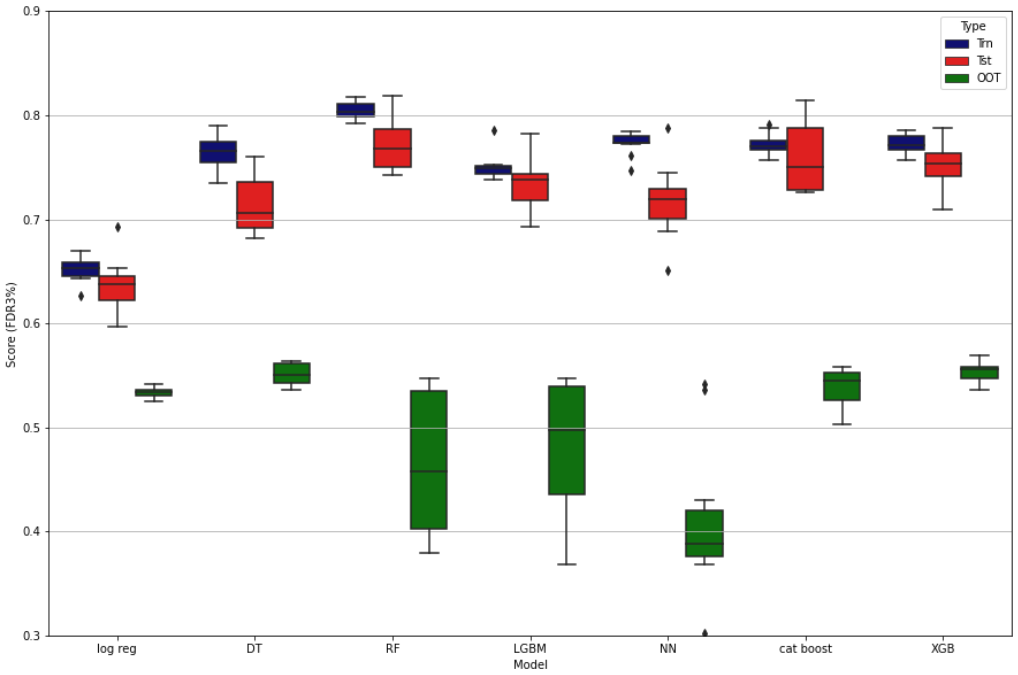
Assignment 4

1. Exploration of Hyperparameters for Logistic Regression, Decision Tree, Random Forest, LightGBM, XGBoost, CatBoost, and Neural Network (MLP).

