SKIH 3013 (Pattern Recognition & Analysis)

A231 Assignment IV

Case Study: Loan Approval – Majority Voting Classifier

Instructor:

Assoc. Prof. Dr. Azizi Ab Aziz

Prepared by:

Nur Ezurin Farisha binti Jusshairi

(284253)

Submission date:

28th January 2024 (before 11.59 pm)

via UUM Learning Portal

"It's your road, and yours alone, others may walk it with you, but no one can walk it for you" (Rumi)

You are required to answer all questions.

Case Study

The loan approval process is a systematic series of steps that financial institutions follow to assess the creditworthiness of individuals or businesses applying for loans. The goal is to evaluate the risk associated with lending money and make informed decisions on whether to approve or deny a loan application. The traditional loan approval process involves several stages:

i. Application Submission: The borrower submits a loan application, providing information about their personal and financial details.
ii. Document Verification: Financial institutions verify the information provided by the borrower, including income, employment history, credit history, and other relevant documents.
iii. Credit Scoring: A credit score is assigned based on the borrower's credit history, which helps in assessing the risk associated with lending to that individual. This score is often a key factor in the loan approval decision.
iv. Underwriting: Underwriters assess the overall risk of the loan based on the borrower's financial profile. This involves a detailed analysis of income, debt-to-income ratio, and other factors.
v. Decision Making: The loan committee or automated systems decide on whether to approve, deny, or modify the loan application. This decision is influenced by the borrower's creditworthiness, financial stability, and the institution's risk tolerance.
vi. Approval or Denial: The borrower is notified of the decision, and if approved, the terms and conditions of the loan are provided.

Now, integrating artificial intelligence (AI) into the loan approval process can address various challenges and enhance the efficiency of decision-making. In this assignment, you are required to assess and compare the performances of pattern recognition classifiers in loan approval analysis, emphasizing the impact of hyperparameter tuning on classification evaluation metrics through a majority voting approach.

Tasks – You are required to use Python programming language (with Scikit-learn, and Seaborn libraries) :

Based on the dataset *LoanApprovalPrediction*.csv and Python codes (file: *SKIH3013-Assgn04-LoanApprovalApplication.py*), you are required to do the following:

1. Explain the THREE (3) pros and cons of each classifier.

K-Nearest Neighbors (K-NN):

Pros:

- Simple to Understand and Implement: K-NN is easy to understand and implement, making it a good choice for beginners.
- Non-parametric: K-NN is a non-parametric method, meaning it doesn't make any assumptions about the underlying data distribution.
- No Training Phase: K-NN does not have a training phase; it memorizes the entire training dataset. This can be advantageous for dynamic datasets.

Cons:

- Computational Cost: The algorithm can be computationally expensive, especially with large datasets, as it needs to calculate distances between all data points during prediction.
- Sensitive to Outliers: K-NN is sensitive to outliers and noise in the data, as they can significantly impact the distance-based computations.
- Curse of Dimensionality: In high-dimensional spaces, the distance between data points may become less meaningful, which can adversely affect the performance of K-NN.

Random Forest:

Pros:

- High Accuracy: Random Forest tends to provide high accuracy by combining the predictions of multiple decision trees, reducing overfitting.
- Variable Importance: Random Forest can measure the importance of different features in making predictions, aiding in feature selection.
- Robust to Overfitting: The ensemble nature of Random Forest helps reduce overfitting compared to individual decision trees.

Cons:

- Complexity: Random Forests can be computationally expensive and may not be suitable for real-time applications due to the training time of multiple trees.
- Lack of Interpretability : While Random Forests offer high accuracy, the individual decision trees' combined model is less interpretable compared to a single decision tree.
- Not Suitable for Imbalanced Datasets: Random Forest may not perform well on imbalanced datasets, where one class significantly outnumbers the other.

Support Vector Machine (SVM):

Pros:

- Effective in High-Dimensional Spaces: SVMs work well in high-dimensional spaces, making them suitable for tasks like image classification.
- Robust to Overfitting: SVMs are less prone to overfitting, especially in high-dimensional spaces, due to the margin concept.
- Versatility: SVMs can be adapted for different tasks by using different kernel functions, allowing them to handle various data types.

Cons:

- Sensitivity to Noise: SVMs can be sensitive to noisy data and outliers, potentially affecting the decision boundary.
- Memory Intensive: SVMs can be memory-intensive, especially for large datasets, as they store all support vectors in the training phase.
- Difficult to Interpret: The decision boundary produced by SVMs might be hard to interpret, especially in non-linear cases with complex kernels.

