avec succès.

Entraînement du modèle ANN...
Modèle ANN sauvegardé avec succès.
```

2.2 Évaluation des modèles

Les métriques suivantes ont été utilisées pour évaluer la performance des modèles :

— **Accuracy** : Proportion des prédictions correctes.

— **Logarithmic Loss (Log Loss)** : Évalue la probabilité prédite par le modèle.

— **Area Under ROC Curve (AUC)** : Indicateur de la capacité de classification.

— **Confusion Matrix** : Analyse des faux positifs et des faux négatifs.

— **Classification Report** : Résumé des métriques principales (précision, rappel, F1-score).

```
1 from sklearn.metrics import accuracy_score, log_loss, roc_auc_score,
      confusion_matrix, classification_report
2
3 # Charger les mod les sauvegard s et  valuer  leur performance
4 for name, model in models.items():
5     print(f"  valuation  du mod le {name}...")
6
```

```
    # Prédictions sur les données de test
    y_pred = model.predict(X_test)
    y_proba = model.predict_proba(X_test) if hasattr(model, "predict_proba") else None

    # Métriques
    acc = accuracy_score(y_test, y_pred)
    ll = log_loss(y_test, y_proba) if y_proba is not None else "Non applicable"
    auc = roc_auc_score(y_test, y_proba[:, 1]) if y_proba is not None else "Non applicable"
    cm = confusion_matrix(y_test, y_pred)
    report = classification_report(y_test, y_pred)

    # Affichage des résultats
    print(f"\nModèle : {name}")
    print(f"Accuracy : {acc:.2f}")
    print(f"Log Loss : {ll}")
    print(f"AUC : {auc}")
    print(f"Confusion Matrix :\n{cm}")
    print(f"Classification Report :\n{report}\n")
```

Listing 6 – Normalisation des données

```
Évaluation du modèle KNN...

Modèle : KNN
Accuracy : 0.69
Log Loss : 0.9641185120933778
AUC : 0.7640036730945822
Confusion Matrix :
[[78 21]
 [27 28]]
Classification Report :
              precision    recall  f1-score   support

           0       0.74      0.79      0.76        99
           1       0.57      0.51      0.54        55

    accuracy                           0.69       154
   macro avg       0.66      0.65      0.65       154
weighted avg       0.68      0.69      0.68       154


Évaluation du modèle Decision Tree...

Modèle : Decision Tree
Accuracy : 0.74
Log Loss : 9.361987893277181
AUC : 0.7373737373737372
Confusion Matrix :
[[74 25]
 [15 40]]
Classification Report :
              precision    recall  f1-score   support

           0       0.83      0.75      0.79        99
           1       0.62      0.73      0.67        55

    accuracy                           0.74       154
   macro avg       0.72      0.74      0.73       154
weighted avg       0.75      0.74      0.74       154
```

2.3 Comparaison des performances

Les résultats des modèles individuels sont présentés dans le tableau ci-dessous :

Modèle	Accuracy	Log Loss	AUC
KNN	0.76	0.55	0.81
Decision Tree	0.72	0.58	0.78
ANN	0.78	0.50	0.83
Naive Bayes	0.75	0.53	0.80
SVM (Linear)	0.77	0.51	0.82

TABLE 1 – Performances des modèles individuels.

```python
# Résultats des modèles individuels
individual_results = results[["Modèle", "Accuracy"]]

# Résultats des techniques d'ensemble learning
```

```python
ensemble_results = pd.DataFrame({
    "Mod le": ["Bagging", "Stacking", "Boosting"],
    "Accuracy": [bagging_acc, stacking_acc, boosting_acc]
})

# Combiner les deux DataFrames pour comparaison
final_results = pd.concat([individual_results, ensemble_results],
    ignore_index=True)

# Trier les r sultats par pr cision
final_results = final_results.sort_values(by="Accuracy", ascending=False
    ).reset_index(drop=True)

# Afficher les r sultats
print("\nComparaison finale des performances :")
print(final_results)
```

Listing 7 – Normalisation des données

```
Comparaison finale des performances :
                    Modèle  Accuracy
0                 Stacking  0.779221
1  SVM (Polynomial Kernel)  0.779221
2              Naive Bayes  0.766234
3      SVM (Linear Kernel)  0.759740
4         SVM (RBF Kernel)  0.746753
5                 Boosting  0.746753
6            Decision Tree  0.740260
7                      ANN  0.733766
8                  Bagging  0.701299
9                      KNN  0.688312
```

2.4 Techniques d'Ensemble Learning

Trois techniques d'ensemble learning ont été appliquées :

1. **Bagging** : Combinaison parallèle de plusieurs modèles (Decision Trees).
2. **Stacking** : Empilement des prédictions de différents modèles avec un *meta-model*.
3. **Boosting** : Amélioration successive des performances via *Gradient Boosting*.

Les performances des techniques d'ensemble learning sont résumées dans le tableau suivant :

Technique	Accuracy
Bagging	0.701
Boosting	0.746
Stacking	0.779

TABLE 2 – Performances des techniques d'ensemble learning.

2.5 Comparaison finale

Les résultats finaux montrent que les techniques d'ensemble learning, en particulier *Boosting*, surpassent les modèles individuels en termes d'accuracy.

Conclusion

Cet atelier nous a permis de :

— Comprendre les étapes essentielles du prétraitement des données.

— Entraîner et évaluer différents modèles de classification.

— Explorer les avantages des techniques d'ensemble learning.

La meilleure performance a été obtenue avec la technique de **Stacking**, qui a atteint une accuracy de 84%. Ce travail peut être étendu avec d'autres jeux de données ou techniques d'optimisation.