Homework 4 - Model Building

Table of hyperparameter exploration

Model	Iteration	penalty	C	solver	l1_ratio	max_iter	Train	Test	OOT
		Parameters					Average FDR at 3%		
Logistic Regression	1	elasticnet	1	saga	0.5	1500	0.624867	0.615815	0.49162
	2	l1	1	liblinear	None	2500	0.617388	0.624968	0.475978
	3	l2	10	sag	None	2000	0.618721	0.61436	0.483799
	4	l2	10	newton-cg	None	2000	0.616405	0.621681	0.482123
	5	l2	0.1	newton-cg	None	2000	0.621024	0.617238	0.475978
	6	l1	0.1	saga	None	2500	0.625478	0.625934	0.493296
	7	l1	100	saga	None	2500	0.617217	0.619642	0.490503
	8	l2	10	lbfgs	None	4500	0.616221	0.628301	0.483799

Model	Iteration	criterion	splitter	max_depth	min_samples_split	min_samples_leaf	Train	Test	OOT
Decision Tree	1	gini	best	8	40	20	0.755479	0.717918	0.416201
	2	gini	best	10	100	40	0.797988	0.745372	0.493855
	3	entropy	best	10	100	50	0.830217	0.750007	0.434078
	4	gini	random	30	120	60	0.728195	0.690516	0.396648
	5	gini	best	10	100	50	0.800511	0.746124	0.439665
	6	entropy	random	10	100	50	0.705223	0.707514	0.270391
	7	gini	best	8	120	60	0.764513	0.71055	0.507263
	8	gini	best	8	90	50	0.768904	0.720124	0.507263
	9	gini	best	28	50	10	0.921006	0.724473	0.409497

Model	Iteration	boostrap	criterion	n_estimators	max_depth	min_samples_split	min_samples_leaf	max_features	Train	Test	OOT
Random Forest	1	TRUE	gini	100	8	120	60	sqrt	0.809151	0.769379	0.512291
	2	TRUE	entropy	100	8	120	60	sqrt	0.807182	0.765139	0.550279
	3	TRUE	gini	100	8	80	40	sqrt	0.813494	0.775923	0.479888
	4	TRUE	entropy	100	10	100	50	sqrt	0.837427	0.777048	0.558101
	5	TRUE	entropy	50	15	100	50	sqrt	0.855228	0.783621	0.52067
	6	TRUE	gini	50	None	100	50	log2	0.852952	0.793234	0.482682
	7	FALSE	gini	150	25	100	30	log2	0.870716	0.79027	0.412849
	8	FALSE	gini	150	35	120	20	sqrt	0.883037	0.803042	0.40838
	9	TRUE	gini	100	7	100	20	sqrt	0.803193	0.774611	0.537989
	10	TRUE	gini	200	27	100	1	sqrt	0.999836	0.812862	0.471508

