

Model Optimization and Tuning Phase Report

| | |
|---------------|--------------------------------------|
| Date | 12 JULY 2024 |
| ID | 740069 |
| Project Title | Lymphography Classification using ML |
| Maximum Marks | 10 Marks |

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining machine learning models for peak performance. It includes optimized model code, fine-tuning hyperparameters, comparing performance metrics, and justifying the final model selection for enhanced predictive accuracy and efficiency.

| Model | Tuned Hyperparameters | Optimal Values |
|---------------------|--|---|
| Logistic Regression | <pre>1 model_lr = LogisticRegression() 2 model_lr.fit(x_train,y_train) 3 lr_pred_test=model_lr.predict(x_test) 4 lr_pred_train=model_lr.predict(x_train) 5 test_acc_lr=accuracy_score(y_test,lr_pred_test) 6 train_acc_lr=accuracy_score(y_train,lr_pred_train) 7 print('logistic @ression test accuracy: ',test_acc_lr) 8 print(classification_report(y_test,lr_pred_test))</pre> | <pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, lr_pred) print(f'Tuned Hyperparameters: {test_params}') print(f'Accuracy on test set: {accuracy}') Tuned Hyperparameters: {'criterion': 'gini', 'max_depth': 100, 'max_features': 0.5, 'max_iter': 1000, 'min_samples_split': 10, 'solver': 'lbfgs'} Accuracy on test set: 0.9250000000000001</pre> |





| | | |
|-----|---|---|
| SVM | <pre> 1 from sklearn.svm import SVC 2 model_svm = SVC() 3 model_svm.fit(x_train,y_train) 4 svm_pred_test=model_svm.predict(x_test) 5 svm_pred_train=model_svm.predict(x_train) 6 test_acc_svm=accuracy_score(y_test,svm_pred_test) 7 train_acc_svm=accuracy_score(y_train,svm_pred_train) 8 print('SVM Test Accuracy: ',test_acc_svm) 9 print(classification_report(y_test,svm_pred_test)) </pre> | <pre> # Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print('Optimal Hyperparameters: (best_params)') print('Accuracy on Test Set: (accuracy)') Optimal Hyperparameters: {'gamma': 0.001, 'kernel': 'rbf', 'max_depth': 20, 'min_samples_leaf': 1, 'min_samples_split': 1, 'n_estimators': 100} Accuracy on Test Set: 0.7734753294852 </pre> |
|-----|---|---|

Hyperparameter Tuning Documentation (6 Marks):

| | | |
|-------------------|---|--|
| KNN | <pre> [34]: 1 knn_model = KNeighborsClassifier(n_neighbors = 100 2 knn_model.fit(X_train, y_train) 3 knn_score = metrics.accuracy_score(y_test, knn_pred) * 100 4 5 # Print classification report for knn 6 print('----- Regular Training Set used -----') 7 print('Classification report for (knn):', knn_model.metrics.classification_report(y_test, knn_pred)) 8 print('Accuracy score: ', knn_score) </pre> | <pre> # Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print('Optimal Hyperparameters: (best_params)') print('Accuracy on Test Set: (accuracy)') Optimal Hyperparameters: {'n_neighbors': 9, 'p': 1, 'weights': 'distance'} Accuracy on Test Set: 0.7218934811242884 </pre> |
| Gradient Boosting | <pre> 1 from sklearn.ensemble import GradientBoostingClassifier 2 gbc=GradientBoostingClassifier() 3 gbc.fit(x_train,y_train) 4 gbc_pred_test=gbc.predict(x_test) 5 gbc_pred_train=gbc.predict(x_train) 6 gbc_test_acc=accuracy_score(y_test,gbc_pred_test) 7 gbc_train_acc=accuracy_score(y_train,gbc_pred_train) 8 print('GB accuracy is: ',gbc_test_acc) 9 print(classification_report(y_test,gbc_pred_test)) </pre> | <pre> # Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print('Optimal Hyperparameters: (best_params)') print('Accuracy on Test Set: (accuracy)') Optimal Hyperparameters: {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 1, 'min_samples_split': 1, 'n_estimators': 100, 'subsample': 0.9} Accuracy on Test Set: 0.5888888888888889 </pre> |

Performance Metrics Comparison Report (2 Marks):

| Model | Optimized Metric |
|-------|------------------|
|-------|------------------|

| | |
|---------------------|--|
| Logistic Regression | <div>  <pre> Logistic Regression test accuracy: 0.8 precision recall f1-score support 2.0 0.71 1.00 0.83 12 3.0 1.00 0.71 0.83 17 4.0 0.00 0.00 0.00 1 accuracy 0.80 30 macro avg 0.57 0.57 0.55 30 weighted avg 0.85 0.80 0.80 30 </pre> </div> <div> <pre>[] 1 confusion_matrix(y_test,lr_pred_test)</pre> </div> <div>  <pre> array([[12, 0, 0], [4, 12, 1], [1, 0, 0]]) </pre> </div> |
| SVM | <div>  <pre> SVM Test Accuracy: 0.8 precision recall f1-score support 2.0 0.73 0.92 0.81 12 3.0 0.87 0.76 0.81 17 4.0 0.00 0.00 0.00 1 accuracy 0.80 30 macro avg 0.53 0.56 0.54 30 weighted avg 0.78 0.80 0.79 30 </pre> </div> <div> <pre>[] 1 confusion_matrix(y_test,svm_pred_test)</pre> </div> <div>  <pre> array([[11, 1, 0], [4, 13, 0], [0, 1, 0]]) </pre> </div> |

| | |
|-------------------|---|
| KNN | <pre> 81.35593220338984 ----- Regular Training Set Used ----- Classification report for KNeighborsClassifier(): precision recall f1-score support 2.0 0.87 0.93 0.90 14 3.0 0.80 0.86 0.83 14 4.0 0.00 0.00 0.00 2 accuracy 0.83 30 macro avg 0.56 0.60 0.57 30 weighted avg 0.78 0.83 0.80 30 Accuracy score: 83.33333333333334 </pre> |
| Gradient Boosting | <pre> GB accuracy is: 0.8333333333333334 precision recall f1-score support 2.0 0.73 0.92 0.81 12 3.0 0.93 0.82 0.87 17 4.0 0.00 0.00 0.00 1 accuracy 0.83 30 macro avg 0.56 0.58 0.56 30 weighted avg 0.82 0.83 0.82 30 </pre> <pre> 1 GBC.predict([[4.0,2.0,1.0,1.0,1.0,1.0,1.0,2.0,1.0,2.0,2.0,2.0,4.0,8.0,1.0,1.0,2.0,2.0]]) array([3.]) </pre> |

Final Model Selection Justification (2 Marks):

| Final Model | Reasoning |
|-------------|-----------|
| | |

| | |
|-----|--|
| KNN | The KNN model was selected for its superior performance, exhibiting high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model. |
|-----|--|