Title:

Group Member Names :

Sanket Nagshetti (200565218)

Shayam Kishore Kotapati (200600387)

INTRODUCTION:

AIM :

This project focuses on enhancing the accuracy of supervised machine learning algorithms for simplified cancer

classification. By employing various models and fine-tuning parameters, we aim to optimize the system's

performance and achieve more precise diagnostic results.

Github Repo:

https://github.com/sanket-nagshetti/Machine_Learning_Programming1_Final-Project
****
****

DESCRIPTION OF PAPER:

This project developed a decision support system (DSS) to assist clinicians in early cancer diagnosis by classifying five different cancer types using 20 cancer exomes and their mutations. Initially, several supervised machine learning algorithms—K-nearest neighbor (KNN), support vector machine (SVM), decision tree, naïve bayes, and random forest—were tested, with decision tree and random forest showing the highest preliminary accuracy. To improve accuracy, 16 key features were selected, and datasets were balanced using SMOTE. The refined Random Forest model achieved an accuracy of 82% and demonstrated superior performance with an area under the curve closer to 1 compared to the decision tree model. The model was validated using Matthew's correlation coefficient (MCC), with high scores confirming its reliability. The final model was deployed as a user-friendly web application using Streamlit, facilitating effective cancer classification.

PROBLEM STATEMENT :

This project seeks to improve the accuracy of cancer classification using supervised machine learning algorithms by optimizing model performance and fine-tuning parameters for more precise diagnostics.

CONTEXT OF THE PROBLEM:

The project focuses on enhancing the accuracy of cancer classification by utilizing supervised machine learning algorithms. By optimizing model performance and fine-tuning parameters, the goal is to achieve more precise diagnostics. This involves experimenting with various machine learning models, adjusting their settings, and evaluating their effectiveness on cancer datasets to improve overall classification accuracy.

SOLUTION:

To potentially increase accuracy, try using the MLPClassifier to see if it improves performance. Additionally, increase the number of trees in the Random Forest from 10 to 100. Also, optimize hyperparameters by limiting the maximum tree depth to 10 and adjusting the minimum samples required for splits and leaves.

Background

|Reference|Explanation|Dataset/Input|Weakness| |Pagel KA, Kim R, Moad K, et al.|Integrated informatics analysis of cancerrelated variants|Cancer Dataset| Criteria to choose parameters not clearly mentioned| |------|------|------|------|

Implement paper code :

Adding new classification technique MLPClassifier to check whether performance will increase or not

```
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report

# Initialize the MLP model
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500,
random_state=42)
# Fit the MLP model on the balanced training data
mlp.fit(X_train_ros, y_train_ros)
```

```python
# Predict using the MLP model on the test data
y_pred_mlp = mlp.predict(x_test)
# Print the classification report for MLP model
print("MLP Classifier Classification Report:")
print(classification_report(y_test, y_pred_mlp))
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from sklearn.neural_network import MLPClassifier
from imblearn.over_sampling import SMOTE

# Initialize models
dtc = DecisionTreeClassifier()
rf = RandomForestClassifier()
mlp = MLPClassifier(hidden_layer_sizes=(100,), max_iter=500,
random_state=42)

# Fit models on the balanced training data
dtc.fit(X_train_ros, y_train_ros)
rf.fit(X_train_ros, y_train_ros)
mlp.fit(X_train_ros, y_train_ros)

# Make predictions
y_pred_dtc = dtc.predict(x_test)
y_pred_rf = rf.predict(x_test)
y_pred_mlp = mlp.predict(x_test)

# Print classification reports
print("Decision Tree Classification Report:")
print(classification_report(y_test, y_pred_dtc))

print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred_rf))

print("MLP Classifier Classification Report:")
print(classification_report(y_test, y_pred_mlp))
```

Tunning parameters to increase the performance of RandomForestClassifier

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
from imblearn.over_sampling import SMOTE

# Initialize and fit SMOTE
smote = SMOTE(random_state=42)
X_train_ros, y_train_ros = smote.fit_resample(x_train, y_train)

# Initialize Random Forest Classifier with optimized parameters
classifier1 = RandomForestClassifier(
    n_estimators=100,         # A larger number of trees for better
performance
    max_depth=10,              # Limit depth to avoid overfitting
    min_samples_split=5,       # Minimum samples required to split an
internal node
    min_samples_leaf=2,        # Minimum samples required to be at a
leaf node
    max_features='sqrt',       # Use a subset of features for each
split
    random_state=42
)

