**Model Selection Report**

**Part 1:**

**Table shows a variety of tests of different reasonable hyperparameters for a variety of model types.**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Decision Tree | #Variables | criterion | splitter | max\_depth | min\_samp\_split | min\_samp\_leaf | trn | tst | oot |
| 1 | 20 | entropy | Best | 10 | 100 | 100 | 0.7 | 0.69 | 0.54 |
| 2 | 20 | gini | random | 10 | 120 | 60 | 0.656 | 0.645 | 0.555 |
| 3 | 20 | Gini | Best | 5 | 20 | 20 | 0.67 | 0.66 | 0.54 |
| 4 | 20 | Gini | Best | 10 | 150 | 60 | 0.7 | 0.69 | 0.52 |
| 5 | 20 | entropy | random | 10 | 80 | 40 | 0.672 | 0.66 | 0.542 |
| 6 | 20 | entropy | best | 5 | 70 | 20 | 0.679 | 0.663 | 0.558 |

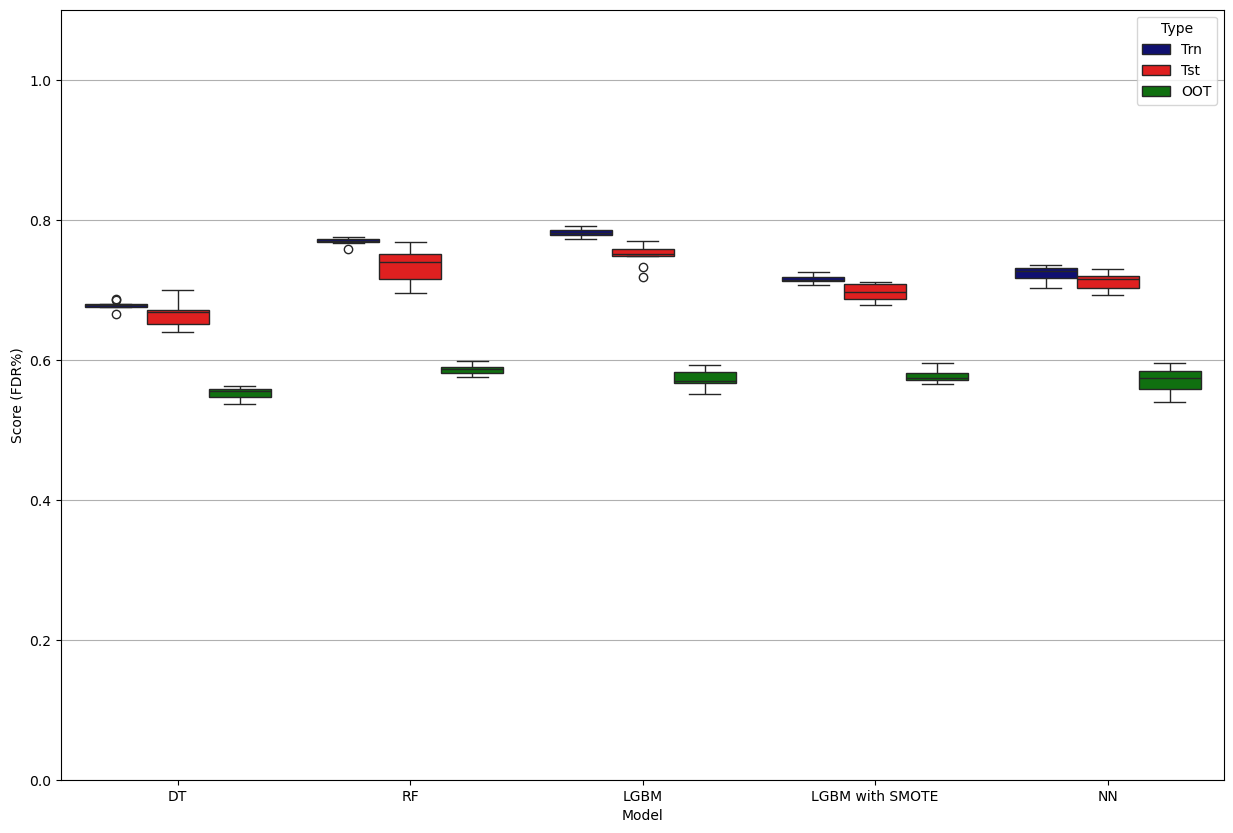
|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Random Forest | # Variables | max\_depth | criterion | n\_estimators | min\_samples\_split | min\_samples\_leaf | Train | Test | OOT |
| 1 | 20 | 5 | gini | 5 | 20 | 10 | 0.67 | 0.663 | 0.533 |
| 2 | 20 | 8 | gini | 15 | 50 | 25 | 0.72 | 0.7 | 0.57 |
| 3 | 20 | 10 | gini | 10 | 20 | 10 | 0.755 | 0.715 | 0.567 |
| 4 | 20 | 15 | entropy | 25 | 60 | 30 | 0.757 | 0.733 | 0.583 |
| 5 | 20 | 15 | entropy | 20 | 40 | 20 | 0.78 | 0.738 | 0.576 |
| 6 | 20 | 15 | entropy | 35 | 50 | 25 | 0.77 | 0.73 | 0.59 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LightGBM | # Variables | boosting\_type | n\_estimators | learning\_rate | max\_depth | num\_leaves | subsample | colsample\_bytree | min\_child\_samples | eval\_metric | random\_state | Train | Test | OOT |
| 1 | 20 | gbdt | 500 | 0.05 | 7 | 31 | 0.8 | 0.8 | 50 | auc | 42 | 0.957 | 0.785 | 0.577 |
