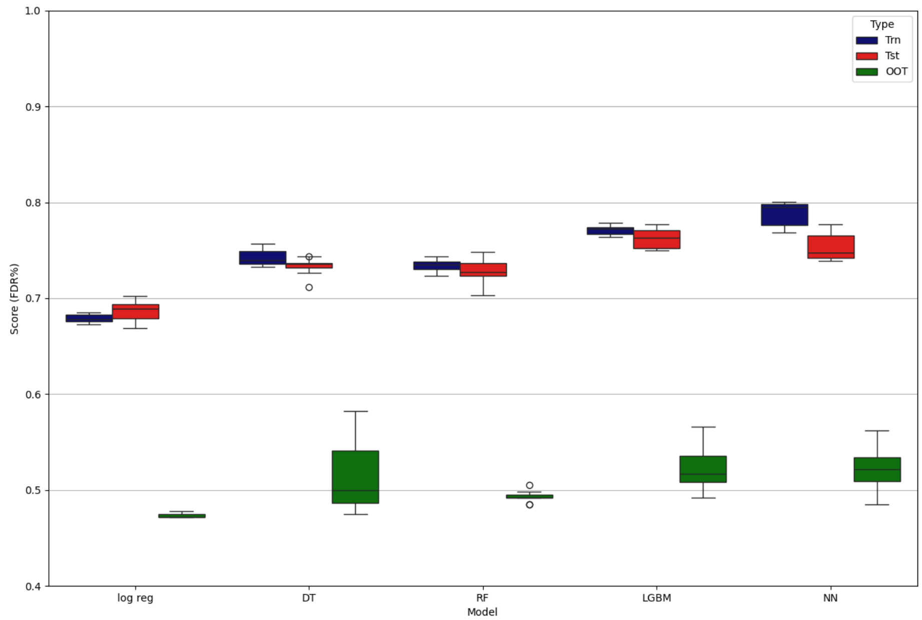
Part 1: Model exploration table

Model	Parameters							Avg FDR at 3%		
Logistic Regression	Iteration	penalty	c	solver		l1_ratio		Train	Test	OOT
	1	l2	1	lbfgs		None		0.682	0.678	0.466
	2	l2	0.1	lbfgs		None		0.681	0.684	0.466
	3	l1	1	saga		None		0.680	0.682	0.468
	4	l1	0.1	saga		None		0.677	0.690	0.471
	5	l2	0.01	lbfgs		None		0.680	0.685	0.471
	6	elasticnet	1	saga		1		0.682	0.680	0.467
	7	elasticnet	0.1	saga		0.4		0.682	0.680	0.469
	8	elasticnet	0.01	saga		0.8		0.682	0.681	0.475
Decision Tree	Iteration	Criterion	splitter	Max_depth	Min_samples_split	min_samples_leaf	max_features	Train	Test	OOT
	1	gini	best	3	20	5	None	0.661	0.656	0.427
	2	gini	best	5	25	7	None	0.705	0.688	0.474
	3	gini	best	10	20	5	None	0.798	0.730	0.525
	4	gini	best	10	20	200	None	0.722	0.708	0.504
	5	gini	best	20	180	90	None	0.745	0.717	0.529
	6	gini	best	10	190	90	None	0.741	0.727	0.527
Random Forest	Iteration	n_estimators	criterion	Max_depth	Min_samples_split	min_samples_leaf	boostrap	Train	Test	OOT
	1	100	gini	None	20	5	TRUE	0.933	0.808	0.569
	2	100	gini	None	25	5	TRUE	0.919	0.799	0.565
	3	100	gini	None	180	90	TRUE	0.747	0.740	0.498
	4	100	gini	10	180	90	TRUE	0.733	0.727	0.492
	5	300	gini	15	65	30	TRUE	0.804	0.777	0.570
LightGBM	Iteration	subsample	max_depth	learning_rate		n_estimators		Train	Test	OOT
	1	0.8	-1	0.1		100		0.985	0.813	0.514
	2	0.8	3	0.05		100		0.772	0.767	0.537
	3	0.8	4	0.05		100		0.813	0.779	0.553
	4	0.9	4	0.1		150		0.883	0.800	0.532
	5	0.7	4	0.1		100		0.855	0.794	0.529
Neural Network	Iteration	hidden_layer	activation	alpha	learning rate	learning_rate_init	solver	Train	Test	OOT
	1	(1,1)	relu	0.0001	constant	0.001	adam	0.687	0.689	0.480
	2	(10,)	relu	0.001	constant	0.001	adam	0.720	0.715	0.485
	3	(100,)	relu	0.001	constant	0.001	adam	0.788	0.756	0.518
	4	(100,)	tanh	0.001	constant	0.001	adam	0.810	0.767	0.507
	5	(200,)	relu	0.001	constant	0.001	adam	0.804	0.766	0.524

Part 2: Box plot



The Decision Tree model's train and test box plots indicate minimal overfitting; however, the Out of Time (OOT) range from 0.47 to 0.58 is quite broad.

