SMARTBRIDGE Let's Bridge the Gap a Veranda Enterprise

Smart Internz

Model Optimization and Tuning Phase Report

Date	25 June 2025
Team ID	SWTID1750155746
Project Title	Human Resource Management: Predicting Employee Promotions using Machine Learning
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):

<table>
<tr><th>Model</th><th>Tuned Hyperparameters</th><th>Optimal Values</th></tr>
<tr><td>Decision Tree</td><td>dt_classifier = DecisionTreeClassifier()

param_grid = {
    'criterion' : ['gini','entropy'],
    'splitter' : ['best','random'],
    'max_depth' : [None, 10, 20, 30, 40, 50],
    'min_samples_split' : [2, 5, 10],
    'min_samples_leaf' : [1,2,4]
}

# Perform GridSearchCV
grid_search = GridSearchCV(dt_classifier, param_grid, cv=5, scoring='accuracy')
grid_search.fit(x_train, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_dt_model = grid_search.best_estimator_</td><td># Make predictions with the best model
y_pred = best_dt_model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')

Optimal Hyperparameters: {'criterion': 'entropy', 'max_depth': 50, 'min_samples_leaf': 1, 'min_samples_split': 2, 'splitter
Accuracy on Test Set: 0.9279351150113017</td></tr>
<tr><td>Random

Forest</td><td>rf_classifier = RandomForestClassifier(random_state=42, n_jobs=-1)

param_dist = {
    'n_estimators': randint(50, 200),
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': randint(2, 11),
    'min_samples_leaf': randint(1, 5),
}</td><td># Make predictions with the best model
y_pred = best_rf_model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')

Optimal Hyperparameters: {'criterion': 'entropy', 'max_depth': 30, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators':
Accuracy on Test Set: 0.9455524531312326</td></tr>
</table>

KNN

```python
knn_classifier = KNeighborsClassifier()

param_dist = {
    'n_neighbors': [3, 5, 7, 9],
    'weights': ['uniform', 'distance'],
    'p': [1, 2]
}

random_search = RandomizedSearchCV(
    knn_classifier,
    param_distributions=param_dist,
    n_iter=10,
    cv=3,
    scoring='accuracy',
    n_jobs=-1,
    verbose=1,
    random_state=42
)
```

```python
# Make predictions with the best model
y_pred = best_knn_model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')
```

```
Optimal Hyperparameters: {'weights': 'distance', 'p': 1, 'n_neighbors': 3}
Accuracy on Test Set: 0.9075255950006648
```

XG Boost

```python
xgb_classifier = XGBClassifier(random_state=42, n_jobs=-1, verbosity=0)


param_dist = {
    'n_estimators': randint(50, 150),
    'max_depth': randint(3, 10),
    'learning_rate': uniform(0.01, 0.3),
    'subsample': uniform(0.7, 0.3),
    'colsample_bytree': uniform(0.7, 0.3),
    'gamma': uniform(0, 5)
}
```

```python
# Make predictions with the best model
y_pred = best_knn_model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

print(f'Optimal Hyperparameters: {best_params}')
print(f'Accuracy on Test Set: {accuracy}')
```

```
Optimal Hyperparameters: {'colsample_bytree': np.float64(0.8199582915145766), 'gamma': np.float64(0.2333283160680714),
Accuracy on Test Set: 0.9075255950006648
```

Performance Metrics Comparison Report (2 Marks):

Model	Optimized Metric
Decision Tree	(see below)

Decision Tree

```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.92      0.93     15065
           1       0.92      0.94      0.93     15019

    accuracy                           0.93     30084
   macro avg       0.93      0.93      0.93     30084
weighted avg       0.93      0.93      0.93     30084
```

```python
print(confusion_matrix(y_test, y_pred))
```

```
[[13818  1247]
 [  921 14098]]
```

<table>
<tr>
<td>Random Forest</td>
<td><pre>
print(classification_report(y_test, y_pred))

              precision   recall f1-score   support

           0   0.95   0.94   0.95   15065
           1   0.94   0.95   0.95   15019

    accuracy   0.95   30084
   macro avg   0.95   0.95   0.95   30084
weighted avg   0.95   0.95   0.95   30084

print(confusion_matrix(y_test, y_pred))

[[14187   878]
 [760 14259]]
</pre></td>
</tr>
<tr>
<td>KNN</td>
<td><pre>
print(classification_report(y_test, y_pred))

              precision   recall f1-score   support

           0   0.95   0.86   0.90   15065
           1   0.87   0.95   0.91   15019

    accuracy   0.91   30084
   macro avg   0.91   0.91   0.91   30084
weighted avg   0.91   0.91   0.91   30084

print(confusion_matrix(y_test, y_pred))

[[12975 2090]
 [692 14327]]
</pre></td>
</tr>
<tr>
<td>XG Boost</td>
<td><pre>
print(classification_report(y_test, y_pred))

              precision   recall f1-score   support

           0   0.95   0.86   0.90   15065
           1   0.87   0.95   0.91   15019

    accuracy   0.91   30084
   macro avg   0.91   0.91   0.91   30084
weighted avg   0.91   0.91   0.91   30084

print(confusion_matrix(y_test, y_pred))

[[12975 2090]
 [692 14327]]
</pre></td>
</tr>
</table>

Final Model Selection Justification (2 Marks):

Final Model	Reasoning
Random Forest	The Random Forest model was selected for its superior performance, exhibiting the highest accuracy during hyperparameter tuning. Its ability to handle complex feature interactions, reduce overfitting, and deliver reliable predictions aligns with the project objectives, justifying its selection as the final model.