SMARTBRIDGE
Let's Bridge the Gap
a Veranda Enterprise

Smart Internz

Model Optimization and Tuning Phase Template

Date	04 October 2024
Team ID	LTVIP2024TMID24922
Project Title	Rainfall Prediction 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>Random Forest</td>
<td>
<pre><code>rf=RandomForestClassifier()
rf.fit(X_train_res,y_train_res)</code></pre>
<pre><code># Number of trees in random forest
n_estimators=[int(x) for x in np.linspace(start=200,stop=2000,num=10)]

# Number of features to consider at every split
max_features=['auto','sqrt', 'log2']

# Maximum number of levels in tree
max_depth=[int(x) for x in np.linspace(10,1000,10)]

# Minimum number of samples required to split a node
min_samples_split=[2,5,10,14]

# Minimum number of samples required at each leaf node
min_samples_leaf=[1,2,4,6,8]

# Create the random grid
random_grid={'n_estimators':n_estimators,
             'max_features':max_features,
              'max_depth':max_depth,
              'min_samples_split':min_samples_split,
              'min_samples_leaf':min_samples_leaf,
              'criterion':['entropy','gini']}
print(random_grid)</code></pre>
</td>
<td>
<pre><code>from sklearn.metrics import accuracy_score   Import "sklearn.metrics" could not
y_pred = best_random_grid.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print('Accuracy score {}'.format(accuracy_score(y_test,y_pred)))
print('Classification report {}'.format(classification_report(y_test,y_pred)))</code></pre>
</td>
</tr>
</table>

<table>
<tr>
<td>Decision Tree</td>
<td>

```
# Setup the parameters and distributions to sample from: param_dist
param_dist = {"max_depth": [3, None],
              "max_features": randint(1, 9),
              "min_samples_leaf": randint(1, 9),
              "criterion": ["gini", "entropy"]}

# Instantiate a Decision Tree classifier: tree
tree = DecisionTreeClassifier()

# Instantiate the RandomizedSearchCV object: tree_cv
tree_cv = RandomizedSearchCV(tree, param_dist, cv=5)

# Fit it to the data
tree_cv.fit(X_train,y_train)

# Print the tuned parameters and score
print("Tuned Decision Tree Parameters: {}".format(tree_cv.best_params_))
print("Best score is {}".format(tree_cv.best_score_))
```

</td>
<td>

```
from sklearn.metrics import accuracy_score
y_pred_tree = tree_cv.predict(X_test)
print(confusion_matrix(y_test,y_pred_tree))
print('Accuracy score {}'.format(accuracy_score(y_test,y_pred_tree)))
print('Classification report {}'.format(classification_report(y_test,y_pred_tree)))
```

</td>
</tr>
<tr>
<td>K-Neighbors Classifier</td>
<td>

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train_res, y_train_res)
```

</td>
<td>

```
y_pred4 = knn.predict(X_test)
print(confusion_matrix(y_test,y_pred4))
print(accuracy_score(y_test,y_pred4))
print(classification_report(y_test,y_pred4))
```

</td>
</tr>
<tr>
<td>Logestic Regression</td>
<td>

```
logreg = LogisticRegression()
logreg.fit(X_train_res, y_train_res)
```

</td>
<td>

```
y_pred2 = logreg.predict(X_test)
print(confusion_matrix(y_test,y_pred2))
print(accuracy_score(y_test,y_pred2))
print(classification_report(y_test,y_pred2))
```

</td>
</tr>
<tr>
<td>XGBoost</td>
<td>

```
# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Various learning rate parameters
learning_rate = ['0.05','0.1', '0.2','0.3','0.5','0.6']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
#Subssample parameter values
subsample=[0.7,0.6,0.8]
# Minimum child weight parameters
min_child_weight=[3,4,5,6,7]


# Create the random grid
random_grid = {'n_estimators': n_estimators,
               'learning_rate': learning_rate,
               'max_depth': max_depth,
               'subsample': subsample,
               'min_child_weight': min_child_weight}

print(random_grid)
```

</td>
<td>

```
from sklearn.metrics import accuracy_score
y_predict = xg_random.predict(X_test)
print(confusion_matrix(y_test,y_predict))
print('Accuracy score {}'.format(accuracy_score(y_test,y_predict)))
print('Classification report {}'.format(classification_report(y_test,y_predict)))
```

</td>
</tr>
<tr>
<td>SVC</td>
<td>

