



### **Model Optimization and Tuning Phase Report**

Date	21 June 2024
Team ID	739705
Project Title	Eudaimonia Engine: Machine Learning Delving into Happiness Classification
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):**

Model	Tuned Hyperparameters	Optimal Values
Decision Tree	### Superparameter Tuning for Decision Tree Model #### Superparameter Tuning param_prig_dt = { ### Superparameter	from sklearn.metrics import accuracy_score  # Assuming you have defined and trained your classifier model classifier = dt classifier, fit(x_rain, y_train)  # Evaluate the performance of the tuned model y_pred = classifier.predict(x_test) accuracy = accuracy_score(y_test, y_pred) print(f'Optimal Hyperparameters: (best_param)') print(f'Accuracy on test set: {accuracy}')  Optimal Hyperparameters: ('entropy', None, 10, 1) Accuracy on test set: 0.7241379310344828





```
from sklearn.metrics import accuracy_score
                                                                       #Hyperparameter Tuning for Random Forest Model
                                                                       #Define Random forest Tree Classifier
                                                                                                                                                                                                       # Assuming you have defined and trained your classifier model
                                                                       rf = RandomForestClassifier()
                                                                                                                                                                                                       classifier = rf
                                                                                                                                                                                                       classifier.fit(x_train, y_train)
                                                                       #Hyperparemeter Tuning
   Random
                                                                      # Define the parameter grid for hyperparameter tuning
                                                                                                                                                                                                       # Evaluate the performance of the tuned model
                                                                      param_grid = {
                                                                                                                                                                                                      y_pred = classifier.predict(x_test)
   Forest
                                                                               'n_estimators': [50, 100, 200],
                                                                                                                                                                                                       accuracy = accuracy_score(y_test, y_pred)
                                                                                                                                                                                                       print(f'Optimal Hyperparameters: {best_param}')
                                                                               'criterion': ['gini', 'entropy'],
                                                                                                                                                                                                       print(f'Accuracy on test set: {accuracy}')
                                                                               'max_depth': [None, 10, 20, 30], 'min_samples_split': [2, 5, 10],
                                                                                                                                                                                                     Optimal Hyperparameters: ('entropy', None, 10, 1)
Accuracy on test set: 0.5862068965517241
                                                                               'min_samples_leaf': [1, 2, 4]
                                                              #Hyperparameter Tuning For KNN Model
from sklearn.model_selection import GridSearchCV
from sklearn.meighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
                                                                                                                                                                                                          # Evaluate the performance of the tuned model
                                                                                                                                                                                                          accuracy = accuracy_score(y_test, y_pred)
   KNN
                                                                                                                                                                                                          print(f'Optimal Hyperparameters: {best_params}')
                                                              # Define the kNN classifier
knn = KNeighborsClassifier()
                                                                                                                                                                                                          print(f'Accuracy on test set: {accuracy}')
                                                              # Define the hyperparameters to tune
parameters = {
