Model Optimization and Tuning Phase

Date	5 July 2024
Team ID	SWTID1720080161
Project Title	Revolutionizing Liver Care : Predicting Liver Cirrhosis Using Advanced Machine Learning Techniques
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.

Hyperparameter Tuning Documentation (6 Marks):

Model	Tuned Hyperparameters	Optimal Values

SUPPORT VECTOR MACHINE	HP1: Baseline Parameters
	`model = svm.SVC()`
	`model.fit(X_resampled, y_resampled)`
	`y_pred = model.predict(X_test)`
	`print("Test Accuracy:", accuracy_score(y_test, y_pred))`
	HP2: C=0.1 , kernel=rbf
	`model = svm.SVC(C=0.1,kernel="rbf")`
	`model.fit(X_resampled, y_resampled)`
	`y_pred = model.predict(X_test)`
	`print("Test Accuracy:", accuracy_score(y_test, y_pred))`
		Value1: Test Accuracy:
		0.902834008097166
		`Test Accuracy: 0.902834008097166`
		Value2 : Test Accuracy :
		0.8663967611336032
		`Test Accuracy: 0.8663967611336032`

<table>
<tr>
<td>LOGISTIC REGRESSION</td>
<td>HP1: max_iter=1000,
penalty="l2",
solver="lbfgs",
C=0.001


```
from sklearn.metrics import confusion_matrix, classification_report

model = LogisticRegression(max_iter=1000,penalty="l2",solver="lbfgs",C=0.001)
model.fit(X_resampled, y_resampled)


y_pred = model.predict(X_test)
```


HP2: penalty="l1"
C=0.01
solver="liblinear"


```
model = LogisticRegression(penalty="l1",C=0.01,solver="liblinear")
model.fit(X_resampled, y_resampled)


y_pred = model.predict(X_test)
```

</td>
<td>Value1: Test Accuracy:
0.9190283400809717


```
Test Accuracy: 0.9190283400809717
```


Value 2: Test Accuracy:
0.951417004048583


```
Test Accuracy: 0.951417004048583
```

</td>
</tr>
<tr>
<td>DECISION TREE CLASSIFIER</td>
<td>HP1: Baseline Parameter


```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
model.fit(X_resampled, y_resampled)
y_pred = model.predict(X_test)
```


HP2:


```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier(criterion="entropy", max_depth=3, min_samples_leaf=30
model.fit(X_resampled, y_resampled)
y_pred = model.predict(X_test)
```

</td>
<td>INEFFICIENT MODEL

Value1:


```
Test Accuracy: 1.0
```


Value2:


```
Test Accuracy: 0.9757085020242915
```

</td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

Model	Baseline Metric	Optimized Metric
SUPPORT VECTOR MACHINE	Test Accuracy: 0.902834008097166 Confusion Matrix: [[56 2] [22 167]] Classification Report: precision recall f1-score support 0 0.72 0.97 0.82 58 1 0.99 0.88 0.93 189 accuracy 0.90 247 macro avg 0.85 0.92 0.88 247 weighted avg 0.92 0.90 0.91 247	Test Accuracy: 0.8663967611336032 Confusion Matrix: [[46 12] [21 168]] Classification Report: precision recall f1-score support 0 0.69 0.79 0.74 58 1 0.93 0.89 0.91 189 accuracy 0.87 247 macro avg 0.81 0.84 0.82 247 weighted avg 0.88 0.87 0.87 247
LOGISTIC REGRESSION	Test Accuracy: 1.0 Confusion Matrix: [[58 0] [0 189]] Classification Report: precision recall f1-score support 0 1.00 1.00 1.00 58 1 1.00 1.00 1.00 189 accuracy 1.00 247 macro avg 1.00 1.00 1.00 247 weighted avg 1.00 1.00 1.00 247	Test Accuracy: 0.951417004048583 Confusion Matrix: [[56 2] [10 179]] Classification Report: precision recall f1-score support 0 0.85 0.97 0.90 58 1 0.99 0.95 0.97 189 accuracy 0.95 247 macro avg 0.92 0.96 0.94 247 weighted avg 0.96 0.95 0.95 247

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Logistic Regression with L1 Hyperparameterization	**Better Performance**: It has a high test accuracy, indicating better performance on unseen data. The precision is well balanced for both the classes. **Feature Selection**: L1 regularization helps in automatic feature selection, which can simplify the model and reduce the risk of overfitting, especially in imbalanced datasets. **Efficiency:** Logistic regression models are generally faster and less computationally intensive compared to SVMs, making them suitable for large datasets.