Model Optimization and Tuning Phase Template

Date	July 2024
Team ID	865503
Project Title	Frappe Activity: mobile Phone Activity classification
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
Bagging Classifier	Tuned a BaggingClassifier by first defining a DecisionTreeClassifier as the base estimator. Then, initialized the BaggingClassifier with this base estimator and specified the hyperparameters to tune using the param_dist dictionary. The key hyperparameters included n_estimators, max_samples, max_features, bootstrap, and bootstrap_features. Used RandomizedSearchCV to search for the best hyperparameter combination, employing 5-fold cross-validation and the accuracy metric to evaluate performance.

```
# Define the hyperparameters and their possible values for tuning

param_grid = {

    'n_estimators': [10, 50, 100],
    'max_samples': [0.5, 0.7, 1.0],
    'max_features': [0.5, 0.7, 1.0],
    'bootstrap': [True, False],
    'bootstrap_features': [True, False]
}
```

```
random_search = RandomizedSearchCV(estimator=bagging_classifier, param_distributions=param_grid,
                                   scoring='accuracy', cv=2, random_state=42)
```

```
random_search.fit(X_train,y_train)
```

```
    print("Best Parameters:",random_search.best_params_)
    print("Best Score:",random_search.best_score_)
```

[46]

```
Best Parameters: {'n_estimators': 100, 'max_samples': 0.7, 'max_features': 1.0, 'bootstrap_features': True, 'bootstrap': False}
Best Score: 0.6545086119554204
```

```
    print("Best Score:",random_search.score(X_test,y_test))
```

[47]

```
Best Score: 0.6861702127659575
```

Decision Tree	Tu ned a DecisionTreeClassifier using RandomizedSearchCV. First, initialized a base DecisionTreeClassifier and defined the hyperparameters and their possible values using the param_dist dictionary. The key hyperparameters included criterion, splitter, max_depth, min_samples_split, min_samples_leaf, max_features, and min_impurity_decrease. I used RandomizedSearchCV to search for the best hyperparameter combination, evaluating the model's performance.

```
# Define the hyperparameters and their possible values for tuning
param_grid = {
    'criterion': ['gini', 'entropy'],
    'splitter': ['best', 'random'],
    'max_depth': [None, 2, 4, 6, 8,10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': [None, 'sqrt', 'log2'],
    'min_impurity_decrease': [0.0, 0.1, 0.2],
    'ccp_alpha': [0.0, 0.1 ,0.2]
}
```

```
# Initialize RandomizedSearchCV with DecisionTreeClassifier
random_search = RandomizedSearchCV(estimator=dt_classifier,
                                   param_distributions=param_grid,
                                   scoring='accuracy',
                                   cv=3,
                                   n_iter=100,
                                   random_state=42)
```

```
random_search.fit(X_train, y_train)
RandomizedSearchCV(cv=3,estimator=DecisionTreeClassifier(),n_iter=100,
                   param_distributions={'ccp_alpha':[0.0, 0.1, 0.2],
                                        'criterion':['gini','entropy'],
                                        'max_depth':[None, 2, 4, 6, 8, 10],
                                        'max_features':[None,'sqrt','log2'],
                                        'min_impurity_decrease':[0.0, 0.1, 0.2],
                                        'min_samples_leaf':[1, 2, 4],
                                        'min_samples_split':[2, 5, 10],
                                        'splitter': ['best', 'random']},
random_state=42, scoring='accuracy')

print("Best Parameters:", random_search.best_params_)

print("Best Score:", random_search.best_score_)
```

Final Model Selection Justification (2 Marks):

Final Model	Reasoning

Bagging Classifier	Bagging Classifier model is chosen for its robustness in handling complex datasets and its ability to mitigate overfitting while providing high predictive accuracy.
	
	`print(train_score)`
	`print(test_score)`
	
	`{'Decision Tree': 0.7697017752521915, 'Random Forest Calssifier': 0.8147658693449628, 'Bagging Classifier': 0.650059`
	`{'Decision Tree': 0.6041473943879124, 'Random Forest Classifier': 0.6387438879344522, 'Bagging Classifier': 0.683796`
	
	Above all the models Bagging classifier have the highest accuracy among all the models.