# Fit the classifier
classifier1.fit(X_train_ros, y_train_ros)

# Predict on test data
y_pred1 = classifier1.predict(x_test)

# Print classification report
print("Random Forest Classification Report:")
print(classification_report(y_test, y_pred1))
```

Contribution Code :

Results :

Accuracy for MLP Classifier Classification Report = 75%

Accuracy for Random forest Classifier after tunning parameter = 82%

Observations :

- Introducing the MLP model did not result in a significant improvement over the Random Forest model, which continues to be the best-performing model in terms of overall accuracy and F1-scores. The MLP model provides similar performance to the Decision Tree model but does not surpass Random Forest. Therefore, while MLP contributes to the analysis, Random Forest remains the preferred model for this classification problem.
- We improved the accuracy from 81% to 82% by increasing the number of trees in the Random Forest from 10 to 100, which enhanced the model's robustness. We optimized hyperparameters, such as limiting the maximum tree depth to 10 and adjusting the minimum samples for splits and leaves, which helped balance model complexity and generalization. Additionally, we implemented feature selection to focus on the most important features, reducing noise and improving performance. These changes collectively refined the model, leading to the increased accuracy.

Conclusion and Future Direction :

In conclusion, the Random Forest model remains the superior choice for our classification problem, outperforming both the Decision Tree and MLP models in terms of accuracy and F1-scores. While the introduction of the MLP model did not yield a significant improvement, our efforts to enhance the Random Forest model proved successful. By increasing the number of trees, optimizing hyperparameters, and implementing feature selection, we managed to improve the model's accuracy from 81% to 82%. These refinements have made the Random Forest model more robust and better suited to the classification task at hand, solidifying its position as the preferred model.

Learnings :

This experience highlighted the importance of model selection, with Random Forest outperforming MLP and Decision Tree models in accuracy and F1-scores. Incremental improvements, such as increasing tree count and optimizing hyperparameters, significantly enhanced the Random Forest model's performance, improving accuracy from 81% to 82%. Feature selection further refined the model by focusing on key features, reducing noise. Although the MLP model didn't surpass Random Forest, exploring alternative models provided valuable insights. Overall, this process emphasized the value of careful tuning and robustness in developing effective machine learning models.

Results Discussion :

The results of our analysis demonstrate that the Random Forest model consistently outperformed both the Decision Tree and MLP models in terms of accuracy and F1-scores. Despite the introduction of the MLP model, which offered performance comparable to the Decision Tree model, it did not surpass the accuracy and robustness of Random Forest. This suggests that for this particular classification problem, Random Forest is better suited due to its ability to handle complex patterns and interactions within the data.

The improvement in Random Forest's accuracy from 81% to 82% was achieved through targeted enhancements, including increasing the number of trees and fine-tuning hyperparameters like maximum tree depth and the minimum samples required for splits and leaves. These adjustments helped balance the model’s complexity and generalization ability, resulting in a more robust and accurate model.

Moreover, the implementation of feature selection played a crucial role in this improvement by reducing noise and focusing on the most important features, which streamlined the model's decision-making process. This not only improved accuracy but also contributed to the model’s overall efficiency.

In summary, while exploring alternative models like MLP is valuable, the Random Forest model remains the most effective choice for this classification task. The results underscore the importance of model selection, hyperparameter tuning, and feature selection in optimizing machine learning models for better performance.
****
****

Limitations :

This analysis has several limitations. First, only a few models were explored, potentially overlooking more effective alternatives. The results are specific to the dataset used, limiting generalizability. Increasing the number of trees in the Random Forest model improved accuracy but at a higher computational cost, which may not be feasible in resource-limited settings. Additionally, there is a risk of overfitting, and the model's performance is sensitive to hyperparameter settings and dependent on the quality of selected features. These factors could affect the model's robustness and replication in different scenarios.

Future Extension :

Future extensions could include exploring additional models like Gradient Boosting or Support Vector Machines to identify more effective alternatives. Testing on different datasets would help validate the Random Forest model's generalizability. Further optimization through advanced techniques like automated hyperparameter tuning or ensemble methods could improve accuracy while managing computational costs. Addressing overfitting with cross-validation or regularization would enhance robustness. Additionally, refining feature engineering and selection could further improve model performance, providing a deeper understanding of its capabilities across various contexts.

References:

1. Liu Sheng OR. Decision support for healthcare in a new information age. Decis Support Syst. 2000;30:101-103.
2. Hicks JK, Dunnenberger HM, Gumpper KF, Haidar CE, Hoffman JM. Integrating pharmacogenomics into electronic health records with clinical decision support. Am J Health Syst Pharm. 2016;73:1967-1976.