Model	Parameters							Average FDR@3%		
	Iteration	Penalty	class_weight	Solver	L1_Ratio	max_iter		Train	Test	OOT
Logistic Regression	1 (default)	l2	None	lbfgs	None	100		61.79	61.57	47.2
	2	None	None	lbfgs	None	1000		62.28	60.99	47.15
	3	l1	None	liblinear	None	1000		62.26	61.5	49.66
	4	l2	Balanced	lbfgs	None	500		64.52	64.26	53.35
	5	elasticnet	None	saga	0.5	500		62.25	61.73	48.49
	Iteration	criterion	max_depth	min_samples_leaf	min_samples_split			Train	Test	OOT
Decision Tree	1 (default)	gini	None	2	2			1	57.51	22.84
	2	gini	10	20	40			80.31	73.03	43.85
	3	gini	8	60	120			75.68	72.41	53.57
	4	entropy	10	80	160			79.8	74.46	41.22
	5	gini	8	40	90			75.02	70.93	52.34
	Iteration	n_estimators	max_depth	min_samples_leaf	min_samples_split			Train	Test	OOT
Random Forest	1 (default)	100	None	1	2			1	83.55	38.54
	2	100	8	60	120			79.68	76.82	55.86
	3	50	8	60	120			80.54	76.62	48.32
	4	20	8	50	90			80.25	76.38	49.27
	Iteration	boosting_type	max_depth	n_estimators	num_leaves	colsample_bytree	subsample	learning_rate	Train	Test	OOT
LightGBM	1 (default)	gbdt	-1	100	31	1	1	0.1	99.65	79.12	29.6
	2	gbdt	8	15	31	1	1	0.01	78.04	73.83	45.69
	3	gbdt	5	50	100	0.8	0.8	0.01	76.96	74.6	39.05
	4	gbdt	5	20	31	0.8	0.8	0.01	74.49	73.99	47.59
	5	gbdt	10	20	31	0.8	0.8	0.01	80.83	75.92	41.22
	6	gbdt	10	20	31	1	1	0.01	79.21	75.96	42.29
	7	dart	5	20	31	0.8	0.8	0.02	75.63	73.56	48.99
	Iteration	booster	max_depth	n_estimators	tree_method	min_child_weight	colsample_bytree	subsample	eta	Train	Test	OOT
XGBoost	1 (default)	gbtree	6	100	auto	1	1	1	0.3	99.51	82.26	35.13
	2	gbtree	10	20	auto	5	1	1	0.05	76.53	73.38	53.96
	3	gbtree	10	20	auto	5	0.8	0.8	0.05	79.62	75.73	55.3
	4	dart	10	20	auto	5	0.8	0.8	0.05	80.53	76.48	53.79
	5	dart	10	20	auto	5	0.8	0.8	0.08	81.42	77.45	51.06
	6	gbtree	8	15	auto	5	0.8	0.8	0.05	77.78	75.5	54.86
	7	gbtree	8	15	approx	3	0.8	0.8	0.08	78.31	76.38	48.21
	Iteration	bootstrap_type	depth	iterations	l2_leaf_reg	random_state	learning_rate			Train	Test	OOT
CatBoost	1 (default)	Bayesian	6	1000	3	None	0.03			95.89	81.53	36.64
	2	Bayesian	7	1000	12	10	0.01			89.83	81.41	44.97
	3	Bayesian	5	2000	8	None	0.01			84.52	78.99	46.53
	4	Bayesian	5	1500	12	10	0.01			81.9	77.31	48.21
	5	Bernoulli	5	1500	12	10	0.01			79.11	74.09	53.91
	6	MVS	5	1500	12	10	0.01			81.29	78.61	45.69
	Iteration	activation	solver	learning_rate	learning_rate_init	hidden_layer_size	max_iter	alpha		Train	Test	OOT
Neural Network	1 (default)	relu	adam	N/A	0.001	100	200	0.0001		72.83	71.38	43.12
	2	relu	sgd	constant	0.01	70	200	0.001		73.14	70.96	41.45
	3	relu	adam	adaptive	0.001	20	200	0.005		68.24	68.06	55.53
	4	relu	adam	adaptive	0.001	70	200	0.005		70.18	69.35	55.86
	5	relu	adam	constant	0.01	200	500	0.0005		73.54	71.5	37.87
	6	logistic	adam	adaptive	0.01	100	200	0.0005		73.51	73.45	49.38
	7	tanh	adam	adaptive	0.01	150	200	0.0005		75.43	71.58	37.7

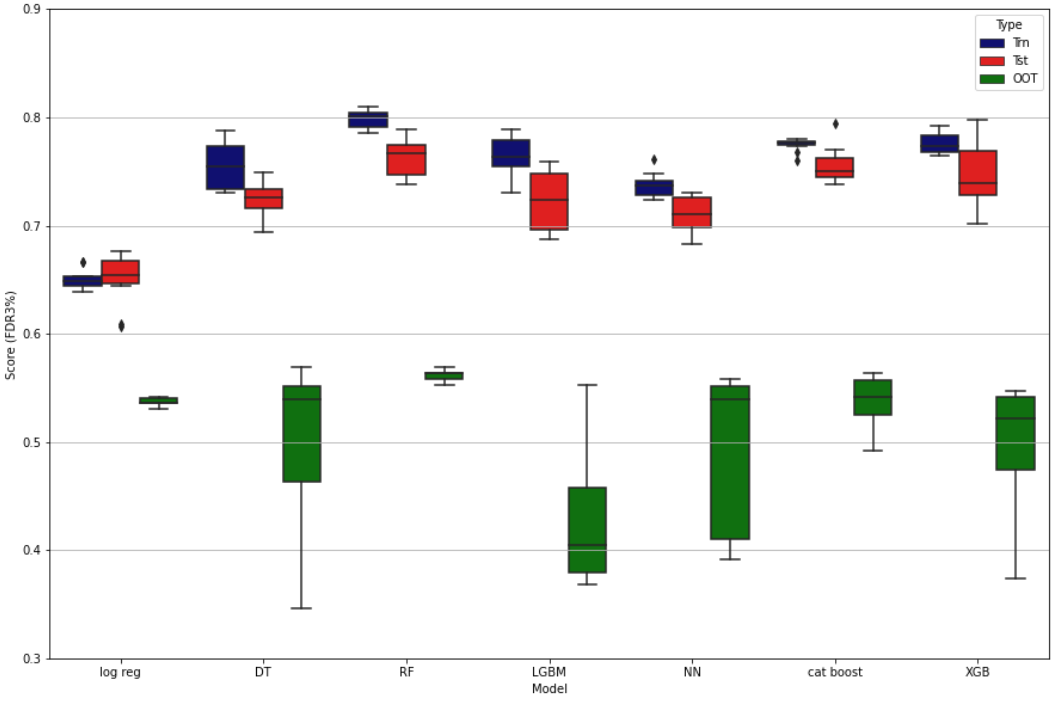
2. Boxplot of all models with my best model hyperparameters