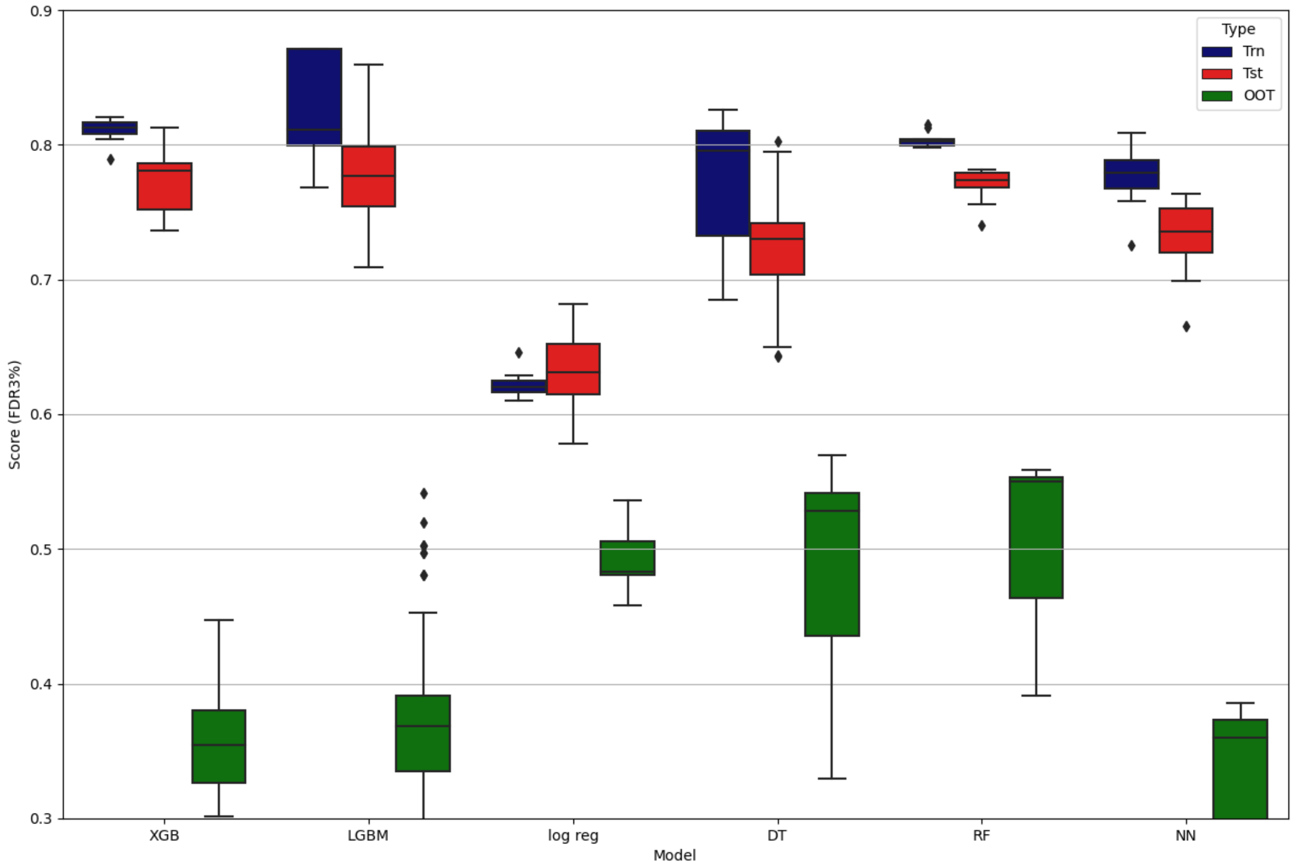
Model	Iteration	activation	solver	learning_rate	alpha	learning_rate_init	batch_size	hidden_layer_size	max_iter	Train	Test	OOT
Nueral Network	1	relu	sgd	constant	0.01	None	auto	(100,)	200	0.619258	0.614538	0.329609
	2	relu	sgd	constant	0.001	None	auto	(100,)	500	0.618472	0.622582	0.319553
	3	relu	sgd	constant	0.0001	0.001	auto	(100,)	200	0.782887	0.72274	0.348045
	4	relu	sgd	constant	0.0001	0.01	auto	(200,)	500	0.773537	0.716709	0.773537
	5	relu	sgd	adaptive	0.0001	0.001	auto	(100,)	200	0.775993	0.730024	0.327933
	6	relu	sgd	adaptive	1	0.03	auto	(200,)	300	0.537184	0.539292	0.260121

Model	Iteration	boosting_type	n_estimator	max_depth	learning_rate	num_leaves	subsample	colsample_bytree	Train	Test	OOT
LGBM Classifier	1	gbdt	200	23	0.05	30	0.5	1	0.994853	0.827198	0.34581
	2	gbdt	80	3	0.05	40	0.5	1	0.807898	0.772485	0.386592
	3	gbdt	100	-1	0.1	31	1	1	0.997568	0.793298	0.31676
	4	gbdt	500	7	0.05	50	0.8	0.8	1	0.817303	0.309497
	5	gbdt	90	3	0.05	30	0.5	1	0.808564	0.761725	0.377654