| 2 | 20 | gbdt | 100 | 0.1 | -1 | 31 | 1 | 1 | 15 | auc | 42 | 0.935 | 0.768 | 0.569 |
| 3 | 20 | GOSS | 800 | 0.03 | 5 | 22 | 0.8 | 0.8 | 50 | AUC | 42 | 0.929 | 0.791 | 0.583 |
| 4 | 20 | gbdt | 100 | 0.1 | -1 | 31 | 1 | 1 | 50 | auc | 42 | 0.903 | 0.775 | 0.575 |
| 5 | 20 | GOSS | 500 | 0.05 | 4 | 5 | 1 | 1 | 80 | auc | 42 | 0.84 | 0.77 | 0.6 |
| 6 | 20 | GOSS | 2000 | 0.01 | 5 | 22 | 0.8 | 0.8 | 50 | AUC | 42 | 0.909 | 0.783 | 0.586 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Neural network | # Variables | Activation | learning\_rate\_init | Alpha | Solver | #Nodes per hidden layer | Hidden Layers | train | Test | OOT |
| 1 | 20 | tanh | 0.0005 | 0.001 | adam | 20 | 2 | 0.723 | 0.696 | 0.556 |
| 2 | 20 | relu | 0.001 | 0.001 | adam | 10 | 2 | 0.7 | 0.68 | 0.553 |
| 3 | 20 | tanh | 0.001 | 0.0001 | adam | 30 | 1 | 0.77 | 0.74 | 0.62 |
| 4 | 20 | relu | 0.0005 | 0.001 | adam | 15 | 2 | 0.701 | 0.695 | 0.571 |
| 5 | 20 | relu | 0.0005 | 0.001 | adam | 20 | 2 | 0.723 | 0.697 | 0.581 |
| 6 | 20 | relu | 0.0001 | 0.01 | adam | 30 | 1 | 0.668 | 0.67 | 0.551 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| LightGBM with SMOTE | # Variables | n\_estimators | learning\_rate | max\_depth | num\_leaves | min\_child\_samples | Train | Test | OOT |
| 1 | 20 | 75 | 0.1 | 4 | 6 | 10 | 0.71 | 0.705 | 0.57 |
| 2 | 20 | 100 | 0.2 | 5 | 6 | 20 | 0.746 | 0.726 | 0.559 |
| 3 | 20 | 20 | 0.1 | 3 | 6 | 15 | 0.677 | 0.667 | 0.566 |
| 4 | 20 | 50 | 0.15 | 3 | 8 | 15 | 0.71 | 0.706 | 0.578 |
| 5 | 20 | 100 | 0.1 | 6 | 6 | 25 | 0.722 | 0.713 | 0.569 |
| 6 | 20 | 100 | 0.15 | 5 | 6 | 20 | 0.732 | 0.727 | 0.576 |

