Model Optimization and Tuning Phase Report

Date	20 February 2026
Team ID	LTVIP2026TMIDS463
Project Title	Transfer Learning for Identifying Rotten Fruits and Vegetables
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

Decision Tree	```python
	# Define the Decision Tree classifier
	dt_classifier = DecisionTreeClassifier()
	
	# Define the hyperparameters and their possible values for tuning
	param_grid = {
	    'criterion': ['gini', 'entropy'],
	    'splitter': ['best', 'random'],
	    'max_depth': [None, 10, 20, 30, 40, 50],
	    'min_samples_split': [2, 5, 10],
	    'min_samples_leaf': [1, 2, 4]
	}```
		``python
		# 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': 'gini', 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'splitter': 'best'}
		Accuracy on Test Set: 0.715976331360947

Rando m Forest	```python
	# Define the Random Forest classifier
	rf_classifier = RandomForestClassifier()
	
	# Define the hyperparameters and their possible values for tuning
	param_grid = {
	    'n_estimators': [50, 100, 200],
	    'criterion': ['gini', 'entropy'],
	    'max_depth': [None, 10, 20, 30],
	    'min_samples_split': [2, 5, 10],
	    'min_samples_leaf': [1, 2, 4],
	}
	``
		``python
		# 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': 20, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
		Accuracy on Test Set: 0.7751479289940828

KNN	```python
	knn_classifier = KNeighborsClassifier()
	
	# Define the hyperparameters and their possible values for tuning
	param_grid = {
	    'n_neighbors': [3, 5, 7, 9],
	    'weights': ['uniform', 'distance'],
	    'p': [1, 2]
	}
	``
		``python
		# 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: {'n_neighbors': 9, 'p': 1, 'weights': 'distance'}
		Accuracy on Test Set: 0.7218934911242604

Gradien t Boostin g	```python
	# Define the Gradient Boosting classifier
	gb_classifier = GradientBoostingClassifier()
	
	# Define the hyperparameters and their possible values for tuning
	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]
	}
	``
		``python
		# 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: {'learning_rate': 0.1, 'max_depth': 5, 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 100, 'subsample': 0.8}
		Accuracy on Test Set: 0.7928994082840237

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric

Decision Tree	```python
	print(classification_report(y_test,y_pred))
	
	                          precision recall f1-score support
	
	    Loan will be Approved 0.67 0.68 0.68 75
	Loan will not be Approved 0.74 0.73 0.74 94
	
	                accuracy 0.71 169
	               macro avg 0.71 0.71 0.71 169
	            weighted avg 0.71 0.71 0.71 169
	
	confusion_matrix(y_test,y_pred)
	
	array([[51, 24],
	       [25, 69]])
	``

Random Forest	

```
print(classification_report(y_test,y_pred))

                          precision    recall  f1-score   support

    Loan will be Approved       0.71      0.83      0.77        75
Loan will not be Approved       0.84      0.73      0.78        94

                 accuracy                           0.78       169
                macro avg       0.78      0.78      0.77       169
             weighted avg       0.78      0.78      0.78       169
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[62, 13],
       [25, 69]])
```

KNN	

```
print(classification_report(y_test,y_pred))

                          precision    recall  f1-score   support

    Loan will be Approved       0.73      0.59      0.65        75
Loan will not be Approved       0.72      0.83      0.77        94

                 accuracy                           0.72       169
                macro avg       0.72      0.71      0.71       169
             weighted avg       0.72      0.72      0.72       169
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[44, 31],
       [16, 78]])
```

Gradient Boosting	

```
print(classification_report(y_test,y_pred))

                          precision    recall  f1-score   support

    Loan will be Approved       0.73      0.85      0.79        75
Loan will not be Approved       0.86      0.74      0.80        94

                 accuracy                           0.79       169
                macro avg       0.80      0.80      0.79       169
             weighted avg       0.80      0.79      0.79       169
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[64, 11],
       [24, 70]])
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Gradient Boosting	The Gradient Boosting 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.