Model Optimization and Tuning Phase Template

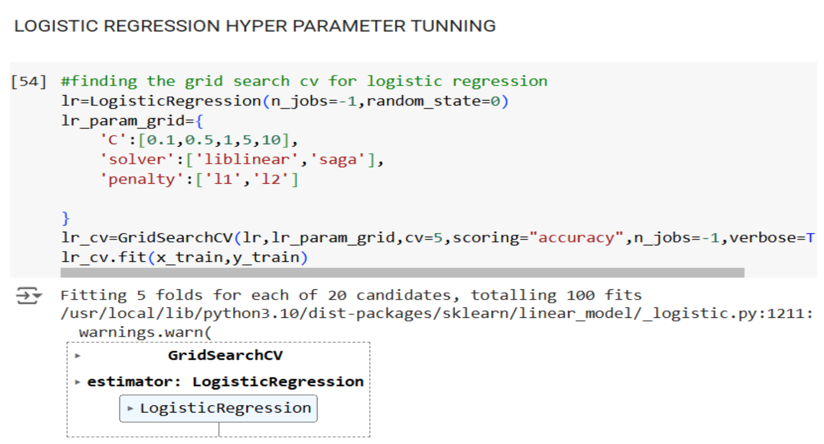
Date	25 June 2024
Team ID	team-739671
Project Title	Ecommerce Shipping Prediction using Machine Learning
Maximum Marks	10 Marks

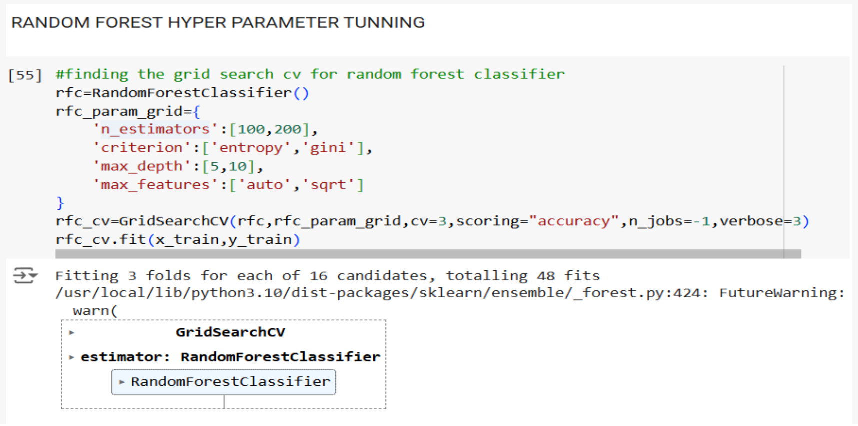
Model Optimization and Tuning Phase

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

Logistic Regression	`#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.
	
	

Random Forest	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"). 

LOGISTIC REGRESSION HYPER PARAMETER TUNNING

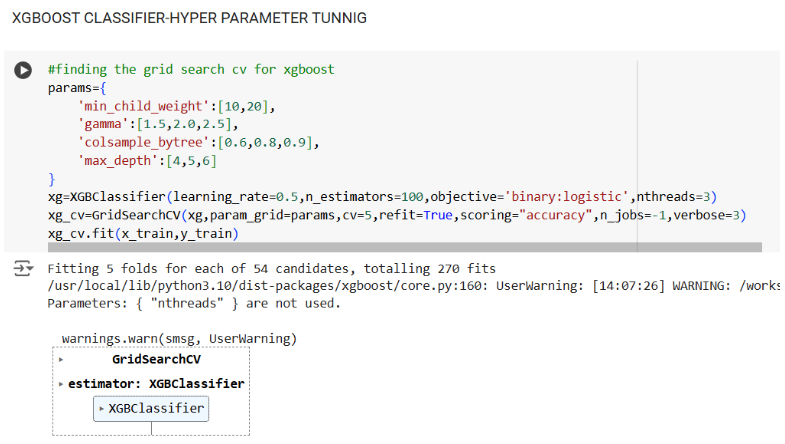
```
[54] #finding the grid search cv for logistic regression
     lr=LogisticRegression(n_jobs=-1,random_state=0)
     lr_param_grid={
         'C':[0.1,0.5,1,5,10],
         'solver':['liblinear','saga'],
         'penalty':['l1','l2']

     }
     lr_cv=GridSearchCV(lr,lr_param_grid,cv=5,scoring="accuracy",n_jobs=-1,verbose=T
     lr_cv.fit(x_train,y_train)
```

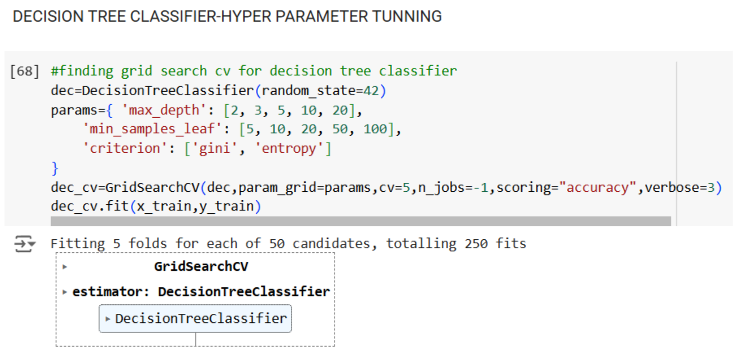
```
Fitting 5 folds for each of 20 candidates, totalling 100 fits
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:1211:
  warnings.warn(
```

