

Model Optimization and Tuning Phase Template

Date	18 June 2025
Team ID	SWTID1749880888
Project Title	Prosperity Prognosticator: Machine Learning for Startup Success Prediction
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
Random forest	<pre>#Defining the random forest (classifier) rf = RandomForestClassifier(random_state=42) #Hyperparameters of Random Forest param_grid = { 'n_estimators': [100, 200], 'max_depth': [10, 20], 'min_samples_split': [2, 4], 'min_samples_leaf': [1, 2], 'bootstrap': [True, False] } grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, verbose=1) grid_search.fit(X_train, y_train)</pre>	<pre>from sklearn.metrics import accuracy_score, classification_report, confusion_matrix #Evaluating the test accuracy test_acc = accuracy_score(y_test, y_pred_test) train_acc = accuracy_score(y_train, y_pred_train) print("Test acc: ", test_acc) print("Train acc: ", train_acc) test_acc: 0.8274861111111111 train_acc: 1.0</pre>
Decision tree	<pre>#Importing and building the Decision Tree model from sklearn.model_selection import GridSearchCV #Hyperparameters of Decision Tree grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-1, verbose=1) grid_search.fit(X_train, y_train) print("Best parameters found: ", grid_search.best_params_)</pre>	<pre>#Printing the accuracy y_pred = grid_search.best_estimator_.predict(X_test) accuracy = accuracy_score(y_test, y_pred) print("Accuracy: ", accuracy) Accuracy: 0.8095138888888889</pre>

Knn model	<pre>[] #Importing and building the KNN model import pandas as pd from sklearn.neighbors import KNeighborsClassifier from sklearn.model_selection import train_test_split, GridSearchCV from sklearn.metrics import accuracy_score knn_classifier = KNeighborsClassifier() #Hyperparameters of KNN param_grid = { 'n_neighbors': [3, 5, 7, 9], 'weights': ('uniform', 'distance'), 'p': [1, 2] } grid_search = GridSearchCV(knn_classifier, param_grid, cv=5, n_jobs=-1, verbose=1) grid_search.fit(X_train, y_train)</pre>	<pre>[] #Printing the accuracy y_pred = grid_search.best_estimator_.predict(X_test) accuracy = accuracy_score(y_test, y_pred) print("Test Accuracy:", accuracy)</pre> <p>Test Accuracy: 0.6188888888888889</p>
-----------	--	---

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric																														
Random forest	<div><div><div></div></div><div><table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.77</td><td>0.58</td><td>0.66</td><td>86</td></tr><tr><td>1</td><td>0.81</td><td>0.91</td><td>0.86</td><td>166</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.80</td><td>252</td></tr><tr><td>macro avg</td><td>0.79</td><td>0.75</td><td>0.76</td><td>252</td></tr><tr><td>weighted avg</td><td>0.79</td><td>0.80</td><td>0.79</td><td>252</td></tr></table></div></div>		precision	recall	f1-score	support	0	0.77	0.58	0.66	86	1	0.81	0.91	0.86	166	accuracy			0.80	252	macro avg	0.79	0.75	0.76	252	weighted avg	0.79	0.80	0.79	252
		precision	recall	f1-score	support																										
	0	0.77	0.58	0.66	86																										
	1	0.81	0.91	0.86	166																										
	accuracy			0.80	252																										
	macro avg	0.79	0.75	0.76	252																										
	weighted avg	0.79	0.80	0.79	252																										
	<div><div><div></div></div><div><pre>[[50 36] [15 151]]</pre></div></div>																														
Decision tree	<div>Classification Report for Decision Tree:</div> <div><table><tr><th></th><th>precision</th><th>recall</th><th>f1-score</th><th>support</th></tr><tr><td>0</td><td>0.81</td><td>0.58</td><td>0.68</td><td>86</td></tr><tr><td>1</td><td>0.81</td><td>0.93</td><td>0.87</td><td>166</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>252</td></tr><tr><td>macro avg</td><td>0.81</td><td>0.75</td><td>0.77</td><td>252</td></tr><tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.80</td><td>252</td></tr></table></div>		precision	recall	f1-score	support	0	0.81	0.58	0.68	86	1	0.81	0.93	0.87	166	accuracy			0.81	252	macro avg	0.81	0.75	0.77	252	weighted avg	0.81	0.81	0.80	252
		precision	recall	f1-score	support																										
	0	0.81	0.58	0.68	86																										
	1	0.81	0.93	0.87	166																										
	accuracy			0.81	252																										
	macro avg	0.81	0.75	0.77	252																										
	weighted avg	0.81	0.81	0.80	252																										
	<div>Confusion Matrix for Decision Tree:</div> <div><pre>[[50 36] [12 154]]</pre></div>																														

KNN model	✦ Classification Report for KNN:				
		precision	recall	f1-score	support
	0	0.46	0.36	0.41	86
	1	0.70	0.78	0.74	166
	accuracy			0.64	252
	macro avg	0.58	0.57	0.57	252
	weighted avg	0.62	0.64	0.63	252
	Confusion Matrix for KNN:				
	[[31 55]				
	[36 130]]				

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	<p>It provides high accuracy, handles both classification and regression well, and is robust to overfitting due to its ensemble nature.</p> <p>It also performs well on structured/tabular data and gives insight into feature importance, making it ideal for our startup success prediction task.</p>