                                                                                                                                                                                                          Optimal Hyperparameters: {'metric': 'manhattan', 'n_neighbors': 7, 'weights': 'uniform'}
                                                                     meters = {
'n_neighbors': [3, 5, 7, 9], # Number of neighbors to consider
'weights': ['uniform', 'distance'], # Weight function used in prediction
'metric': ['euclidean', 'manhattan'] # Distance metric to use for the tree
                                                                                                                                                                                                          Accuracy on test set: 0.5517241379310345
                                                             # Perform grid search with cross-validation
grid_search = GridSearchCV(knn, parameters, cv=5)
grid_search.fit(x_train, y_train)
                                                              best_params = grid_search.best_params_
                                                            Whyperparameter Tuning For SVC Model
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
                                                            # Define the SVC classifier
svc = SVC()
                                                             # Define the hyperparameters to tune
parameters = (
'C': [0:1, 1, 10],  # Regularization parameter
'kernel': ['linear', 'rbf'], # Kernel type
'gamma': ['soale', 'auto']  # Kernel coefficient
                                                                                                                                                                                                         # Evaluate the performance of the tuned model
   SVC
                                                                                                                                                                                                         accuracy = accuracy_score(y_test, y_pred)
                                                                                                                                                                                                         print(f'Optimal Hyperparameters: {best_params}')
                                                                                                                                                                                                         print(f'Accuracy on test set: {accuracy}')
                                                             # Get the best hyperparameters
best_params = grid_search.best_params_
                                                                                                                                                                                                         Optimal Hyperparameters: {'C': 1, 'gamma': 'auto', 'kernel': 'rbf'}
                                                            # Use the best model for prediction
best_model = grid_search.best_estimator_
v pred = best model.predict(x test)
                                                                                                                                                                                                         Accuracy on test set: 0.4827586206896552
                                                                                                                                                                                                             # Evaluate the performance of the tuned model
                                                                                                                                                                                                             accuracy = accuracy_score(y_test, y_pred)
                                                             #Hyperparameter Tuning For Logistic Model
from sklearn.model_meletion import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
                                                                                                                                                                                                             print(f'Optimal Hyperparameters: {best params}')
Logistic Model
                                                                                                                                                                                                             print(f'Accuracy on test set: {accuracy}')
                                                              # Define the Logistic Regression classified log_reg = LogisticRegression()
                                                                                                                                                                                                             Optimal Hyperparameters: {'C': 2, 'max_iter': 100, 'penalty': '12', 'solver': 'liblinear'}
Accuracy on test set: 0.4827586206696552
                                                             # Oefine the hyperparameters to tune
parameters = {
    penalty'; ['11', '12'],
    'C': [0:1, 0:5, 1, 2, 5, 10],
    'solver': ['iinlinear', 'sage'],
    'max_iter': [100, 200, 300]
                                                              # Perform grid search with cross-validation
grid_search = GridSearchCV(log_reg, parameters, cv=5)
grid_search.fit(x_train, y_train)
                                                              # Get the best hyperparameters
best_params = grid_search.best_params_
                                                              # Use the best model for prediction
best_model = grid_search.best_estimator_
y_pred = best_model.predict(x_test)
```





