HW1: Supervised Learning

Dataset: BNP Paribas Cardif Claims Management:

The said dataset has 131 predictors where 113 predictors are numerical and 19 are categorical with varying cardinality. Also, 33.5% of the dataset has missing values. Owing to high no. of predictors columns with both numerical and categorical columns and large set of missing values makes this dataset challenging and exiting to compare performance of different ML algorithms. Most of the predictors are also highly correlated and right- skewed.

The target column is the binary column and the objective to predict if a sample belongs to category 0 or 1. The target column is also imbalance with 76% samples belong to category 1.

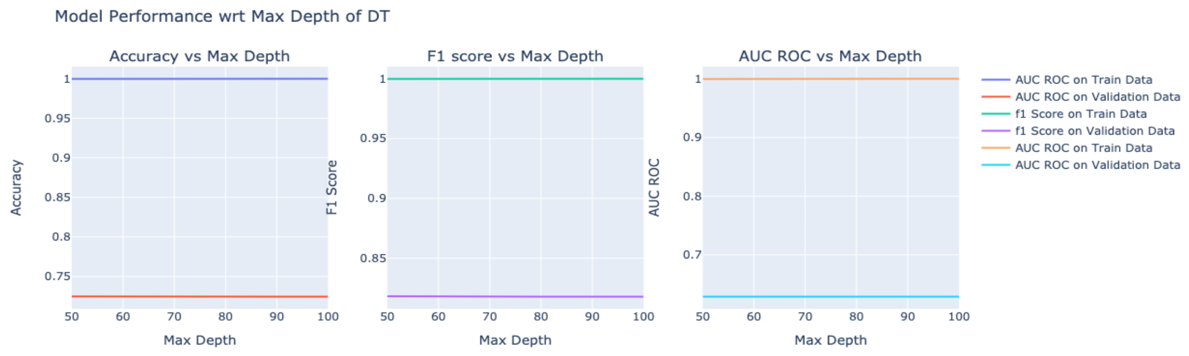
Pre-processing on Dataset:

The dataset has 19 categorical, hence conversion of categorical columns to numerical is mandated by most of the ML algorithms. One standard approach of representing the categorical variables is One hot encoding, however, owing to high cardinality in some of the categories which in effect would have induced high dimensionality in the data, it was decided to use Target encoding with Laplace prior with smoothness value of 300. The missing values in categorical data is filled by adding another factor "__MISS__" whereas for numerical dataset the missing value was filled using median. Also, no emphasis was given to feature selection algorithms as it would have comprised the evaluation of different ML algorithms on the said dataset. In case of SVM and KNN, the predictors are scaled with mean=0 and std = 1 as decision boundaries are affected by different scales of predictors.

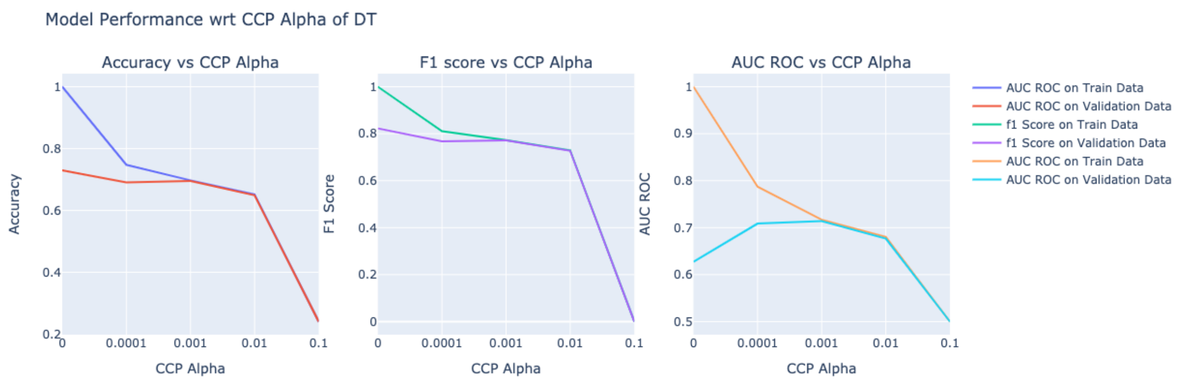
Analysis of Decision Tree on BNP Paribas Cardif Claims Management

Since the dataset is imbalanced we set the class_weight="balanced" in decision tree classifier. Also, the metric used to choose best model is set to "balanced_accuracy".