The Random Forest model's train and test plots show consistency, suggesting no significant overfitting. Nevertheless, the OOT performance is relatively low, failing to exceed 0.5, which could indicate underfitting or poor generalization on unseen data.

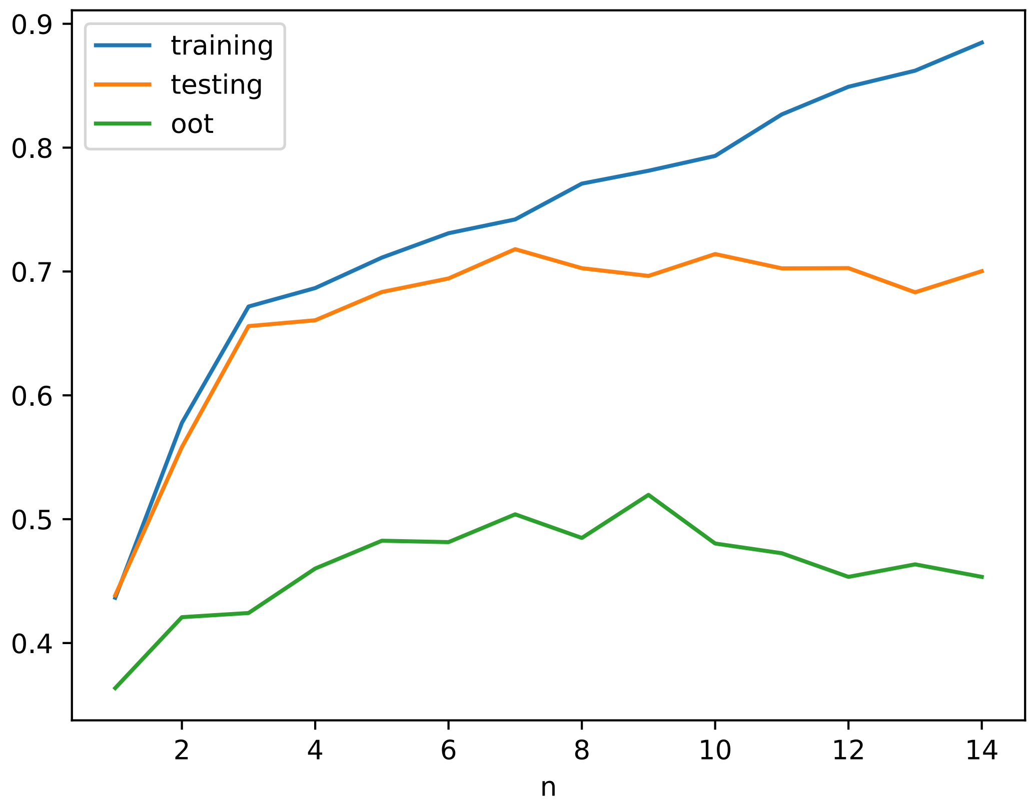
The LightGBM model's train and test box plots demonstrate stable and moderate fitting. The OOT range is narrower, from 0.48 to 0.55, tighter compared to the Decision Tree model.

The Neural Network model's train and test box plots reveal overfitting. Despite numerous tuning attempts, this remains the best result I could achieve.

Overall, I would choose **LightGBM** as the best model due to its robust performance on both train and test data and its smaller variability in OOT performance.

Part 3: Four “complexity plots”

