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	`#defining the random forest classifier`
	`rf = RandomForestClassifier(random_state=42)`
	
	`#Hyperparameteres 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)`
		`from sklearn.metrics import accuracy_score, classification_report, confusion_matrix`
		
		`#printing 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:  0.8174603174603174`
		`train_acc:  1.0`

Decision tree	`#importing and building the Decision Tree model`
	`from sklearn.model_selection import GridSearchCV`
	
	`#Hyperparameteres 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_)`
		`[ ] #printing the accuracy`
		`    y_pred = grid_search.best_estimator_.predict(X_test)`
		`    accuracy = accuracy_score(y_test, y_pred)`
		`    print("Accuracy:", accuracy)`
		
		`Accuracy: 0.8095238095238095`

<table>
<tr>
<td>Knn model</td>
<td><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()

    #Hyperparameteres 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></td>
<td><pre>
[ ] #printing the accuracy
    y_pred = grid_search.best_estimator_.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print("Test Accuracy:", accuracy)

Test Accuracy: 0.6388888888888888
</pre></td>
</tr>
</table>

Performance Metrics Comparison Report (2 Marks):

<table>
<tr><th>Model</th><th>Optimized Metric</th></tr>
<tr>
<td>Random forest</td>
<td><pre>
              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

[[50 36]
 [15 151]]
</pre></td>
</tr>
<tr>
<td>Decision tree</td>
<td><pre>
Classification Report for Decision Tree:
              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

Confusion Matrix for Decision Tree:
[[50 36]
 [12 154]]
</pre></td>
</tr>
</table>

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