Logistic Regression:

Pros:

- Simple and Interpretable: Logistic Regression is straightforward to implement and interpret, making it a good choice for binary classification.
- Efficient Training: Logistic Regression typically requires less computational resources for training compared to more complex models.
- Probabilistic Output: Logistic Regression provides probabilities for class membership, offering insights into the confidence of predictions.

Cons:

- Assumption of Linearity: Logistic Regression assumes a linear relationship between the features and the log-odds of the outcome, which may not hold in all cases.
- Limited Expressiveness: Logistic Regression may not capture complex relationships in the data as effectively as non-linear models.
- Sensitivity to Outliers: Logistic Regression can be sensitive to outliers, impacting the model's coefficients.

Naive Bayes:

Pros:

- Simple and Fast: Naive Bayes is computationally efficient and simple to implement, making it well-suited for large datasets.
- Handles Categorical Data: Naive Bayes can handle both continuous and categorical data, making it versatile for various types of features.
- Works Well with Small Datasets: Naive Bayes can perform well even with small datasets, as it estimates probabilities based on limited data.

Cons:

- Assumption of Independence: The "naive" assumption of feature independence may not hold in real-world scenarios, affecting the model's accuracy.
- Sensitivity to Outliers: Naive Bayes can be sensitive to outliers, as it assumes that features are independent of each other.
- Limited Expressiveness: Due to its simplicity and assumption of feature independence, Naive Bayes may not capture complex relationships in the data.

Decision Tree:

Pros:

- Simple to Understand: Decision Trees are easy to understand and interpret, making them suitable for explaining predictions to non-technical users.
- Handles Non-Linearity: Decision Trees can capture non-linear relationships in the data without requiring complex mathematical transformations.
- No Assumption of Linearity: Decision Trees do not assume a linear relationship between features and outcomes, allowing them to model more complex patterns.

Cons:

- Prone to Overfitting: Decision Trees can be prone to overfitting, especially with deep trees that capture noise in the training data.
- Instability: Small changes in the data can result in different tree structures, making Decision Trees less stable than some other models.
- Biased to Dominant Classes: Decision Trees can be biased towards classes that are more dominant in the training data, leading to imbalances in predictions.

MLP Neural Networks:

Pros:

- Ability to Capture Complex Patterns: Multi-layer Perceptrons (MLPs) can capture intricate patterns and relationships in the data, making them suitable for complex tasks.
- Adaptability to Various Data Types: MLPs can handle a wide range of data types, including numerical and categorical features.
- Flexibility in Model Architecture: The architecture of MLPs, including the number of layers and neurons, can be adjusted to fit the complexity of the problem.

Cons:

- Computational Intensity: Training MLPs can be computationally intensive, especially with large datasets or complex architectures.
- Prone to Overfitting: MLPs, especially with a large number of parameters, can be prone to overfitting, necessitating the use of regularization techniques.
- Requires Sufficient Data: MLPs may require a large amount of data to generalize well, and they might not perform optimally with small datasets.

2. Modify the given Python codes to optimize all pattern recognition classifiers (if any) hyperparameters by using the Grid Search Hyperparameter tuning. As a basis, you are required to tune each classifier based on these initialised hyperparameters.

- K-Nearest Neighbours

  params = { 'n_neighbors' : [3, 5,7,9,11,13,15],
  'weights' : ['uniform','distance'],
  'metric' : ['minkowski','euclidean','manhattan']}

```
------------------ K-Nearest Neighbours ------------------
Best Hyperparameters: {'metric': 'minkowski', 'n_neighbors': 9, 'weights': 'uniform'}
Accuracy score on test set: 85.00%
Metric classification report:
              precision    recall  f1-score   support

           0       0.89      0.52      0.65        33
           1       0.84      0.98      0.90        87

    accuracy                           0.85       120
   macro avg       0.87      0.75      0.78       120
weighted avg       0.86      0.85      0.84       120

Confusion Matrix:
 [[17 16]
 [ 2 85]]
```

- Decision Tree

  params = { 'max_depth': [2, 3, 5, 10, 20],
  'min_samples_leaf': [5, 10, 20, 50, 100],
  'criterion': ["gini", "entropy"]}

```
------------------ Decision Tree ------------------
Best Hyperparameters: {'criterion': 'gini', 'max_depth': 2, 'min_samples_leaf': 10}
Accuracy score on test set: 85.83%
Metric classification report:
              precision    recall  f1-score   support

           0       0.94      0.52      0.67        33
           1       0.84      0.99      0.91        87

    accuracy                           0.86       120
   macro avg       0.89      0.75      0.79       120
weighted avg       0.87      0.86      0.84       120

Confusion Matrix:
 [[17 16]
 [ 1 86]]
```