a) With 15 variables



I ran all my best models for the top 15 variables after last week's feature selection. Here, Logistic Regression, Decision Tree, CatBoost, and XGB perform the best on OOT validation set with low variation as compared to the other models. When it comes to training and testing accuracy, Catboost and XGBoost are significantly better than Logistic Regression and slightly better than Decision Tree. I would probably pick XGBoost here because of high OOT accuracy with little variation.

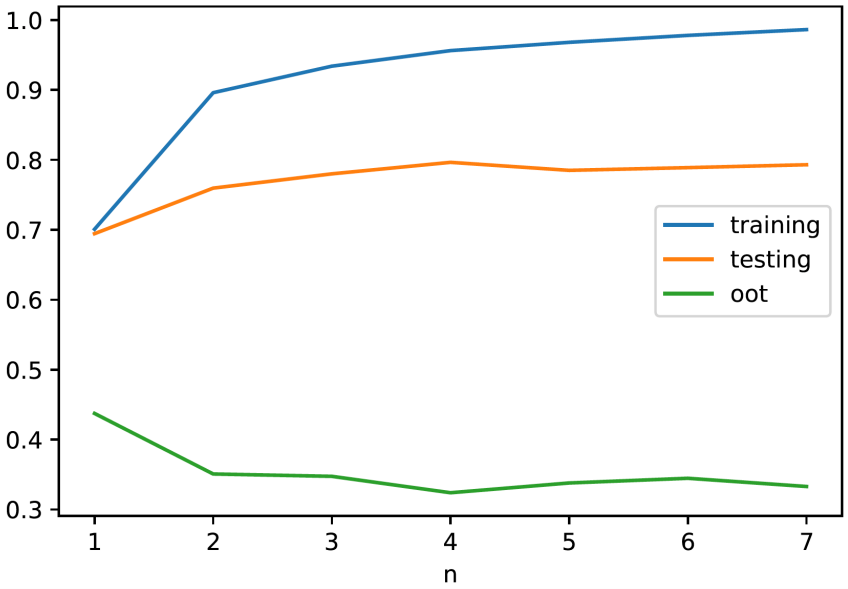
b) With 10 variables



I ran all my best models for the top 10 variables after last week's feature selection. Here, logistic regression, Random Forest and Catboost perform well on OOT data with relatively lower variation as compared to the other models. However, LR has much lower training and testing accuracy when compared to RF and Catboost. In this case, my best model performance seems to be with Random Forest with the highest training, testing and OOT accuracies with very little variation. I would go with RF here.

