Week 5 Hands -on Activity

Group: Neha Thakur, Mukesh Ravi,

Amoolsiri Sunkaraboina , Jithendar Amanaganti

Model Comparison & Reflection:

Summary table comparing the models' performance based on the metrics.

Metric	SVM (Initial)	GBM (Initial)	Random Forest (Initial)	SVM (Tuned)	GBM (Tuned)	Random Forest (Tuned)
Accuracy	0.7049	0.7705	0.8361	0.7541	0.8033	0.8525
Precision	0.6667	0.8	0.8438	0.7143	0.8333	0.8571
Recall	0.875	0.75	0.8438	0.8571	0.7857	0.8571
F1 Score	0.7568	0.7742	0.8438	0.7805	0.8085	0.8571
AUC-ROC	0.8394	0.903	0.9203	0.8654	0.9152	0.9301

Metric	Decision Tree	Random Forest
Accuracy	0.74948	0.82395
Precision	0.66667	0.78788
Recall	0.75758	0.78788
F1 Score	0.70909	0.78788
AUC-ROC	0.83440	Not provided

Reflect on which model performed best and why. Discuss how hyperparameter tuning affected your results.

Best Performing Model:

From the results obtained the Random Forest Classifier seems to have the highest accuracy, both with and without the application of hyperparameters optimization steps. It was the best in most aspect with the highest figure in all the Assessment, such as Accuracy, Precision, Recall, F1-Score, AUC-ROC.

Reasons for Best Performance :

Ensemble Method: Random Forest is an extension of boosting techniques such as decision trees, but it uses many decision trees simultaneously, hence minimizing on the overfitting encountered when using a single decision tree thereby enhancing performance when used for testing data.

Robustness:

Despite this, it is less sensitive to noise and can immediately input multiple features, ideal for most challenging datasets such as healthcare data. Adverse effects of hyperparameters tuning

Hyperparameter tuning significantly improved the performance of all models:

SVM: The accuracy of the SVM model rose from 0.7049 to 0.7541. This has been very useful in the tuning process, where the values of C, gamma, as well as the kernel, were able to be found to fine tune the model's performance to classify the data appropriately.

GBM: In the present study, the transition from the previously applied GBM model accuracy from 0.7705 to new 0.8033 was observed. Other hyperparameters such as estimators, learning rate and max_depth were tweaked to be able to capture data patterns better.

Random Forest: From the Random Forest model, the accuracy raised from 0.8361 to 0.8525. Other hyperparameters that were fitted to the model include, n_estimators, max_depth, min_samples_split, and min_samples_leaf. Hyperparameter tuning and its specific explanations

SVM: To control the trade off between getting a small error on the training data and the margin, a parameter C was tuned during the tuning process. The thing that sheds light upon the extent of how far a single training example affects its corresponding output is the gamma parameter. Kernel parameter determines what type of hyperplane will be used in order to separate data or samples. Thus, if the parameters were adjusted to their correct level, the performance of the SVM model increased to a great extent.

GBM: In the case of the GBM model, all the parametersn_estimators, learning_rate and max_depth worked well in controlling the balance between bias and variance. The high value of the estimators and the low value of the learning rate usually result in more accurate models, yet they consume more resources.

Random Forest: We notice here that also the tuning techniques such as number of trees, max depth, and so on like Number of trees (n_estimators), Maximum trees depth (max_depth), Minimum number of samples required to split an internal node (min_samples_split), Minimum number of samples required at each leaf node (min_samples_leaf) have played its role in the enhancement of the accuracy and the decrease in variability of the model. These parameters determine the complexity of the trees and, therefore, the overall model, and improve generalization.

In general, hyperparameters tuning was critical in fine-tuning the performance of all models and identify the best setting to balance high bias and high variance and thus improve the model's generalization on the test data.

GitHub link: https://github.com/Mukhesh19/Week-5-Hands-On-Activity-5pts-Extra-credit-

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.ensemble import GradientBoostingClassifier,
RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_auc_score

# Load the dataset
file_path ='heart.csv'  # Update this path if your file is in a
different location
heart_data = pd.read_csv(file_path)

# Split the data into features (X) and target (y)
X = heart_data.drop("target", axis=1)
y = heart_data["target"]

# Split the dataset into training and test sets (80% training, 20%
test)
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Initialize the models
svm_model = SVC(probability=True, random_state=42)
gbm_model = GradientBoostingClassifier(random_state=42)
rf_model = RandomForestClassifier(random_state=42)

# Train the models
svm_model.fit(X_train, y_train)
gbm_model.fit(X_train, y_train)
rf_model.fit(X_train, y_train)

# Predict on the test set
svm_pred = svm_model.predict(X_test)
gbm_pred = gbm_model.predict(X_test)
rf_pred = rf_model.predict(X_test)

# Calculate probabilities for ROC-AUC
svm_prob = svm_model.predict_proba(X_test)[:, 1]
gbm_prob = gbm_model.predict_proba(X_test)[:, 1]
rf_prob = rf_model.predict_proba(X_test)[:, 1]

# Evaluate each model using various metrics
svm_metrics = {
    'Accuracy': accuracy_score(y_test, svm_pred),
    'Precision': precision_score(y_test, svm_pred),
    'Recall': recall_score(y_test, svm_pred),
    'F1 Score': f1_score(y_test, svm_pred),
    'AUC-ROC': roc_auc_score(y_test, svm_prob)
```