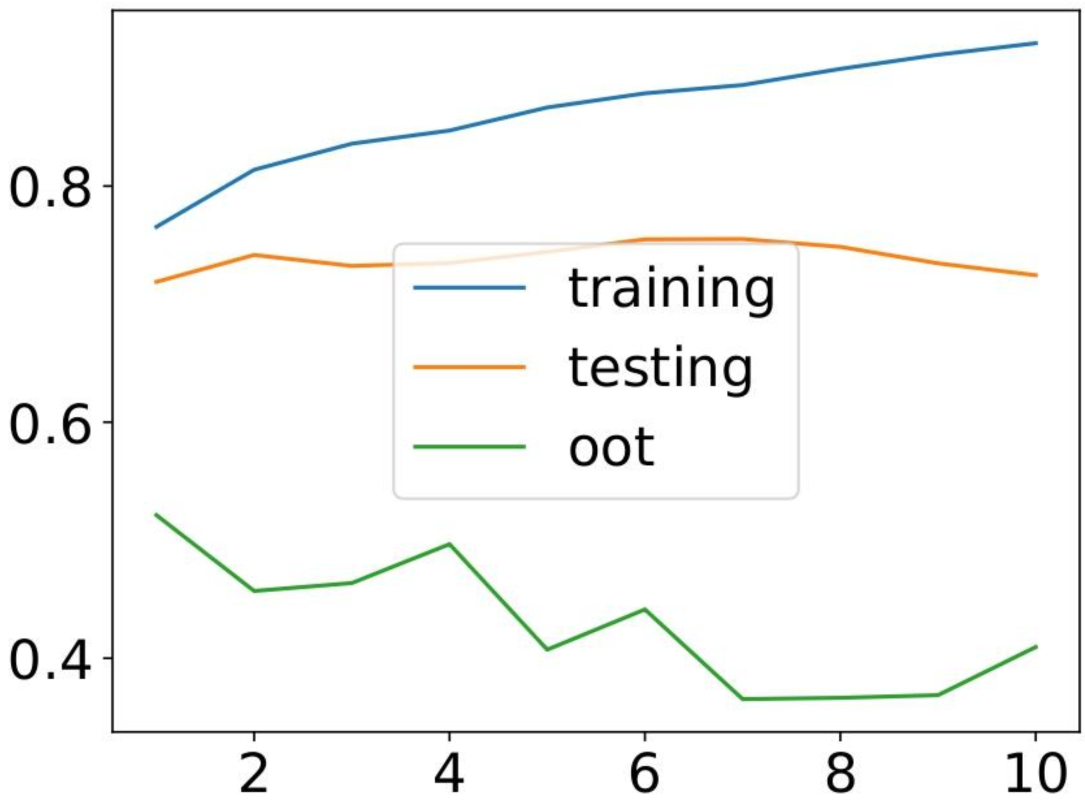
Model	Iteration	booster	n_estimator	max_depth	min_child_weight	eta	tree_method	subsample	colsample_bytree	Train	Test	OOT
XGBoost	1	gbtree	15	4	7	0.4	auto	1	0.5	0.810716	0.772675	0.357542
	2	gbtree	100	6	1	0.3	auto	1	1	1	0.816645	0.29162
	3	gbtree	15	4	7	0.4	approx	1	0.5	0.792273	0.764359	0.380447
	4	gbtree	15	8	5	0.05	auto	0.8	0.8	0.776215	0.758858	0.553631
	5	dart	20	10	5	0.08	auto	0.8	0.8	0.804949	0.751255	0.52514

Akansha Lalwani (A59019733)