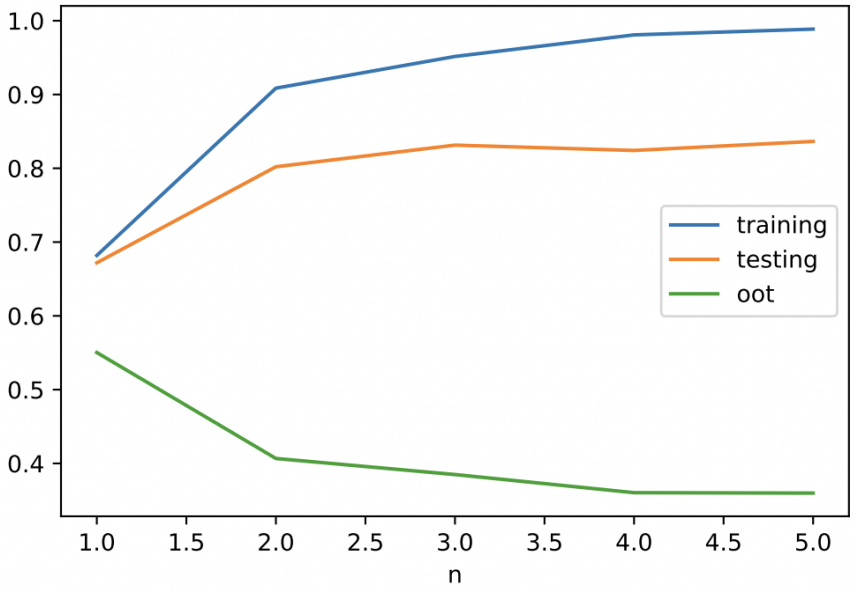
3. Performance vs Complexity of Models when they overfit:

a) LGBM



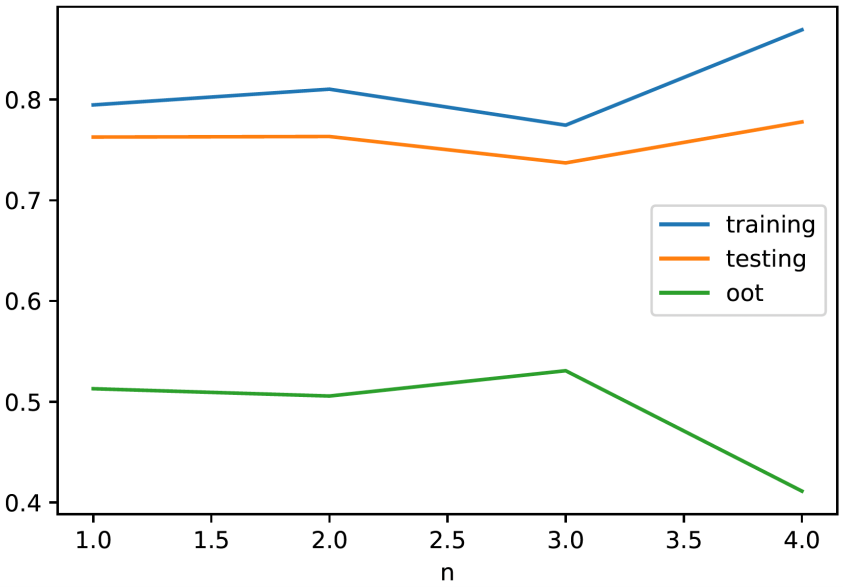
For LBGM, I kept all default parameters except n_estimators (number of estimators), which I varied from 1 to 61 with a step size of 10 (n = 7 on the x-axis). In LightGBM, the default number of estimators is 100 (which gives an overfit model) and my best model takes around 20, so increasing the number of estimators seemed like a good choice to cause this model to overfit. We see that as the distance between training and testing accuracy increases, the accuracy on OOT validation set goes down indicating an overfit model which is not able to generalize to unseen data.