```python
}

gbm_metrics = {
    'Accuracy': accuracy_score(y_test, gbm_pred),
    'Precision': precision_score(y_test, gbm_pred),
    'Recall': recall_score(y_test, gbm_pred),
    'F1 Score': f1_score(y_test, gbm_pred),
    'AUC-ROC': roc_auc_score(y_test, gbm_prob)
}

rf_metrics = {
    'Accuracy': accuracy_score(y_test, rf_pred),
    'Precision': precision_score(y_test, rf_pred),
    'Recall': recall_score(y_test, rf_pred),
    'F1 Score': f1_score(y_test, rf_pred),
    'AUC-ROC': roc_auc_score(y_test, rf_prob)
}

# Combine results into a DataFrame for comparison
results_df = pd.DataFrame([svm_metrics, gbm_metrics, rf_metrics],
                          index=['SVM', 'GBM', 'Random Forest'])

print("Initial Model Evaluation Results:")
print(results_df)

# Hyperparameter tuning using GridSearchCV for each model

# Define parameter grids for each model
param_grid_svm = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf', 'linear']
}

param_grid_gbm = {
    'n_estimators': [50, 100, 150],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 4, 5]
}

param_grid_rf = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Set up GridSearchCV for each model
grid_svm = GridSearchCV(SVC(probability=True, random_state=42),
param_grid_svm, cv=5, scoring='accuracy')
```

```
grid_gbm = GridSearchCV(GradientBoostingClassifier(random_state=42),
param_grid_gbm, cv=5, scoring='accuracy')
grid_rf = GridSearchCV(RandomForestClassifier(random_state=42),
param_grid_rf, cv=5, scoring='accuracy')

# Fit the grid searches to the data
grid_svm.fit(X_train, y_train)
grid_gbm.fit(X_train, y_train)
grid_rf.fit(X_train, y_train)

# Get the best models and parameters
best_svm = grid_svm.best_estimator_
best_gbm = grid_gbm.best_estimator_
best_rf = grid_rf.best_estimator_

print("\nBest Hyperparameters for each model:")
print("SVM:", grid_svm.best_params_)
print("GBM:", grid_gbm.best_params_)
print("Random Forest:", grid_rf.best_params_)

# Re-evaluate each model using the best hyperparameters

# Predict on the test set using the best models
best_svm_pred = best_svm.predict(X_test)
best_gbm_pred = best_gbm.predict(X_test)
best_rf_pred = best_rf.predict(X_test)

# Calculate probabilities for ROC-AUC using the best models
best_svm_prob = best_svm.predict_proba(X_test)[:, 1]
best_gbm_prob = best_gbm.predict_proba(X_test)[:, 1]
best_rf_prob = best_rf.predict_proba(X_test)[:, 1]

# Evaluate each model using the best parameters
best_svm_metrics = {
    'Accuracy': accuracy_score(y_test, best_svm_pred),
    'Precision': precision_score(y_test, best_svm_pred),
    'Recall': recall_score(y_test, best_svm_pred),
    'F1 Score': f1_score(y_test, best_svm_pred),
    'AUC-ROC': roc_auc_score(y_test, best_svm_prob)
}

best_gbm_metrics = {
    'Accuracy': accuracy_score(y_test, best_gbm_pred),
    'Precision': precision_score(y_test, best_gbm_pred),
    'Recall': recall_score(y_test, best_gbm_pred),
    'F1 Score': f1_score(y_test, best_gbm_pred),
    'AUC-ROC': roc_auc_score(y_test, best_gbm_prob)
}

best_rf_metrics = {
```

```
    'Accuracy': accuracy_score(y_test, best_rf_pred),
    'Precision': precision_score(y_test, best_rf_pred),
    'Recall': recall_score(y_test, best_rf_pred),
    'F1 Score': f1_score(y_test, best_rf_pred),
    'AUC-ROC': roc_auc_score(y_test, best_rf_prob)
}

# Combine results into a DataFrame for comparison after tuning
tuned_results_df = pd.DataFrame([best_svm_metrics, best_gbm_metrics,
best_rf_metrics],
                                index=['Tuned SVM', 'Tuned GBM',
'Tuned Random Forest'])

print("\nModel Evaluation Results After Hyperparameter Tuning:")
print(tuned_results_df)
```

```
Initial Model Evaluation Results:
               Accuracy  Precision   Recall  F1 Score   AUC-ROC
SVM            0.704918   0.666667  0.87500  0.756757  0.839440
GBM            0.770492   0.800000  0.75000  0.774194  0.903017
Random Forest  0.836066   0.843750  0.84375  0.843750  0.920259

Best Hyperparameters for each model:
SVM: {'C': 100, 'gamma': 1, 'kernel': 'linear'}
GBM: {'learning_rate': 0.2, 'max_depth': 5, 'n_estimators': 50}
Random Forest: {'max_depth': None, 'min_samples_leaf': 4,
'min_samples_split': 2, 'n_estimators': 200}

Model Evaluation Results After Hyperparameter Tuning:
                     Accuracy  Precision  Recall  F1 Score   AUC-ROC
Tuned SVM            0.836066   0.866667  0.8125  0.838710  0.921336
Tuned GBM            0.803279   0.857143  0.7500  0.800000  0.907328
Tuned Random Forest  0.852459   0.848485  0.8750  0.861538  0.938578
```