## **Performance Metrics Comparison Report (2 Marks):**

	#Classification						
	from sklearn.metrics import classification_report						
	cr=classificat	ion_report(	y_test,y_	_pred)			
	print(cr)						
	i	precision	recall	f1-sc	ore su	pport	
		0.55	0.26				
	0	0.56			.43	14 15	
Random Forest	1	0.55	0.75	6	.63	15	
Random i orest	accuracy			0	.55	29	
	macro avg	0.55	0.55	0	.53	29	
	weighted avg	0.55	0.55	0	.54	29	
	#Confusion Mat	rix					
	from sklearn.m		rt confus	sion ma	trix		
	cm=confusion_m						
	print(cm)	· · · · · · · · · · · · · · · · · · ·					
	[[ 5 9] [ 4 11]]						
	[ - 11]]						
	#Classificati	on Report					
	from sklearn.	metrics in	mport cl			_report	
	from sklearn. cr=classifica	metrics in	mport cl			_report	
	from sklearn.	metrics in	mport cl			_report	
	from sklearn. cr=classifica	metrics in	mport cl rt(y_tes	st,y_p	red)	report suppor	t j
	from sklearn. cr=classifica print(cr)	metrics in tion_report precision 0.29	mport cl rt(y_tes n rec	all -	red) f1-score	suppor	4
	from sklearn. cr=classifica print(cr)	metrics in tion_report	mport cl rt(y_tes n rec	all	red) f1-score	suppor	4
	from sklearn. cr=classifica print(cr)  0 1	metrics in tion_report precision 0.25 0.45	mport cl rt(y_tes n rec	all -	red) f1-score	support	4
	from sklearn. cr=classifica print(cr)	metrics in tion_report precision 0.29 0.43	mport cl rt(y_tes n rec	all -	f1-score 0.1 0.50	support 14 19 19 19 19 19 19 19 19 19 19 19 19 19	4 5
IZNINI	from sklearn. cr=classifica print(cr)  0 1 accuracy	metrics in tion_report precision 0.29 0.43	mport cl rt(y_tes n rec 5 0 3 0	all -	f1-score 0.1 0.5 0.3 0.3	support 10 11 12 13 14 12 14 15 16 17 17 18 18 18 18 18 18 18 18 18 18 18 18 18	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg	metrics in tion_report precision 0.29 0.43	mport cl rt(y_tes n rec 5 0 3 0	all	f1-score 0.1 0.5 0.3 0.3	support 10 11 12 13 14 12 14 15 16 17 17 18 18 18 18 18 18 18 18 18 18 18 18 18	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg	metrics in tion_report precision 0.2! 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0	all	f1-score 0.1 0.5 0.3 0.3	support 10 11 12 13 14 12 14 15 16 17 17 18 18 18 18 18 18 18 18 18 18 18 18 18	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg	metrics in tion_repoil precision 0.2: 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0	all	f1-score 0.1 0.5 0.3 0.3	support 10 1: 3 2: 4 2: 5 2:	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg	metrics in tion_report precision 0.2: 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0 4 0 mport cc	st,y_p rall 0.14 0.60 0.37 0.38	f1-score 0.1 0.5 0.3 0.3	support 10 1: 3 2: 4 2: 5 2:	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0  accuracy macro avg weighted avg  #Confusion Ma from sklearn.	metrics in tion_report precision 0.2: 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0 4 0 mport cc	st,y_p rall 0.14 0.60 0.37 0.38	f1-score 0.1 0.5 0.3 0.3	support 10 1: 3 2: 4 2: 5 2:	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg  #Confusion Ma from sklearn. cm=confusion_ print(cm)  [[ 2 12]	metrics in tion_report precision 0.2: 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0 4 0 mport cc	st,y_p rall 0.14 0.60 0.37 0.38	f1-score 0.1 0.5 0.3 0.3	support 10 1: 3 2: 4 2: 5 2:	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0  accuracy macro avg weighted avg  #Confusion Ma from sklearn. cm=confusion_ print(cm)	metrics in tion_report precision 0.2: 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0 4 0 mport cc	st,y_p rall 0.14 0.60 0.37 0.38	f1-score 0.1 0.5 0.3 0.3 0.3	support 10 1: 3 2: 4 2: 5 2:	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg  #Confusion Ma from sklearn. cm=confusion_ print(cm)  [[ 2 12]	metrics in tion_report precision 0.2: 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0 4 0 mport cc	st,y_p rall 0.14 0.60 0.37 0.38	f1-score 0.1 0.5 0.3 0.3 0.3	support 10 1: 3 2: 4 2: 5 2:	4 5 9
KNN	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg  #Confusion Ma from sklearn. cm=confusion_ print(cm)  [[ 2 12]	metrics in tion_report precision 0.2: 0.4: 0.34	mport cl rt(y_tes n rec 5 0 3 0 4 0 mport cc	st,y_p rall 0.14 0.60 0.37 0.38	f1-score 0.1 0.5 0.3 0.3 0.3	support 10 1: 3 2: 4 2: 5 2:	4 5 9

Model	Optimized Metric							
	#Classification from sklearn.me cr=classificati print(cr)	trics impo			eport			
	p	recision	recall	f1-score	support			
	0 1	0.73 0.67	0.57 0.80					
Decision Tree	accuracy macro avg weighted avg	0.70 0.70	0.69 0.69					
	#Confusion Matr from sklearn.me cm=confusion_ma print(cm)	trics impo						
	[[ 8 6] [ 3 12]]							