1. Single Decision Tree:

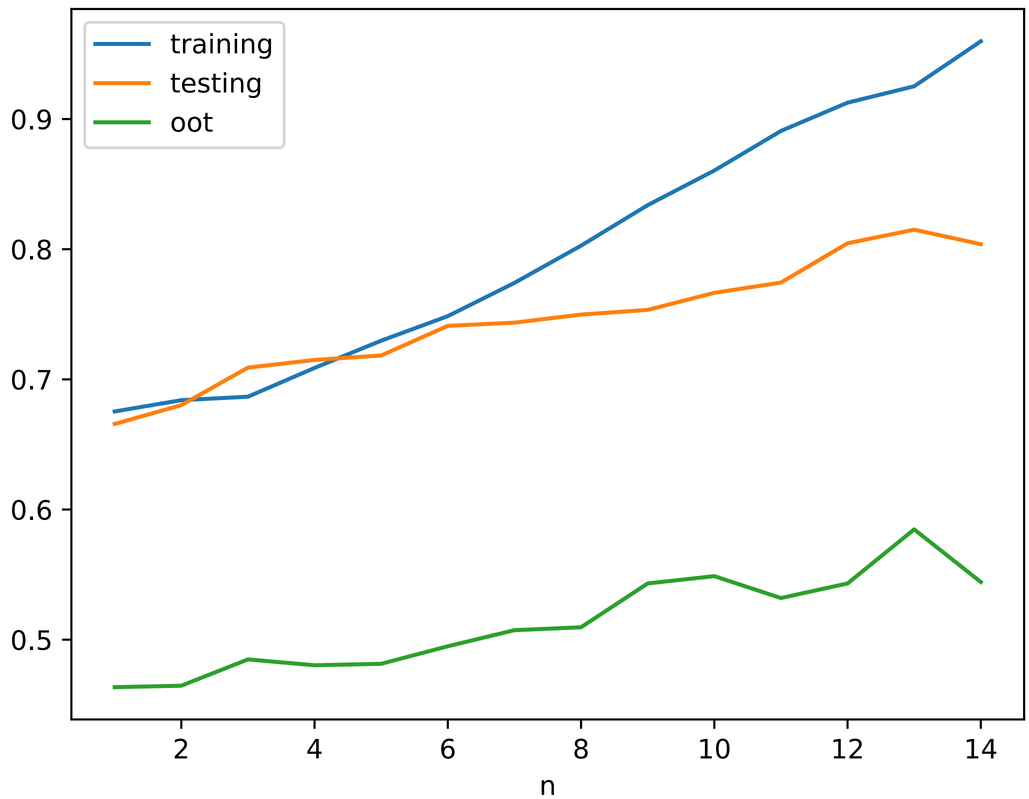


model = DecisionTreeClassifier(max_depth=i)

The graph displays how a DecisionTreeClassifier behaves as its depth increases from 1 to 15. As the tree depth goes up, the model fits the training data better, shown by the rising blue line. However, its ability to perform well on new data (orange line for testing and green line for OOT) starts well but then stops improving and even gets worse, which means the model is too complex and not generalizing well.

This indicates that the best tree depth is probably around where the testing accuracy stops getting better.

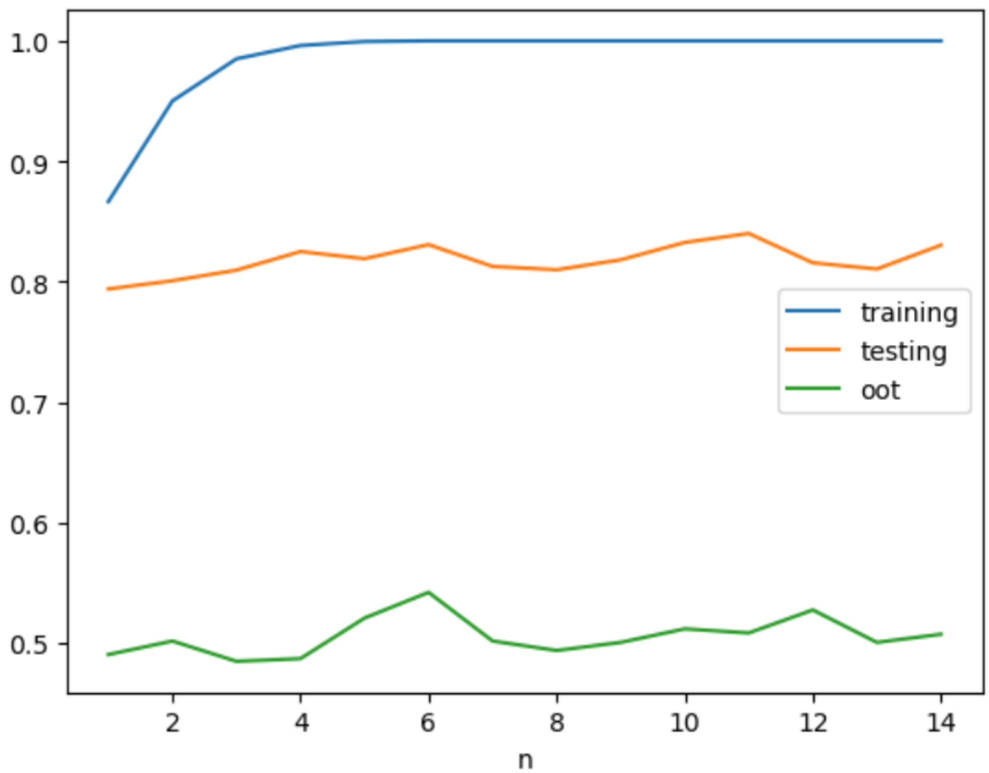
2. Random Forest



model = RandomForestClassifier(max_depth=i, random_state=42)

The graph illustrates the performance of a RandomForestClassifier as its depth increases from 1 to 15. As the tree depth increases, the model fits the training data increasingly well, as shown by the ascending blue line. Meanwhile, its ability to perform well on new data (orange line for testing) also improves, but at a slower pace, hinting at the beginning of overfitting as the model complexity continues to rise. The OOT performance, represented by the green line, remains relatively stable yet low, indicating that the model's generalization to completely new datasets is not improving in tandem with the training and testing improvements. This suggests that an optimal tree depth might be at a point before the testing accuracy gains begin to level off.