```
svc = SVC()
svc.fit(X_train_res, y_train_res)
```

</td>
<td>

```
y_pred5 = svc.predict(X_test)
print(confusion_matrix(y_test,y_pred5))
print(accuracy_score(y_test,y_pred5))
print(classification_report(y_test,y_pred5))
```

</td>
</tr>
</table>

<table>
<tr>
<td>CatBoost</td>
<td># Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Various learning rate parameters
learning_rate = [0.05,0.1, 0.2,0.3,0.5,0.6]
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
#Subssample parameter values
subsample=[0.7,0.6,0.8]
# Minimum child samples parameters
min_child_samples=[3,4,5,6,7]

# Create the random grid
random_grid = {'n_estimators': n_estimators,
'learning_rate': learning_rate,
'max_depth': max_depth,
'subsample': subsample,
'min_child_samples': min_child_samples}

print(random_grid)</td>
<td>cat_random.best_params_

{'subsample': 0.6,
'n_estimators': 300,
'min_child_samples': 5,
'max_depth': 5,
'learning_rate': 0.05}

best_random_grid=cat_random.best_estimator_

from sklearn.metrics import accuracy_score
y_pred = best_random_grid.predict(X_test)
print(confusion_matrix(y_test,y_pred))
print('Accuracy score {}'.format(accuracy_score(y_test,y_pred)))
print('Classification report {}'.format(classification_report(y_test,y_pred)))</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>print('Classification report {}'.format(classification_report(y_test,y_pred)))

Classification report precision recall f1-score support
0 0.89 0.89 0.89 1897
1 0.58 0.58 0.58 503
accuracy 0.82 2400
macro avg 0.74 0.73 0.73 2400
weighted avg 0.82 0.82 0.82 2400

print(confusion_matrix(y_test,y_pred))

[[1690 207]
[213 290]]</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>print('Classification report {}'.format(classification_report(y_test,y_pred_tree)))

Classification report precision recall f1-score support
0 0.80 0.99 0.89 1892
1 0.71 0.09 0.17 508
accuracy 0.80 2400
macro avg 0.75 0.54 0.53 2400
weighted avg 0.78 0.80 0.73 2400

print(confusion_matrix(y_test,y_pred_tree))

[[1872 20]
[460 48]]</td>
</tr>
</table>

K-Neighbors Classifier	print(classification_report(y_test,y_pred4)) precision recall f1-score support 0 0.91 0.77 0.83 22717 1 0.46 0.72 0.56 6375 accuracy 0.76 29092 macro avg 0.68 0.74 0.70 29092 weighted avg 0.81 0.76 0.77 29092 print(confusion_matrix(y_test,y_pred4)) [[17409 5308] [1808 4567]]
Logistic Regression	print(classification_report(y_test,y_pred2)) precision recall f1-score support 0 0.92 0.77 0.84 22717 1 0.48 0.76 0.59 6375 accuracy 0.77 29092 macro avg 0.70 0.77 0.71 29092 weighted avg 0.82 0.77 0.78 29092 print(confusion_matrix(y_test,y_pred2)) [[17439 5278] [1507 4868]]
XGBoost	print('Classification report {}'.format(classification_report(y_test,y_predict))) Classification report precision recall f1-score support 0 0.87 0.93 0.90 1874 1 0.68 0.52 0.59 526 accuracy 0.84 2400 macro avg 0.78 0.73 0.75 2400 weighted avg 0.83 0.84 0.83 2400 print(confusion_matrix(y_test,y_predict)) [[1745 129] [250 276]]

<table>
<tr>
<td>SVC</td>
<td>

```
print(classification_report(y_test,y_pred5))
```

```
              precision    recall  f1-score   support

           0       0.92      0.77      0.84      1887
           1       0.47      0.74      0.57       513

    accuracy                           0.76      2400
   macro avg       0.69      0.76      0.71      2400
weighted avg       0.82      0.76      0.78      2400
```

```
print(confusion_matrix(y_test,y_pred5))
```

```
[[1453  434]
 [ 132  381]]
```

</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 high accuracy during hyperparameter tuning. Its ability to handle complex relationships, minimize overfitting, and optimize predictive accuracy aligns with project objectives, justifying its selection as the final model