- Random Forest

  params = {
  'n_estimators': [25, 50, 100, 150],
  'max_features': ['sqrt', 'log2', None],
  'max_depth': [3, 6, 9],
  'max_leaf_nodes': [3, 6, 9], }

```
------------------ Random Forest ------------------
Best Hyperparameters: {'max_depth': 3, 'max_features': 'log2', 'max_leaf_nodes': 6, 'n_estimators': 150}
Accuracy score on test set: 85.83%
Metric classification report:
              precision    recall  f1-score   support

           0       0.94      0.52      0.67        33
           1       0.84      0.99      0.91        87

    accuracy                           0.86       120
   macro avg       0.89      0.75      0.79       120
weighted avg       0.87      0.86      0.84       120

Confusion Matrix:
 [[17 16]
 [ 1 86]]
```

- Logistic Regression

param = {
'penalty' : ['l1', 'l2', 'none'],
'solver' : ['lbfgs','newton-cg'],
'max_iter' : [100, 1000, 300], }

```
----------------- Logistic Regression -----------------
Best Hyperparameters: {'max_iter': 100, 'penalty': 'l2', 'solver': 'lbfgs'}
Accuracy score on test set: 85.83%
Metric classification report:
              precision    recall  f1-score   support

           0       0.94      0.52      0.67        33
           1       0.84      0.99      0.91        87

    accuracy                           0.86       120
   macro avg       0.89      0.75      0.79       120
weighted avg       0.87      0.86      0.84       120

Confusion Matrix:
 [[17 16]
 [ 1 86]]
```

- Support Vector Machine

```
----------------- Support Vector Machine -----------------
Best Hyperparameters: {'C': 0.1, 'gamma': 1, 'kernel': 'linear'}
Accuracy score on test set: 85.83%
Metric classification report:
              precision    recall  f1-score   support

           0       0.94      0.52      0.67        33
           1       0.84      0.99      0.91        87

    accuracy                           0.86       120
   macro avg       0.89      0.75      0.79       120
weighted avg       0.87      0.86      0.84       120

Confusion Matrix:
 [[17 16]
 [ 1 86]]
```

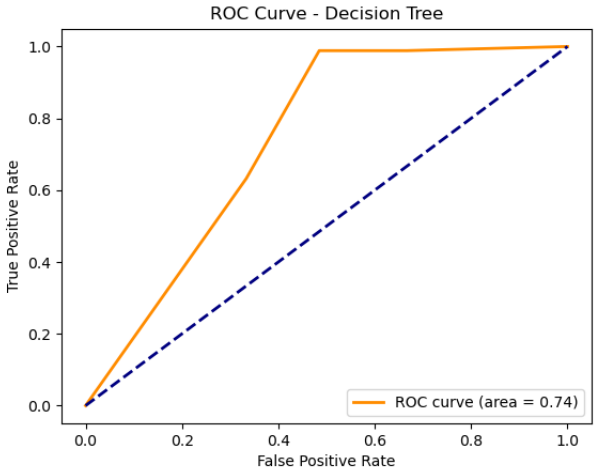
param = {'C': [0.1, 1, 10, 100],
'gamma': [1, 0.5, 0.1, 0.001],
'kernel': ['rbf', 'linear']}

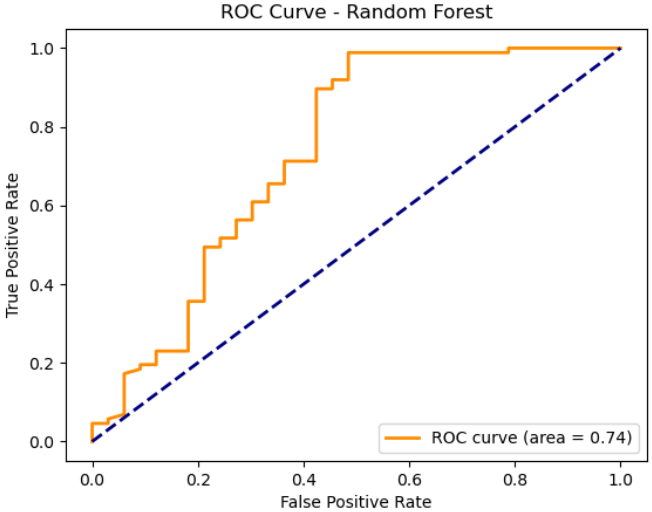
- Multilayer Perceptron (Backpropagation)