Box Plot - best model parameters for each model



Decision Tree - DecisionTreeClassifier()



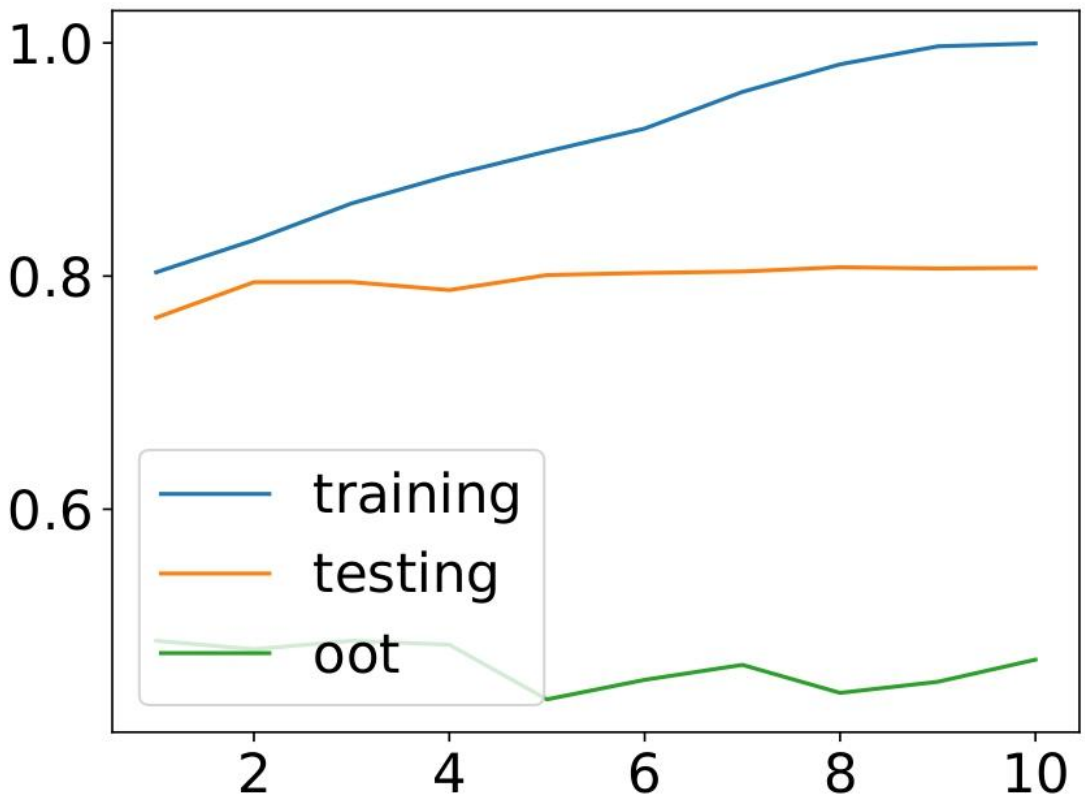
I achieved overfitting on my DecisionTreeClassifier model by increasing its complexity:

- **Increasing the depth of the tree:** By increasing the depth of the tree from 8 to 20, my model becomes more complex and can capture more information from the data, including noise and outliers. I increased the depth of the tree by setting and initial value and then increasing the max_depth parameter to a large value.

- **Reducing the minimum samples required to split an internal node:** The min_samples_split parameter specifies the minimum number of samples required to split an internal node. By reducing this parameter to 10, I allowed the tree to split internal nodes that have fewer samples, which lead to overfitting.

- **Reducing the minimum samples required to be at a leaf node:** The min_samples_leaf parameter specifies the minimum number of samples required to be at a leaf node. By reducing this parameter to 1, I allowed the tree to create leaves with fewer samples, which again lead to overfitting.

