



Model Optimization and Tuning Phase Template

Date	July 2024
Team ID	739859
Project Title	Auto insurance fraud detection using michine learning
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





#importing the library for grid search from sklearn.model selection import GridSearchCV

The 'lr_param_grid' specifies different values for regularization strength (C), solvers (solver), and penalty types (penalty). GridSearchCV (lr_cv) is employed with 5-fold cross-validation (cv=5), evaluating model performance based on accuracy (scoring="accuracy"). The process uses all available CPU cores (n_jobs=-1) for parallel processing and provides verbose output (verbose=True) to track progress.

Logistic Regression





The parameter grid (rfc_param_grid) for hyperparameter tuning. It specifies different values for the number of trees (n_estimators), splitting criterion (criterion), maximum depth of trees (max_depth), and maximum number of features considered for splitting (max_features). GridSearchCV (rfc_cv) is employed with 3-fold cross-validation (cv=3), evaluating model performance based on accuracy (scoring="accuracy").

Random Forest

```
from sklearn.neighbors import KNeighborsclassifier
model = KNeighborsclassifier()
model = model.fit(X_Train, Y_Train)
pred = model.predict(X_Test)

print('Accuracy:', accuracy_score(Y_Test, pred))
print('\n classification report:\n', classification_report(Y_Test, pred))
print('\n confusion matrix:\n', confusion_matrix(Y_Test, pred))

Accuracy: 0.71

classification report:
    precision recall f1-score support

0 0.29 0.15 0.19 48
1 0.77 0.89 0.82 152

accuracy 0.71 200
macro avg 0.53 0.52 0.51 200
weighted avg 0.65 0.71 0.67 200

confusion matrix:

OPPSA12 · OKarthn [for: 7 41]
2024/07/22 18:39 [17 135]]
```

The (params) define a grid for hyperparameter tuning of the XGBoost Classifier (XGBClassifier), including min_child_weight, gamma, colsample_bytree, and max_depth. The XGBClassifier is configured with a learning rate of 0.5, 100 estimators, using a binary logistic regression objective, and utilizing 3 threads for processing. GridSearchCV (xg_cv) is used with 5-fold cross-validation (cv=5), refitting the best model (refit=True), evaluating based on accuracy (scoring="accuracy")

XGBoost

```
model = XGBClassifier()
             model = model.fit(X_Train, Y_Train)
{x}
            pred = model.predict(X_Test)
            print('Accuracy:', accuracy_score(Y_Test, pred))
            print('\n classification report:\n', classification_report(Y_Test, pred))
print('\n confusion matrix:\n', confusion_matrix(Y_Test, pred))
Accuracy: 0.815
             classification report:
                   precision recall f1-score \sqrt{support}
                          0.60 0.67 0.63 48
0.89 0.86 0.88 152
                                                  0.81
                                                             200
            ()
             confusion matrix:
```





The parameters (params) define a grid for hyperparameter tuning of the Decision Tree Classifier (DecisionTreeClassifier), including max_depth, min_samples_leaf, and criterion ('gini' or 'entropy'). GridSearchCV (dec_cv) is used with 5-fold cross-validation (cv=5), evaluating model performance based on accuracy (scoring="accuracy")

Decision Tree

The parameters (params) define a grid for hyperparameter tuning of the Decision Tree Classifier (DecisionTreeClassifier), including max_depth, min_samples_leaf, and criterion ('gini' or 'entropy'). GridSearchCV (dec_cv) is used with 5-fold cross-validation (cv=5), evaluating model performance based on accuracy (scoring="accuracy")

Ridge Classifier

RIDGE-CLASSIFIER-HYPER PARAMETER TUNNING

```
#finding the grid search cv for ridge classifier
rg=RidgeClassifier(random_state=42)
params={
        'alpha':(np.logspace(-8,8,100))
}
rg_cv=GridSearchCV(rg,param_grid=params,cv=5)
rg_cv.fit(x_train,y_train)
##finding the grid search cv for ridge classifier
params=42

##finding the grid search cv for ridge classifier
rg=RidgeClassifier
rg=Ri
```

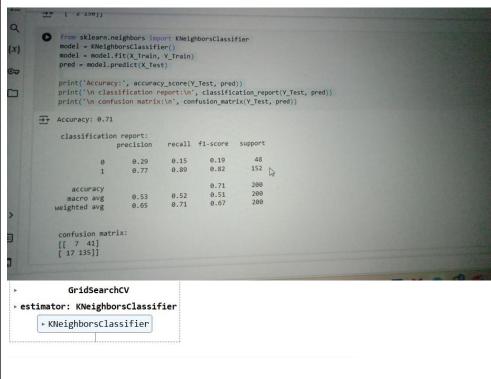
▶ GridSearchCV
 ▶ estimator: RidgeClassifier
 ▶ RidgeClassifier





The parameters (params) define a grid for hyperparameter tuning of the K-Nearest Neighbors Classifier (KNeighborsClassifier), including n_neighbors, weights ('uniform' or 'distance'), and metric ('minkowski', 'euclidean', or 'manhattan'). GridSearchCV (knn_cv) is used with 5-fold cross-validation (cv=5), evaluating model performance based on accuracy (scoring="accuracy")

K- Nearest Neighbors



Final Model Selection Justification (2 Marks):





Final Model		Reasoning							
	Random Forest model is chosen for its robustness in handling complex datasets and its ability to mitigate overfitting while providing high predictive accuracy. Name Accuracy f1_score Recall Precision								
	0	Logistic Regression	67.90	64.68	59.16	71.35			
	1	Decision Tree Classifier	73.88	66.60	52.41	91.32			
Random Forest	2	Random Forest	74.68	66.70	51.03	96.24			
	3	K-Nearest Nieghbors	74.56	71.57	64.44	80.48			
	4	Xgboost	74.18	68.61	56.78	86.67			
	5	Ridge Classifier	68.39	63.91	56.32	73.87			
	Above all the models Random Forest model have the highest accuracy among all the models.								