



## **Model Optimization and Tuning Phase Template**

Date	July 2024
Team ID	Team-739662
Project Title	Ecommerce Shipping Prediction using Machine 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

LOGISTIC REGRESSION HYPER PARAMETER TUNNING

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





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**

```
#finding the grid search cv for xgboost

params={
    'min_child_weight':[10,20],
    'gamma':[1.5,2.0,2.5],
    'colsample_bytree':[0.6,0.8,0.9],
    'max_depth':[4,5,6]
}

xg=XGBClassifier(learning_rate=0.5,n_estimators=100,objective='binary:logistic',nthreads=3)
    xg_cv=GridSearchCV(xg,param_grid=params,cv=5,refit=True,scoring="accuracy",n_jobs=-1,verbose=3)
    xg_cv.fit(x_train,y_train)

Fitting 5 folds for each of 54 candidates, totalling 270 fits
    //usr/local/lib/python3.10/dist-packages/xgboost/core.py:160: UserWarning: [14:07:26] WARNING: /works
Parameters: { "nthreads" } are not used.

warnings.warn(smsg, UserWarning)

* GridSearchCV

* estimator: XGBClassifier

* XGBClassifier

* XGBClassifier
```

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**

DECISION TREE CLASSIFIER-HYPER PARAMETER TUNNING

XGBOOST CLASSIFIER-HYPER PARAMETER TUNNIG

```
[68] #finding grid search cv for decision tree classifier
dec=DecisionTreeClassifier(nandom_state=42)
params={ 'max_depth': [2, 3, 5, 10, 20],
    'min_samples_leaf': [5, 10, 20, 50, 100],
    'criterion': ['gini', 'entropy']
}
dec_cv=GridSearchcV(dec,param_grid=params,cv=5,n_jobs=-1,scoring="accuracy",verbose=3)
dec_cv.fit(x_train,y_train)

→ Fitting 5 folds for each of 50 candidates, totalling 250 fits

    * GridSearchcV

* estimator: DecisionTreeClassifier

    * DecisionTreeClassifier

* DecisionTreeClassifier
```





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)
```

FridgeClassifier
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-HYPER PARAMETER TUNNING

#### 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 provihigh predictive accuracy.							
						Precision		
Random Forest	0	Logistic Regression	67.90	64.68	59.16	71.35		
	1	Decision Tree Classifier	73.88	66.60	52.41	91.32		
	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 accuration among all the models.							