**Part 2: Plot shows a single well-tuned performance of trn, tst, oot for a variety of models.**



**Part 3: Demonstration of overfitting. As the hyperparameter changes, trn gets better, tst gets worse or stays about the same.**

1. **Decision Tree**

This Decision Tree Classifier is configured to demonstrate overfitting. It uses the 'entropy' criterion to aggressively split nodes based on information gain, seeking the purest possible class separation. The splitter='best' ensures optimal splits at each step. With max\_depth=None, the tree can grow indefinitely, while min\_samples\_split=2 and min\_samples\_leaf=1 allow splits and leaf creation with the smallest possible subsets of data. These permissive settings enable the model to perfectly memorize the training data, including noise, resulting in minimal training error but poor performance on unseen data—clearly illustrating the classic signs of overfitting.

**Model Code:**

*model = DecisionTreeClassifier(*

*criterion='entropy',*

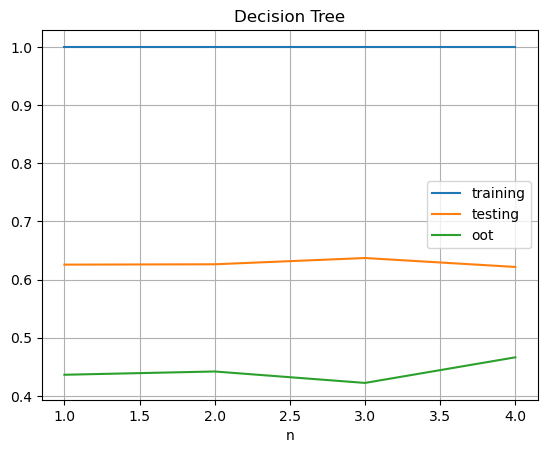
*splitter='best',*

*max\_depth=None, # no depth limit*

*min\_samples\_split=2, # split with just 2 samples*

*min\_samples\_leaf=1 # leaves can be 1 sample*

*)*



1. **Random Forest**

To design this Random Forest Classifier so that it leans into overfitting, I set `max\_depth=20`—not infinite, but deep enough to let each tree chase complex patterns and noise. With `criterion='entropy'`, the trees aggressively hunt for the purest splits, and `min\_samples\_split=2` with `min\_samples\_leaf=1` give them full freedom to fragment the data down to its finest details. I kept `n\_estimators=100` to get ensemble behavior, but not so many that averaging would completely wash out the overfitting.

**Model Code:**

*RandomForestClassifier(*

*max\_depth=20, # still deep enough to overfit*

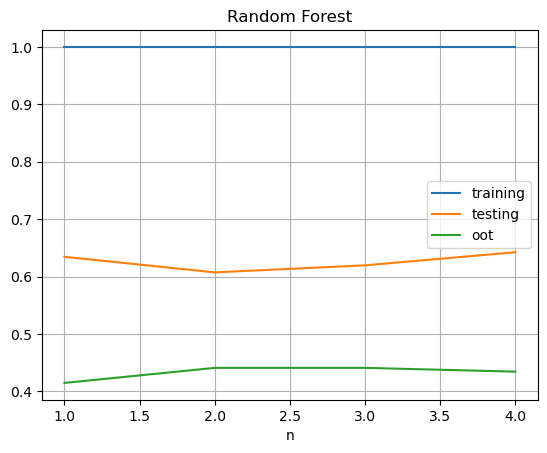
*criterion='entropy', # sensitive to information gain*

*n\_estimators=100, # fewer trees → much faster than 1000*

*min\_samples\_split=2, # very flexible splitting*

*min\_samples\_leaf=1, # allows pure leaves*

*)*



1. **Neural network**

This MLP Classifier is highly overfitting. I gave it a single hidden layer with 150 neurons so it has the capacity to memorize patterns in small or noisy datasets. Using the 'relu' activation keeps the model non-linear and expressive, while the 'adam' optimizer helps it converge very fast. The alpha value (the L2 regularization strength) was cracked down to a very low 0.00001, so that it basically memorizes everything. With a constant learning rate and learning\_rate\_init=0.001, the network maintains stable updates that can easily zero in on local noise. With this setup, the neural net hugs the training data way too tightly and leads to high overfitting.

