

Model Optimization and Tuning Phase Template

Date	Nov 30, 2024
Team ID	739891
Project Title	Unlocking the Minds: Analyzing Mental Health with NLP
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
SVC	<pre>[25]: from sklearn.svm import SVC from sklearn.metrics import classification_report, accuracy_score sv = SVC() sv.fit(x_train,y_train) y_pred = sv.predict(x_test) print(classification_report(y_test,y_pred))</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f"Optimal Hyperparameters: {best_params}") print(f"Accuracy on Test Set: {accuracy}") Optimal Hyperparameters: {'gamma': 0.001, 'kernel': 'rbf', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'best'} Accuracy on Test Set: 0.7559333333333333</pre>
Decision tree classifier	<pre>[27]: from sklearn.tree import DecisionTreeClassifier Dt = DecisionTreeClassifier() Dt.fit(x_train,y_train) y_pred = Dt.predict(x_test) print(classification_report(y_test,y_pred)) print(accuracy_score(y_test,y_pred))</pre>	<pre># Evaluate the performance of the tuned model accuracy = accuracy_score(y_test, y_pred) print(f"Optimal Hyperparameters: {best_params}") print(f"Accuracy on Test Set: {accuracy}") Optimal Hyperparameters: {'criterion': 'entropy', 'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100} Accuracy on Test Set: 0.7514792899404913</pre>

Random forest classifier	<pre> from sklearn.ensemble import RandomForestClassifier rf = RandomForestClassifier() rf.fit(x_train,y_train) y_pred = rf.predict(x_test) print(classification_report(y_test,y_pred)) print(accuracy_score(y_test,y_pred)) </pre>	<p>rate the performance of the base model</p> <pre> y = accuracy_score(y_test, y_pred) </pre> <p>Optional Hyperparameters: (best_params)</p> <p>Accuracy on Test Set: (accuracy)</p> <p>Hyperparameters: ('learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100, 'subsample': 0.8) y on Test Set: 0.7509940304637</p>
Ada boost classifier	<pre> [31]: from sklearn.ensemble import AdaBoostClassifier # clf = AdaBoostClassifier(algorithm='SAMME') ab = AdaBoostClassifier(algorithm='SAMME') ab.fit(x_train,y_train) y_pred = ab.predict(x_test) print(classification_report(y_test,y_pred)) print(accuracy_score(y_test,y_pred)) </pre>	<p>rate the performance of the base model</p> <pre> y = accuracy_score(y_test, y_pred) </pre> <p>Optional Hyperparameters: (best_params)</p> <p>Accuracy on Test Set: (accuracy)</p> <p>Hyperparameters: ('learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100, 'subsample': 0.8) y on Test Set: 0.7509940304637</p>
Gradient boosting classifier	<pre> [3]: from sklearn.ensemble import GradientBoostingClassifier gb = GradientBoostingClassifier() gb.fit(x_train,y_train) y_pred = gb.predict(x_test) print(classification_report(y_test,y_pred)) print(accuracy_score(y_test,y_pred)) </pre>	<p>rate the performance of the base model</p> <pre> y = accuracy_score(y_test, y_pred) </pre> <p>Optional Hyperparameters: (best_params)</p> <p>Accuracy on Test Set: (accuracy)</p> <p>Hyperparameters: ('learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100, 'subsample': 0.8) y on Test Set: 0.7509940304637</p>
Logistic Regression	<pre> [35]: from sklearn.linear_model import LogisticRegression lr = LogisticRegression() lr.fit(x_train,y_train) y_pred = lr.predict(x_test) print(classification_report(y_test,y_pred)) print(accuracy_score(y_test,y_pred)) </pre>	<p>rate the performance of the base model</p> <pre> y = accuracy_score(y_test, y_pred) </pre> <p>Optional Hyperparameters: (best_params)</p> <p>Accuracy on Test Set: (accuracy)</p> <p>Hyperparameters: ('learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100, 'subsample': 0.8) y on Test Set: 0.7509940304637</p>

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

SVC

	precision	recall	f1-score	support
0	0.89	0.94	0.91	4271
1	0.93	0.88	0.91	4121
accuracy			0.91	8392
macro avg	0.91	0.91	0.91	8392
weighted avg	0.91	0.91	0.91	8392

```
sv_acc = accuracy_score(y_test,y_pred)
sv_acc
0.9101525262154433
```

Decision tree classifier

	precision	recall	f1-score	support
0	0.82	0.83	0.82	4271
1	0.82	0.81	0.82	4121
accuracy			0.82	8392
macro avg	0.82	0.82	0.82	8392
weighted avg	0.82	0.82	0.82	8392

```
0.819351763584366
[28]: dt_acc = accuracy_score(y_test,y_pred)
      dt_acc
[28]: 0.819351763584366
```

Random forest classifier

	precision	recall	f1-score	support
0	0.87	0.91	0.89	4271
1	0.90	0.86	0.88	4121
accuracy			0.89	8392
macro avg	0.89	0.88	0.89	8392
weighted avg	0.89	0.89	0.89	8392

```
0.8852478551000953
[30]: RF_acc = accuracy_score(y_test,y_pred)
      RF_acc
[30]: 0.8852478551000953
```

Ada boost classifier

	precision	recall	f1-score	support
0	0.80	0.89	0.84	4271
1	0.87	0.76	0.81	4121
accuracy			0.83	8392
macro avg	0.83	0.83	0.83	8392
weighted avg	0.83	0.83	0.83	8392

```
0.8286463298379408
[32]: ab_acc = accuracy_score(y_test,y_pred)
      ab_acc
[32]: 0.8286463298379408
```

Gradient boosting classifier	<pre> precision recall f1-score support 0 0.83 0.92 0.87 4271 1 0.90 0.81 0.85 4121 accuracy 0.86 8392 macro avg 0.87 0.86 8392 weighted avg 0.87 0.86 8392 0.861415633937083 [34]: gb_acc = accuracy_score(y_test,y_pred) gb_acc [34]: 0.861415633937083 </pre>
Logistic Regression	<pre> precision recall f1-score support 0 0.89 0.94 0.91 4271 1 0.93 0.88 0.90 4121 accuracy 0.91 8392 macro avg 0.91 0.91 8392 weighted avg 0.91 0.91 8392 0.9074118207816969 [36]: lr_acc = accuracy_score(y_test,y_pred) lr_acc [36]: 0.9074118207816969 </pre>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
SVC (support vector classifier)	SVC performs well with limited data since it focuses on support vectors rather than the entire dataset. SVC's combination of flexibility (kernels), accuracy, robustness, and theoretical rigor makes it the "best" choice for projects where these qualities are critical and evaluating alternatives based on dataset's size, structure, and requirements is essential.