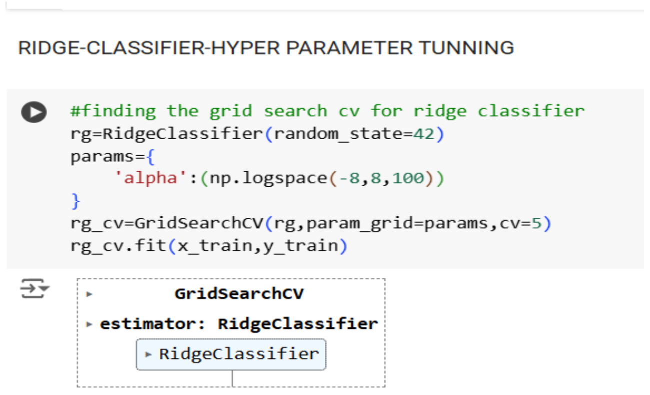
RANDOM FOREST HYPER PARAMETER TUNNING

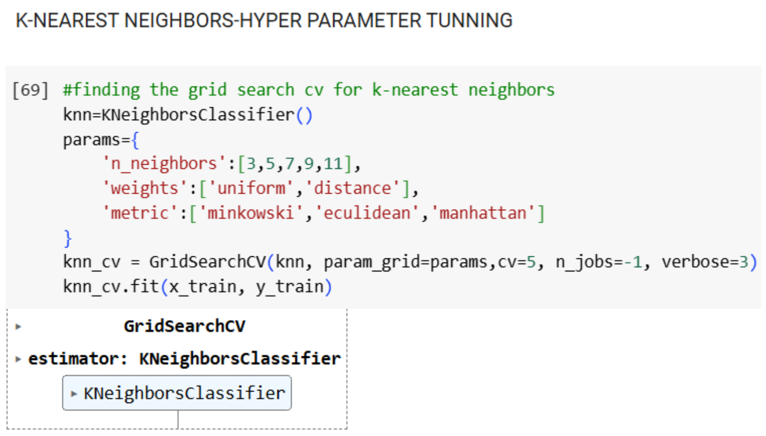
```
[55] #finding the grid search cv for random forest classifier
     rfc=RandomForestClassifier()
     rfc_param_grid={
         'n_estimators':[100,200],
         'criterion':['entropy','gini'],
         'max_depth':[5,10],
         'max_features':['auto','sqrt']
     }
     rfc_cv=GridSearchCV(rfc,rfc_param_grid,cv=3,scoring="accuracy",n_jobs=-1,verbose=3)
     rfc_cv.fit(x_train,y_train)
```

```
Fitting 3 folds for each of 16 candidates, totalling 48 fits
/usr/local/lib/python3.10/dist-packages/sklearn/ensemble/_forest.py:424: FutureWarning:
  warn(
```

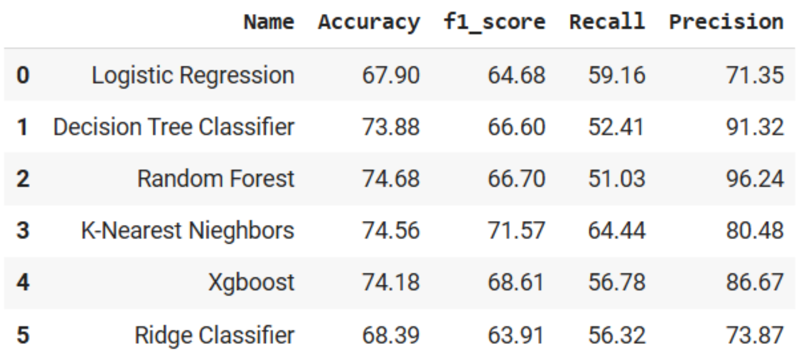
XGBoost	The (params) define a grid for hyperparameter tuning of the XGBoost Classifier (XGBClassifier), including min_child_weight, gamma, colsample_bytree, and max_depth. The XGBClassifier is configured with a learning rate of 0.5, 100 estimators, using a binary logistic regression objective, and utilizing 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 CLASSIFIER-HYPER PARAMETER TUNNIG
	`#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`

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")
	
	DECISION TREE CLASSIFIER-HYPER PARAMETER TUNNING
	`[68] #finding grid search cv for decision tree classifier`
	`dec=DecisionTreeClassifier(random_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`

Ridge Classifier	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-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)`
	 `▸ GridSearchCV`
	 `▸ estimator: RidgeClassifier`
	 `▸ RidgeClassifier`

K- Nearest Neighbors	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
	 `[69] #finding the grid search cv for k-nearest neighbors`
	 `knn=KNeighborsClassifier()`
	 `params={`
	 `    'n_neighbors':[3,5,7,9,11],`
	 `    'weights':['uniform','distance'],`
	 `    'metric':['minkowski','eculidean','manhattan']`
	 `}`
	 `knn_cv = GridSearchCV(knn, param_grid=params,cv=5, n_jobs=-1, verbose=3)`
	 `knn_cv.fit(x_train, y_train)`
	 `▸ GridSearchCV`
	 `▸ estimator: KNeighborsClassifier`
	 `▸ KNeighborsClassifier`

Final Model Selection Justification (2 Marks):

Final Model	**Reasoning**
Random Forest	Random Forest model is chosen for its robustness in handling complex datasets and its ability to mitigate overfitting while providing high predictive accuracy.  Above all the models Random Forest model have the highest accuracy among all the models.

	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
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