b) XGBoost



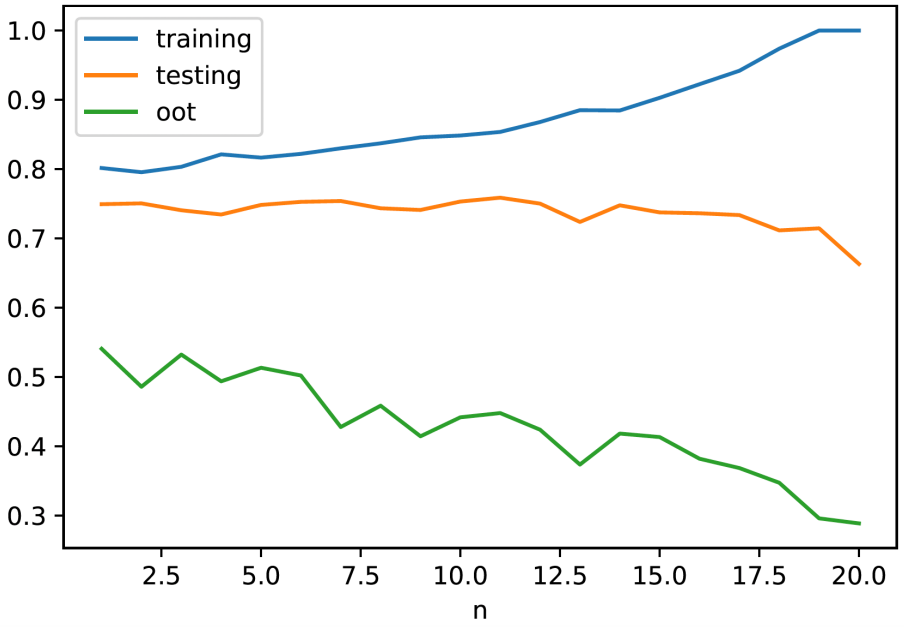
For XGBoost, I varied n_estimators from 1 to 81 with a step size of 20 (n=5 on the x-axis). In XGBoost, the default number of estimators is 100 (which gives an overfit model) and my best model takes around 15, so increasing the number of estimators seemed like a good choice to cause this model to overfit. We see that as the distance between training and testing accuracy increases, the accuracy on OOT validation set goes down (starts from 0.55 and goes below 0.4) indicating an overfit model which is not able to generalize to unseen data.

3) CatBoost



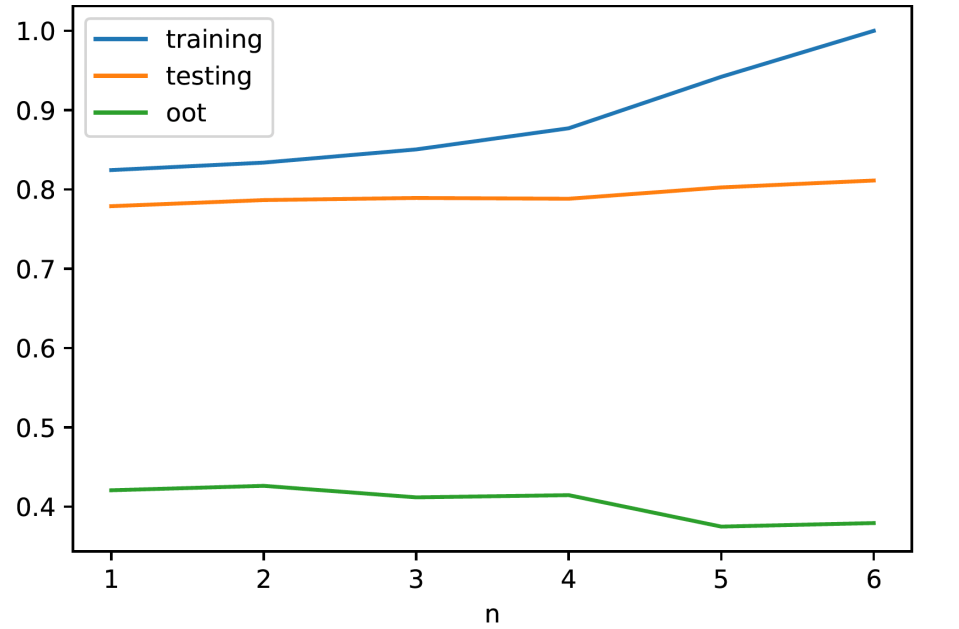
For Catboost, I trained 4 models while varying iterations, l2_leaf_reg and eta simultaneously (n = 4 on the x-axis). I kept iterations as [1500, 1500, 1000, 1000], l2_leaf_reg as [12,8,8,3] and eta as [0.01,0.01,0.01,0.03]. I found that if you reduce the L2 regularization on Catboost model (indicative by smaller values of l2_leaf_reg), increase eta (learning rate) and reduce iterations of the model, the model tends to overfit so that is what I did and it caused the model to overfit on the 4th iteration. We see that for that iteration, our OOT accuracy goes down from 0.5 to 0.4 and the gap between training and testing accuracy is also high.

