

Model Optimization and Tuning Phase Template

Date	15 March 2024
Team ID	740018
Project Title	Determine : Loan from KIVA crowdfunding data
Maximum Marks	10 Marks

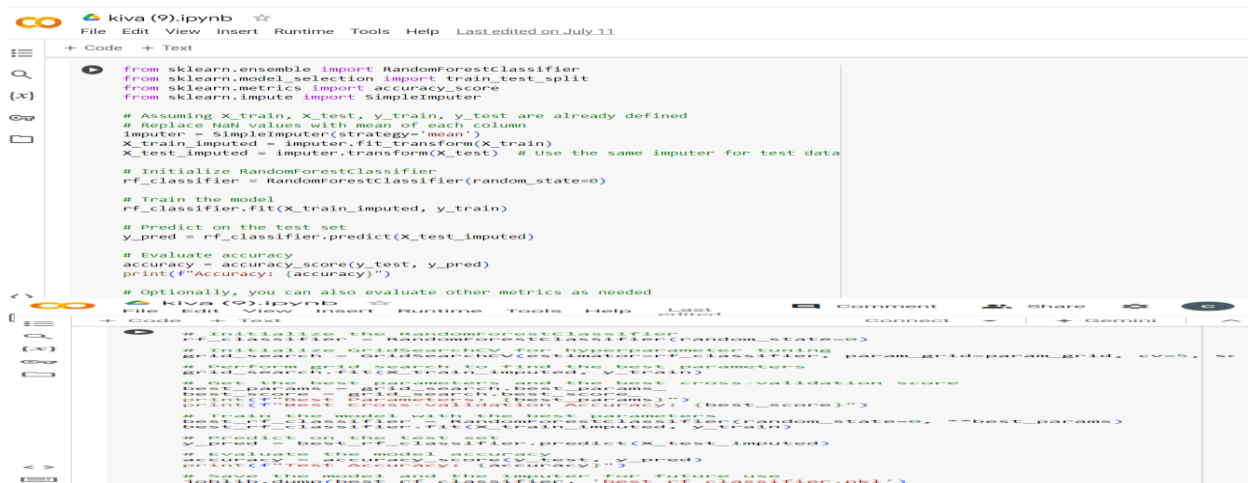
Model Optimization and Tuning Phase

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
-----	-----	-----

Performance Metrics Comparison Report (2 Marks):



```

kiva (9).ipynb
File Edit View Insert Runtime Tools Help Last edited on July 11

+ Code + Text

from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.impute import SimpleImputer

# Assuming X_train, X_test, y_train, y_test are already defined
# Replace NaN values with mean of each column
imputer = SimpleImputer(strategy='mean')
X_train_imputed = imputer.fit_transform(X_train)
X_test_imputed = imputer.transform(X_test) # Use the same imputer for test data

# Initialize RandomForestClassifier
rf_classifier = RandomForestClassifier(random_state=0)

# Train the model
rf_classifier.fit(X_train_imputed, y_train)

# Predict on the test set
y_pred = rf_classifier.predict(X_test_imputed)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Optionally, you can also evaluate other metrics as needed

# Initialize the RandomForestClassifier
rf_classifier = RandomForestClassifier(random_state=0)

# Initialize GridSearchCV for hyperparameter tuning
grid_search = GridSearchCV(estimator=rf_classifier, param_grid=param_grid, cv=5, scoring='accuracy')

# Perform Grid Search to find the best parameters
best_params = grid_search.best_params_
best_score = grid_search.best_score_
print(f"Best cross-validation accuracy: {best_score}")

# Initialize the RandomForestClassifier with the best parameters
best_rf_classifier = RandomForestClassifier(**best_params, random_state=0)

# Predict on the test set
y_pred = best_rf_classifier.predict(X_test_imputed)

# Evaluate the model accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")

# Save the model and the imputer for future use
joblib.dump(best_rf_classifier, 'best_rf_classifier.pkl')
  
```



```
Best Parameters: {'max_depth': None, 'min_samples_split': 2, 'n_estimators': 150}  
Best Cross-validation Accuracy: 0.6125  
Test Accuracy: 0.45  
['best_rf_classifier.pkl']
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random_forest Classifier	The Random_forest Classifier model was selected for its superior performance, exhibiting high accuracy during hyperparameter model. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model.