	#Classification		100	9.50		
	from sklearn.me				report	
	cr=classificati print(cr)	.on_report(	y_test,y_	prea)		
SVC	F	recision	recall	f1-score	supp	ort
B V C	9	0.25	0.14	0.18		14
	1	0.43				15
	accuracy			0.38		29
	macro avg weighted avg	0.34	0.37	0.34		29
	weighted dvg	0.54	0.50	0.55		
	#Confusion Matr					
	from sklearn.me				×	
	<pre>cm=confusion_ma print(cm)</pre>	trix(y_tes	t,y_pred)	D.		
	[[ 2 12]					
	[6 9]]					
	#Classification	as Bonost				
Logistic Model	#Classificati from sklearn. cr=classifica print(cr)	metrics i	mport cl			port
Logistic Model	from sklearn. cr=classifica	metrics i tion_repo	mport cl rt(y_tes	t,y_pred		port
Logistic Model	from sklearn. cr=classifica	metrics i tion_repo	mport cl rt(y_tes n rec	t,y_pred		support
Logistic Model	from sklearn. cr=classifica print(cr)	metrics in tion_repo	mport cl rt(y_tes n rec 5 0	t,y_pred	core	support 14
Logistic Model	from sklearn. cr=classifica print(cr)	metrics in tion_repo precision	mport cl rt(y_tes n rec 5 0	t,y_pred) all f1-s .14 .60	0.18 0.50	support 14 15
Logistic Model	from-sklearn. cr=classifica print(cr) 0 1 accuracy	metrics in tion_repo precision 0.2 0.4	mport cl rt(y_tes n rec 5 0 3 0	t,y_pred) all f1-s .14 .60	0.18 0.50	support 14 15 29
Logistic Model	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg	metrics in tion_repo precision 0.2 0.4	mport cl rt(y_tes n rec 5 0 3 0	t,y_pred all f1-s .14 .60	0.18 0.50 0.38 0.34	support 14 15 29 29
Logistic Model	from-sklearn. cr=classifica print(cr) 0 1 accuracy	metrics in tion_repo precision 0.2 0.4	mport cl rt(y_tes n rec 5 0 3 0	t,y_pred all f1-s .14 .60	0.18 0.50	support 14 15 29 29
Logistic Model	from sklearn. cr=classifica print(cr)  a accuracy macro avg weighted avg  #Confusion Ma	precision  0.2: 0.4  0.3  0.3	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0	all f1-9 .14 .60 .37	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg  #Confusion Ma from sklearn.	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	from-sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg  #Confusion Ma from sklearn. cm=confusion_	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	from sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg  #Confusion Ma from sklearn.	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	#Confusion Mafrom sklearn.  #curacy macro avg weighted avg  #confusion Mafrom sklearn.  cm=confusion_print(cm)	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	from-sklearn. cr=classifica print(cr)  0 1 accuracy macro avg weighted avg  #Confusion Ma from sklearn. cm=confusion_	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	#Confusion Mafrom sklearn.  #Confusion Mafrom sklearn.  #confusion_print(cm)  [[ 2 12]	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	#Confusion Mafrom sklearn.  #Confusion Mafrom sklearn.  #confusion_print(cm)  [[ 2 12]	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	#Confusion Mafrom sklearn.  #Confusion Mafrom sklearn.  #confusion_print(cm)  [[ 2 12]	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29
Logistic Model	#Confusion Mafrom sklearn.  #Confusion Mafrom sklearn.  #confusion_print(cm)  [[ 2 12]	metrics intion_repo  precision  0.22  0.4  0.3  0.3  trix metrics in	mport cl rt(y_tes n rec 5 0 3 0 4 0 4 0 mport co	all f1-s .14 .60 .37 .38	0.18 0.50 0.38 0.34 0.35	support 14 15 29 29

# **Final Model Selection Justification (2 Marks):**

Final Model	Reasoning
Decision Tree Model	The Decision Tree 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.