d) Decision Tree



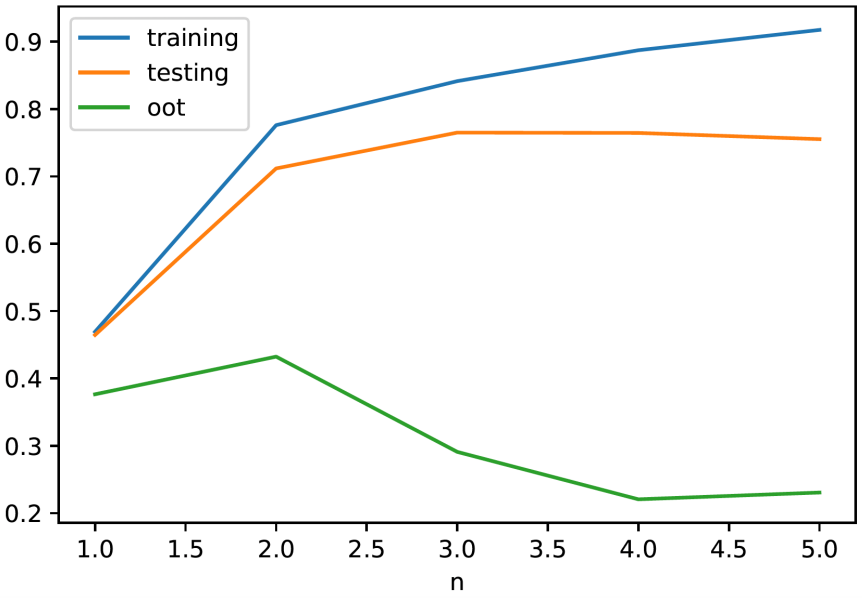
For the Decision Tree model, I kept max_depth at 8 and varied min_samples_leaf and min_samples_split from higher values to less values. min_samples_leaf is the minimum number of samples required to be at a leaf node, min_samples_split is the minimum number of samples required to split an internal node. Decreasing these values leads to a more complex tree structure. I started with min_samples _leaf as 60 (i) and min_samples_split as 120 (2*i) and reduced the value of i by 3, so min_samples_leaf went from 60 to 3 (n=20 on the x-axis) and min_samples_split went from 120 to 6 (2*i). Smaller values of min_samples_split and min_samples_leaf causes the model to overfit. We see that as the distance between training and testing accuracy increases, the accuracy on OOT validation set goes down (starts from 0.55 and goes to around 0.3) indicating an overfit model which is not able to generalize to unseen data.

e) Random Forest



For the Random Forest model, I kept n_estimators at 50 and varied min_samples_leaf and min_samples_split from higher values to less values. min_samples_leaf is the minimum number of samples required to be at a leaf node, min_samples_split is the minimum number of samples required to split an internal node. Decreasing these values leads to a more complex tree structure. I started with min_samples _leaf as 80 (i) and min_samples_split as 160 (2*i) and reduced the value of i by 15, so min_samples_leaf went from 80 to 5 (n=6 on the x-axis) and min_samples_split went from 160 to 10 (2*i). Smaller values of min_samples_split and min_samples_leaf causes the model to overfit. We see that as the distance between training and testing accuracy increases, the accuracy on OOT validation set goes down indicating an overfit model which is not able to generalize to unseen data.

f) Neural Network



For the neural network, I changed the hidden layer size from 1 to 81 with a step size of 20 (n=5 on the x-axis) which increases model complexity. We see that for 1 hidden layer, the model is underfit and the training and testing accuracy is low. As we go till hidden layer 41, our training and testing accuracy improves and so does our OOT accuracy. However, beyond that point the model starts to overfit the data with further increase in hidden layers and as the gap between training and testing accuracy widens (due to overfitting), the OOT accuracy goes down indicating an overfit model which is not able to generalize to unseen data.