

Model Optimization and Tuning Phase Template

Date	15 March 2024
Team ID	SWTID1727180793
Project Title	SMS- Spam Detection Using NLP
Maximum Marks	10 Marks

Model Optimization and Tuning Phase

The Model Optimization and Tuning Phase involves refining neural network 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 (8 Marks):

Model	Tuned Hyperparameters	Optimal Values
DecisionTree	<pre>vectorizer = CountVectorizer() X = vectorizer.fit_transform(df['text']) X_train, X_test, y_train, y_test = train_test_split(X, df['label'], dt_classifier = DecisionTreeClassifier(criterion='entropy', max dt_classifier.fit(X_train, y_train) y_pred = dt_classifier.predict(X_test) print("Decision Tree - Accuracy:", accuracy_score(y_test, y_pre print("Classification Report:\n", classification_report(y_test,</pre>	<pre>print("Decision Tree - Accuracy:", accuracy_score(y_test, y_pred)) print("Classification Report:\n", classification_report(y_test, y_pred)) Index(['Unnamed: 0', 'label', 'text', 'label_num'], dtype='object') Decision Tree - Accuracy: 0.9439613526570049</pre>
Random Forest	<pre>vectorizer = CountVectorizer() X = vectorizer.fit_transform(df['text']) X_train, X_test, y_train, y_test = train_test_split(X, df['label'], test_size=0.2, random_state=42) rf_classifier = RandomForestClassifier(n_estimators=100, max_depth=30) rf_classifier.fit(X_train, y_train) y_pred = rf_classifier.predict(X_test) print("Random Forest - Accuracy:", accuracy_score(y_test, y_pred)) print("Classification Report:\n", classification_report(y_test, y_pred))</pre>	<pre>print("Random Forest - Accuracy:", accuracy_score(y_test, y_pred)) print("Classification Report:\n", classification_report(y_test, y_pred)) Random Forest - Accuracy: 0.9342995169082126</pre>

KNN	<pre>knn_classifier = KNeighborsClassifier() param_grid = { 'n_neighbors': [3, 5, 7, 9], 'weights': ['uniform', 'distance'], 'p': [1, 2] }</pre>	<pre>accuracy = accuracy_score(y_test, test_pred) print(f'Optimal hyperparameters: {grid_search.best_params_}') print(f'Accuracy on Test Set: {accuracy}')</pre> <p>Optimal hyperparameters: {'n_neighbors': 3, 'p': 2, 'weights': 'distance'} Accuracy on Test Set: 0.8782608695652174</p>
Gradient Boosting	<pre>gb_classifier = GradientBoostingClassifier() param_grid = { 'n_estimators': [50, 100, 200], 'learning_rate': [0.01, 0.1, 0.2], 'max_depth': [3, 4, 5], 'min_samples_split': [2, 5, 10], 'min_samples_leaf': [1, 2, 4], 'subsample': [0.8, 1.0] }</pre>	<pre>accuracy = accuracy_score(y_test, test_pred) print(f'Optimal hyperparameters: {grid_search.best_params_}') print(f'Accuracy on Test Set: {accuracy}')</pre> <p>Optimal hyperparameters: {'n_estimators': 100, 'learning_rate': 0.1, 'max_depth': 5, 'min_samples_split': 5, 'min_samples_leaf': 1, 'subsample': 0.8} Accuracy on Test Set: 0.8782608695652174</p>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
DecisionTree	<pre> Decision Tree - Accuracy: 0.9343362576613 Classification Report: precision recall f1-score support ham 0.96 0.96 0.96 742 spam 0.90 0.91 0.90 293 accuracy 0.94 1035 macro avg 0.93 0.93 0.93 1035 weighted avg 0.94 0.94 0.94 1035 </pre>