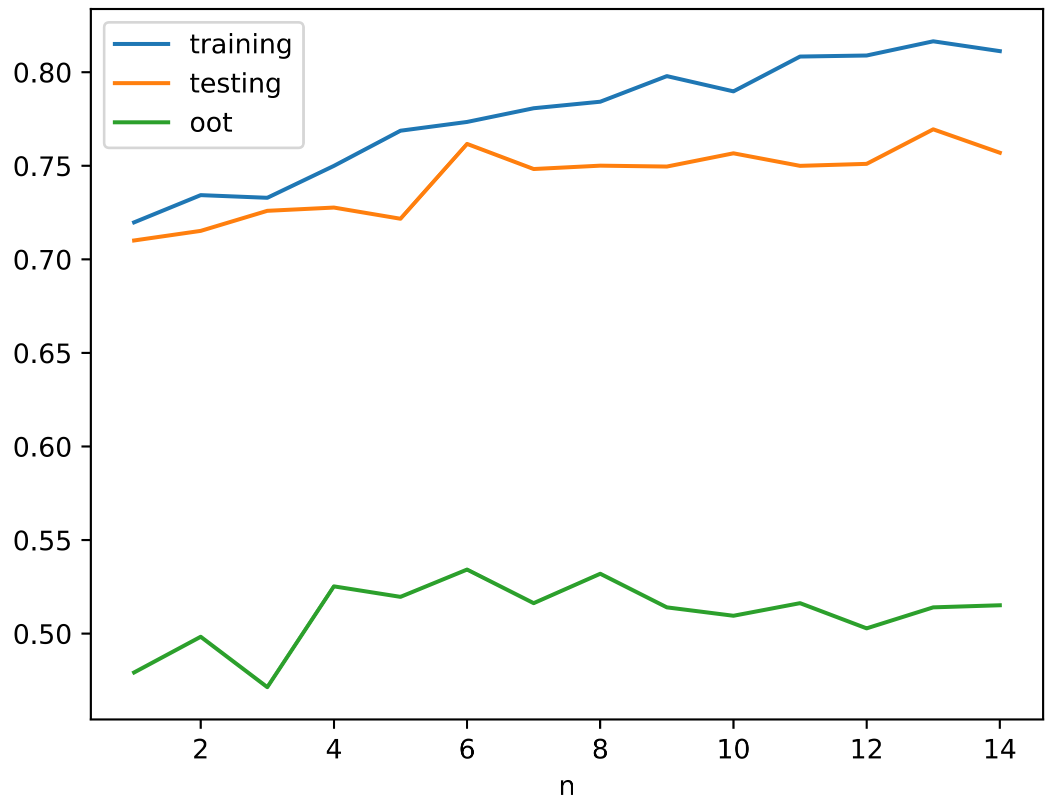
3. LightGBM



model = lgb.LGBMClassifier(num_leaves=10*i, random_state=42)

The graph illustrates the performance of a LightGBMClassifier as the number of leaves increases, controlled by 10*i. Training accuracy (blue line) rapidly improves and then stabilizes, indicating a good fit to the training data. Testing accuracy (orange line) also improves but levels off, suggesting an optimal complexity point where further increases do not enhance model performance on unseen data. The OOT performance (green line), however, remains low and stable, highlighting a consistent failure to generalize to entirely new datasets. This suggests that while initial increases in complexity benefit the model, there's a threshold beyond which more complexity adds little value and could hinder generalization. Optimizing model settings or simplifying the model could improve its applicability and robustness in real-world scenarios. Regular evaluation with new data is crucial to maintain effectiveness.

4. Neural Network



model = MLPClassifier(hidden_layer_sizes=(10*i,),random_state=42)

The graph shows the performance of an MLPClassifier as its complexity increases, indicated by the number of neurons which scale with 10*i. Training accuracy (blue line) consistently improves, demonstrating that more complex models fit training data better. However, testing accuracy (orange line) improves initially but begins to plateau, suggesting the beginning of overfitting. OOT performance (green line) remains flat and low, indicating poor generalization to completely new data. This implies that the optimal complexity, where the model best balances fit and generalization, may occur just before the testing accuracy levels off.