**Model Code:**

MLPClassifier(

hidden\_layer\_sizes=(150,),

activation='relu',

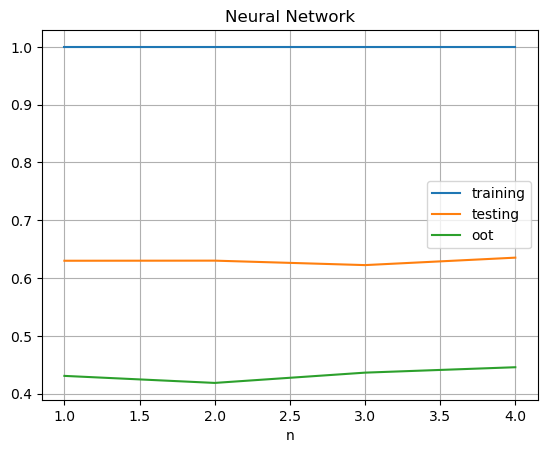
solver='adam',

alpha=0.00001,

learning\_rate='constant',

learning\_rate\_init = 0.001

)



1. **Light GBM**

This LGBMClassifier is intentionally configured to induce overfitting in a controlled manner. By selecting 'GOSS' (Gradient-based One-Side Sampling) as the boosting type, the model prioritizes instances with larger gradients, which can lead to an overemphasis on harder-to-predict or noisy samples. The high number of estimators (n\_estimators=2000) combined with a very small learning rate (learning\_rate=0.001) allows the model to incrementally fit the data with high precision, increasing the risk of capturing noise. Although num\_leaves=5 constrains the complexity of individual trees, the deep structure allowed by max\_depth=100 offsets this, creating the potential for complex decision paths. Additionally, full row and feature sampling (subsample=1, colsample\_bytree=1) removes regularization through randomness, while the relatively high min\_child\_samples=80 offers only a minimal constraint. Overall, this setup shows how overfitting can still occur in gradient boosting frameworks, even when some conservative hyperparameters are applied.

**Model Code:**

*lgb.LGBMClassifier(*

*boosting\_type='GOSS',*

*n\_estimators=2000, # Number of trees*

*learning\_rate = 0.001, # Step size shrinkage*

*max\_depth=100, # Maximum tree depth*

*num\_leaves=5, # Number of leaves per tree*

*subsample=1,*

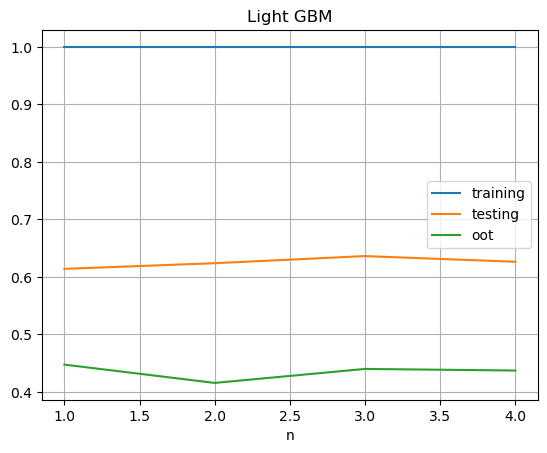
*colsample\_bytree=1,*

*min\_child\_samples=80,*

*random\_state=42, # For reproducibility*

*eval\_metric='auc'*

*)*



1. **Light GBM with SMOTE**

This model shows overfitting even after applying SMOTE. While SMOTE helps balance class distribution by generating synthetic minority class samples, it can also introduce noise and artificial patterns. The LGBMClassifier that follows is configured to exploit this: with only 50 trees and a relatively high learning\_rate=0.15, the model is encouraged to aggressively fit the resampled data. Although the tree depth is shallow (max\_depth=3), the num\_leaves=8 and low min\_child\_samples=15 allow it enough flexibility to tightly fit the synthetic and potentially noisy patterns introduced by SMOTE. The combination creates a setup where the model can overfit not only to the original data but also to the artificial signal. This is an example of how overfitting can persist even with imbalance mitigation techniques.

**Model Code:**

*sm = SMOTE()*

*X\_trn\_sm, Y\_trn\_sm = sm.fit\_resample(X\_trn,Y\_trn)*

*lgb.LGBMClassifier(*

*n\_estimators = 50,*

*learning\_rate =0.15,*

*max\_depth = 3,*

*num\_leaves = 8,*

*min\_child\_samples = 15*

*)*