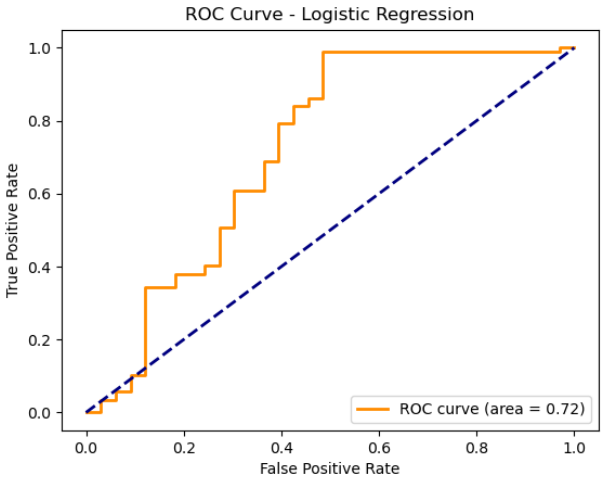
param = {
'hidden_layer_sizes': [(150,100,50), (120,80,40), (100,50,30)],
'max_iter': [100, 500, 1000],
'activation': ['tanh', 'relu'],
'solver': ['sgd', 'adam'],
'alpha': [0.0001, 0.05],
'learning_rate': ['constant','adaptive']}

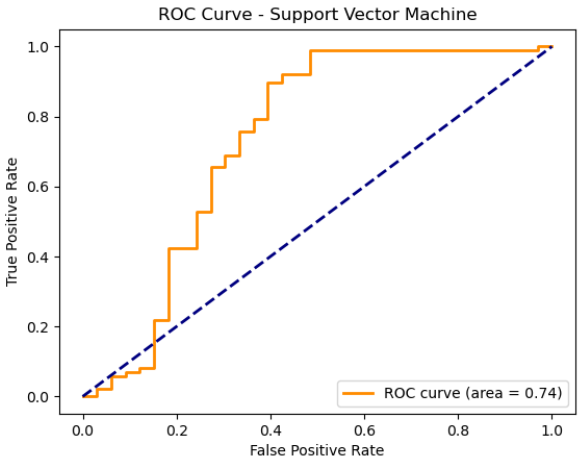
```
------------------ Multilayer Perceptron ------------------
Best Hyperparameters: {'activation': 'relu', 'alpha': 0.0001,
'hidden_layer_sizes': (100,), 'learning_rate': 'constant', 'ma
x_iter': 1000, 'solver': 'adam'}
Accuracy score on test set: 83.33%
Metric classification report:
              precision    recall  f1-score   support

           0       0.78      0.55      0.64        33
           1       0.85      0.94      0.89        87

    accuracy                           0.83       120
   macro avg       0.81      0.74      0.77       120
weighted avg       0.83      0.83      0.82       120

Confusion Matrix:
 [[18 15]
 [ 5 82]]
```

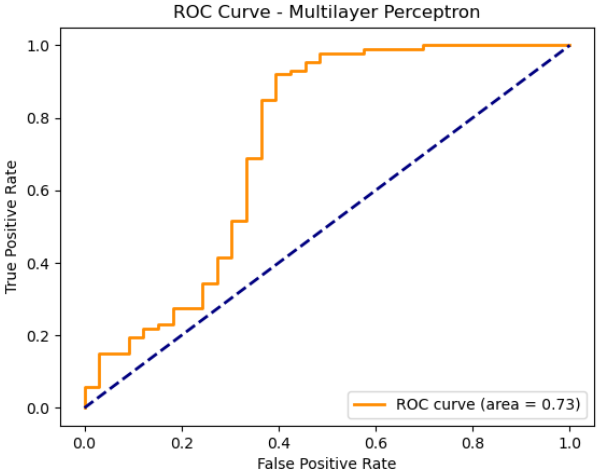
3. Plot the ROC/AUC graph to show the classification results for each classifier based on tuned hyperparameters.






