

Model Development Phase Template

Date	10 July 2024
Team ID	739861
Project Title	Frappe Activity :Mobile Phone Activity Classification
Maximum Marks	10 Marks

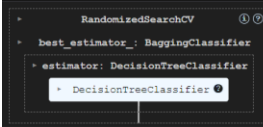
Initial Model Training Code, Model Validation and Evaluation Report

The initial model training code will be showcased in the future through a screenshot. The model validation and evaluation report will include a summary and training and validation performance metrics for multiple models, presented through respective screenshots.

Initial Model Training Code (5 marks):

Paste the screenshot of the model training code

Model Validation and Evaluation Report (5 marks):

Model	Summary	Training and Validation Performance Metrics
Model 1	<p>bagging classifier, short for bootstrap aggregating classifier, is an ensemble machine learning technique designed to improve the stability and accuracy of machine learning algorithms. It works by generating multiple versions of a predictor and using these to get an aggregated prediction. Here's how it works:</p>	<pre>bagging_classifier = BaggingClassifier(estimator=base_estimator) 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)</pre>  <pre>> print("Best Parameters:",random_search.best_params_) print("Best Score:",random_search.best_score_) Best Parameters: {'n_estimators': 100, 'max_samples': 0.7, 'max_features': 0.7, 'bootstrap_features': False, 'bootstrap': False} Best Score: 0.64986229719911 + Code + Markdown print("Best Score:",random_search.score(X_test,y_test)) Best Score: 0.6789018104922929 bc_final=random_search print(classification_report(y_test,y_pred,digits=4)) precision recall f1-score support 0 0.5476 0.6109 0.5803 68666 1 0.6879 0.4761 0.5328 68827 2 0.6668 0.6992 0.6826 68715 accuracy 0.6057 0.6017 0.6019 181508 macro avg 0.6058 0.6039 0.6011 181508 weighted avg 0.6058 0.6039 0.6011 181508</pre>

Model 2

Decision tree classifier model commonly include accuracy, precision, recall, F1 score which help assess the model's prediction accuracy and generalizability

```
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]
}
```

```
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']},
                  scoring='accuracy')
```

```
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']},
                  scoring='accuracy')

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

Best Parameters: {'splitter': 'best', 'min_samples_split': 2, 'min_samples_leaf': 2, 'min_impurity_decrease': 0.0, 'max_features': None, 'max_depth': None}
Best Score: 0.592460372900
```

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

Best Score: 0.6038665697546364

```
dt_final=random_search
```

```
from sklearn.metrics import classification_report, confusion_matrix
print(classification_report(y_test, y_pred, digits=4))
```

	precision	recall	f1-score	support
0	0.5946	0.6167	0.6055	60466
1	0.6352	0.5323	0.5792	60427
2	0.6813	0.7659	0.7211	60715
accuracy			0.6385	181608
macro avg	0.6370	0.6383	0.6353	181608
weighted avg	0.6371	0.6385	0.6354	181608

Model 3

Random forest classifier model often encompass accuracy, precision, recall, F1 score to measure its prediction quality and robustness.

```
rf_final=RandomForestClassifier()

rf_final.fit(X_train,y_train)
accuracy_score(rf_final.predict(X_test),y_test)

0.6401810492929827

print("Best score :",rf_final.score(X_test,y_test))

Best score : 0.6397019955068058

print(classification_report(y_test,y_pred,digits=4))
```

	precision	recall	f1-score	support
0	0.5474	0.6359	0.5883	60466
1	0.6029	0.4761	0.5320	60427
2	0.6668	0.6992	0.6826	60715
accuracy			0.6039	181608
macro avg	0.6057	0.6037	0.6010	181608
weighted avg	0.6058	0.6039	0.6011	181608