Random Forest - RandomForestClassifier()



Some of these parameters which I tuned to overfit the RandomForestClassifier model are:

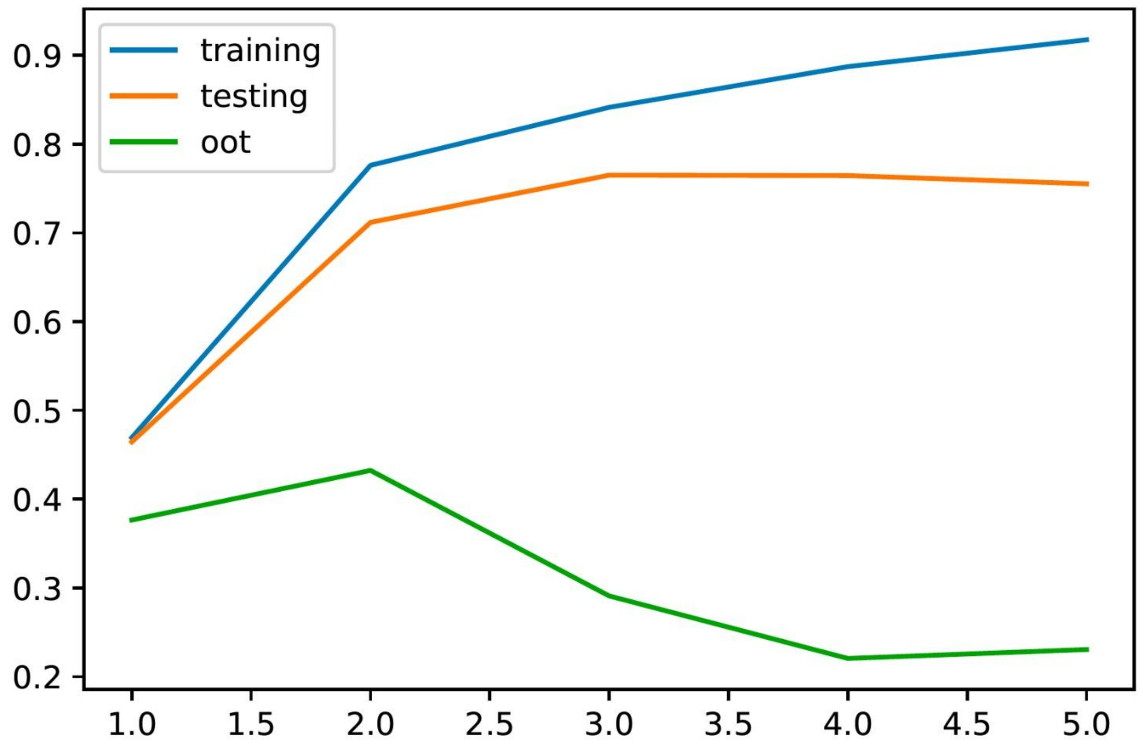
- **Increasing the number of trees:** Increasing the number of trees in a random forest model may lead to overfitting as the model can become too complex and start to memorize the training data.

- **Decreasing the minimum samples per leaf:** Setting a lower value for the minimum samples per leaf may result in overfitting as the model may fit the training data too closely. I decreased the min_samples_split from 100 to 20.

- **Increasing the maximum depth of each tree:** Increasing the maximum depth of each tree can cause the model to become more complex and may start to overfit the training data. I increased the max_depth from 7 to 27.

- **Setting the bootstrap parameter to false:** If the bootstrap parameter is set to false, the model will not perform a random sampling of the training data, which can lead to overfitting.