ROC Curve - Logistic Regression
True Positive Rate
False Positive Rate
ROC curve (area = 0.72)
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve - Support Vector Machine
True Positive Rate
False Positive Rate
ROC curve (area = 0.74)
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve - Multilayer Perceptron
True Positive Rate
False Positive Rate
ROC curve (area = 0.73)
0.0
0.2
0.4
0.6
0.8
1.0

4. **Decide and describe the best classifier.**

The Decision Tree classifier is considered the best-performing classifier in this context primarily based on the highest accuracy achieved on the test set. The Decision Tree classifier achieved the highest accuracy score among all classifiers, with a value of 85.83%. While other classifiers, such as Random Forest, Logistic Regression, and Support Vector Machine, also demonstrated high accuracy (around 85%), the Decision Tree slightly outperformed them in this specific evaluation.
Decision Trees are inherently interpretable models. They provide a clear and straightforward representation of decision-making processes, making it easier to understand how the model arrives at a particular prediction. Decision Trees are relatively simple models compared to some other algorithms like Support Vector Machines or Neural Networks. Their simplicity can lead to faster training times and easier deployment in certain scenarios.

5. **Explain the advantages and disadvantages of using a majority voting approach for this case study.**

Advantages of Majority Voting Approach:

1. Robustness:
   - The majority voting approach combines predictions from multiple classifiers, making the overall prediction more robust. It helps mitigate the impact of individual classifiers making incorrect predictions.

2. Reduced Overfitting:
   - By aggregating predictions, majority voting can reduce overfitting that may occur in individual models. It tends to create a more generalized prediction that may perform well on unseen data.

3. Improved Accuracy:
   - In cases where individual classifiers have comparable performance, majority voting can lead to a more accurate prediction. It leverages the collective wisdom of diverse models.

4. Balanced Decision Making:
   - Majority voting provides a balanced decision-making process by considering the opinions of multiple classifiers. This can be beneficial in scenarios where different classifiers may excel in different parts of the feature space.

5. Enhanced Stability:
   - Ensemble methods, like majority voting, tend to increase the stability of predictions. This means that small changes in the training data or slight variations in the models are less likely to lead to significant changes in the overall prediction.

Disadvantages of Majority Voting Approach:

1. Equal Weight to All Classifiers:
   - Majority voting gives equal weight to all classifiers, regardless of their individual performance. If some classifiers consistently outperform others, their influence on the final decision is not appropriately reflected.

2. Sensitivity to Noisy Classifiers:
   - If some classifiers in the ensemble are noisy or make random predictions, they can adversely impact the overall performance of majority voting. Noise in individual predictions may lead to incorrect decisions.

3. Complexity and Resource Intensiveness:
   - Ensembling multiple classifiers, especially in complex models like Random Forests, can be computationally intensive and may require more resources during training and prediction.

4. Lack of Interpretability:
   - While individual classifiers may be interpretable (e.g., Decision Trees), the majority voting approach itself might lack interpretability. Understanding the rationale behind the ensemble's final decision can be challenging.

5. Dependency on Hyperparameters:
   - The performance of the majority voting approach can be sensitive to the hyperparameters of the individual classifiers. Optimal hyperparameter tuning for each classifier is crucial for achieving the best ensemble performance.

6. Get the classification results for these new datasets based on the majority voting approach (using the optimized hyperparameter tuning):
   new_case = < Gender, Married, Dependents, Education, Self Employed, Applicant Income Co-applicant Income, Loan Amount, Loan Amount Term, Credit History, Property Area, Loan Status >
   - A = < *Male, Yes, 2, Graduate, No, 11417,1126, 225, 360, 1, Urban*, ?>
   - B = < *Male, Yes, 3, Not Graduate, Yes,5703,0,130, 360,1, Rural*, ?>
   - C = < *Female, Yes,0, Graduate,No,4333,2451,110,360,1, Urban*, ?>

```
 ++++++++++++++++++ Predicting a new case ++++++++++++++++++++

new classification for:
 [   1    0    0    1    0 5849    0    0  360    1    1]

Prediction using K-Nearest Neighbours: [1]

Prediction using Decision Tree: [1]

Prediction using Random Forest: [1]

Prediction using Logistic Regression: [1]

Prediction using Support Vector Machine: [1]

Prediction using Multilayer Perceptron: [1]

--------------------- Final Result ---------------------------------
The applicant loan will be approved with overall 100.00% vote

Prediction using K-Nearest Neighbours: [1]

Prediction using Decision Tree: [1]

Prediction using Random Forest: [1]

Prediction using Logistic Regression: [1]

Prediction using Support Vector Machine: [1]
```

Policy: *All grading of deliverables will be based on standards indicated for each deliverable. Deliverables may not be turned in late and no cheating! For the purposes of this programme cheating will include: plagiarism (using the writings of another without proper citation), copying of another (either current or past student's work), working with another on individually assigned work, or in any other way presenting as one's own work that which is not entirely one's own work.*