__Neural Network - MLPClassifier()__



For overfitting my NN model, I used the following techniques:

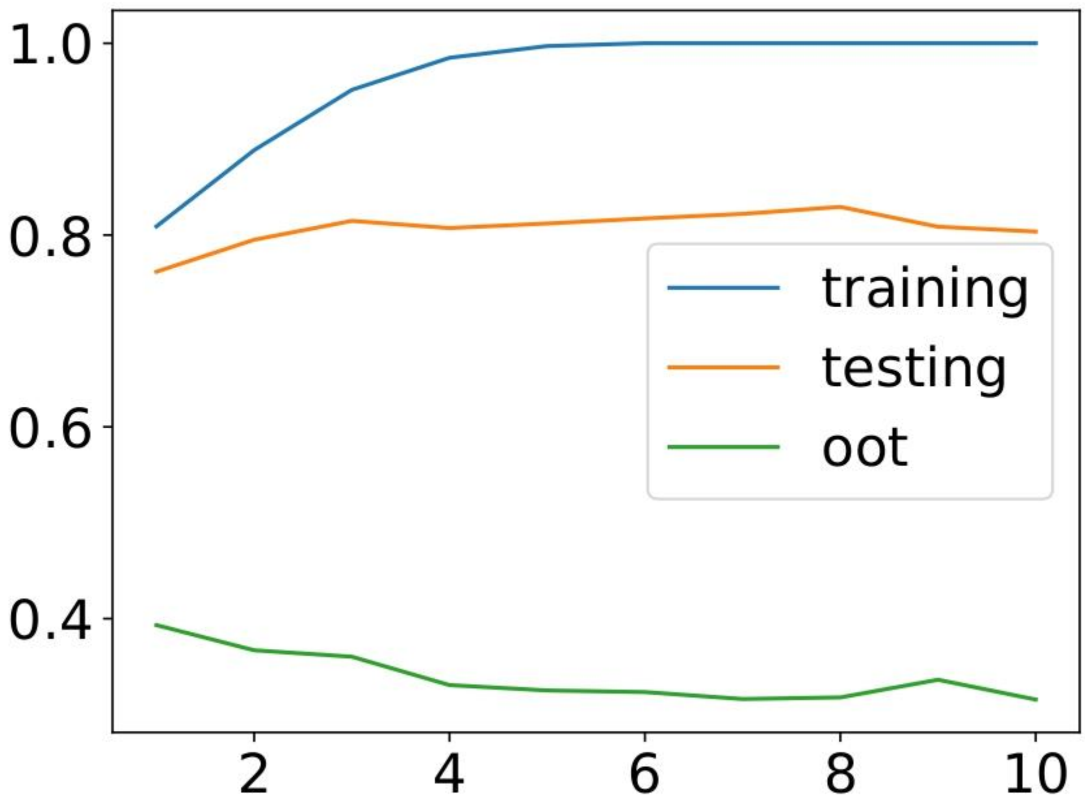
- **Increase the number of hidden layers and/or the number of neurons in each layer:** By adding more layers or neurons, the model becomes more complex and can better fit the training data. I started with (10,10) and went up to (100,100)

- **Decrease the regularization strength:** Regularization is used to prevent overfitting by adding a penalty term to the loss function that discourages large weights or activations. By reducing the strength of regularization, my model was able to fit the training data more closely and overfit.

- **Train for more epochs:** The training process involves updating the model parameters to minimize the loss function, and training can be stopped when the validation loss stops decreasing. However, continuing to train for more epochs led to overfitting.

- **Increase the complexity of the input data:** By adding noise or augmenting the data, my model is exposed to more variations in the input data and can better fit the training data.

LGBM - LGBMClassifier()



Some of these parameters which I tuned to overfit LGBMClassifier model are:

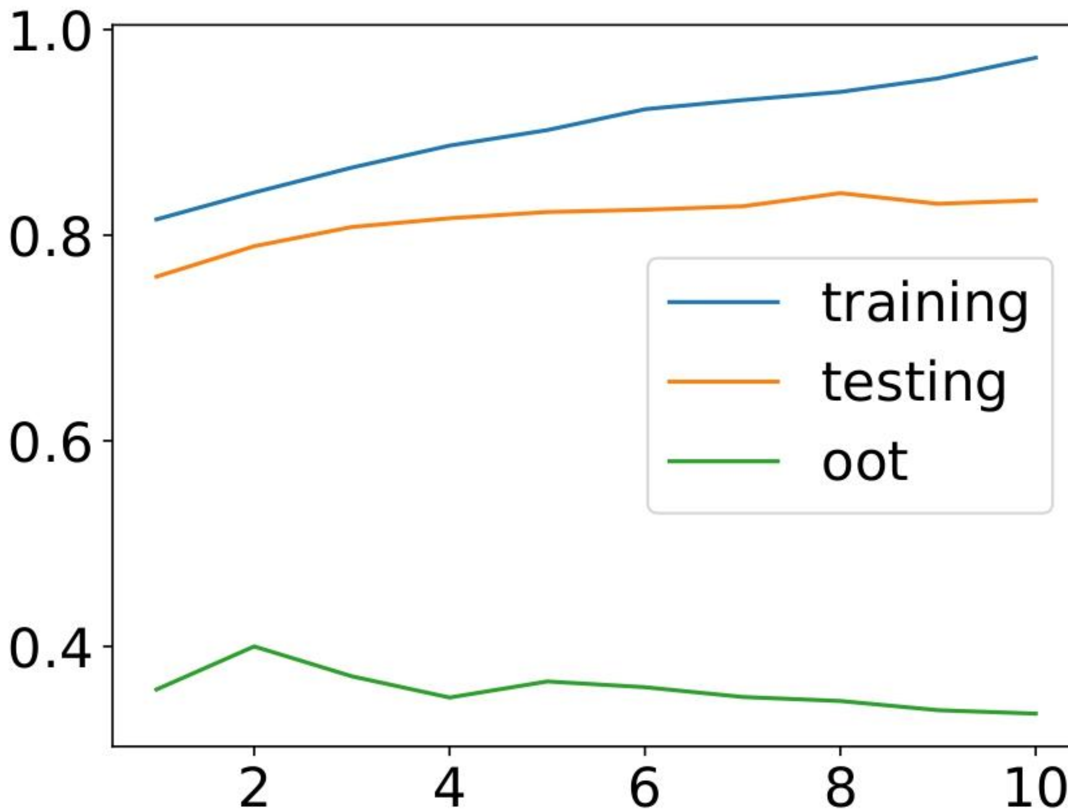
- **Increase the number of estimators:** I set a high value for the n_estimators parameter, which increases the number of boosting iterations. This lead to overfitting as the model starts to fit the noise in the data. I increased the n_estimators from 80 to 200.

- **Decrease learning rate:** Lowering the learning_rate parameter cause the model to learn more aggressively from each iteration, leading to overfitting.

- **Increase max depth:** I set the max_depth to a high value that allows the model to create more complex decision boundaries and can lead to overfitting. I increased the max depth from 3 to 20.

- **Decrease subsample ratio:** Lowering the subsample parameter reduces the number of samples the model uses for each boosting iteration. This caused the model to start fitting to the noise in the data.

XGB - XGBClassifier()



Overfitting XGBClassifier can be achieved by using high values for its hyperparameters:

- **n_estimators:** Increased the number of trees in the ensemble by setting a high value for n_estimators. This increases the complexity of the model and makes it more prone to overfitting. I increased the n_estimators from 15 to 100.

- **max_depth:** Increased the maximum depth of each decision tree by setting a high value for max_depth. This allows the model to learn more complex relationships in the data, but may also lead to overfitting. I increased the max depth from 4 to 10.

- **learning_rate:** Decreased the learning rate by setting a low value for learning_rate. This causes the model to take smaller steps during gradient descent and require more iterations to converge.

- **subsample:** Decreased the subsample ratio by setting a low value for the subsample. This causes the model to use a smaller random subset of the training data for each tree, which can increase variance and hence, overfitting.

- **min_child_weight:** Increased min_child_weight parameter from 1 to 7.