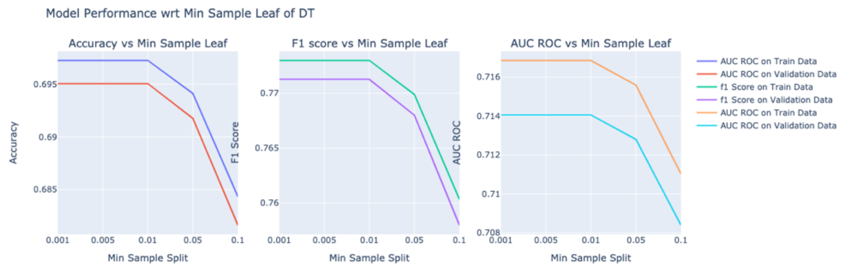
Tuning Max Depth for decision trees:
We tried different values of max_depth 50, 60, 70, 80, 90, 100. It is quite visible through the graphs that there was no effect of using different values of max_depth as different metrics remain more or less same for different values of max_depth. Also, it is evident that decision tree is overfitting the data since mean cross validation score on validation fold is nowhere near to the mean cross validation score on training folds.

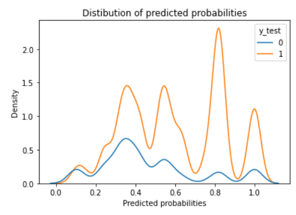


Tuning Pruning param CCP Alpha:
We tried different values of ccp alpha [0, 1e-4, 1e-3, 1e-2, 0.1]. We started with the small value of 0 so as to fully grow the tree. It is visible from the graph that for ccp alpha = 0, there was huge difference between training and validation error, thus we can safely say that without pruning the decision tree was behaving as its policy of overfitting the training data. With ccp alpha value of 0.001 the training and validation error on different metrics nearly overlaps. So, it is safe to assume that ccp alpha value of 0.001 is the optimal value for this dataset. Increasing ccp value worsen error metrics on both training and validation folds.



Tuning min sample leaf:
The param min_sample_leaf specifies minimum number of samples required to split an internal node. Greater value of memorising the data instead learning from it which is evident for figure 3. The error metric on training and validation folds look optimal for the min_leaf_split of 0.01.





On the unseen data with min_leaf_split = 0.01 and ccp alpha = 0.001, we achieved f1 score of 0.7317

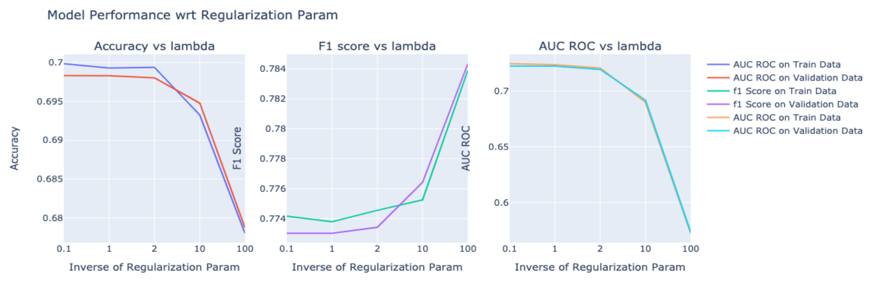
Accuracy	0.6393270548444471
F1 Score	0.7317052010584306
AUC ROC	0.6358509132553408

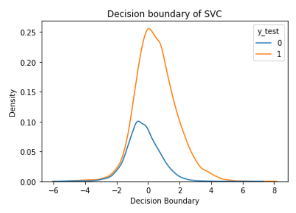
	Confusion Matrix	
0	**5059**	2980
1	**9390**	16868

	Precision	**Recall**	**F1-Score**	**Support**
0	**0.35**	0.63	0.45	8039
1	**0.85**	0.73	0.73	26258

Analysis of SVC on BNP Paribas Cardif Claims Management

Running SVM with RBF kernel took 2.30 hrs for training a single svc model, the reason behind this would high dimensionality of data since SVM does not scale with high dimensional data. To solve this problem, we retracted to use LinearSVC where kernel is set to “linear”. The loss function was set to “hinge”and class_weight to “balanced”. We tried different values of regularization parameter “C” [0.1, 1, 2, 10, 100]. The best C that was found using cross validation is 0.1.





Accuracy	0.3797708254366271
F1 Score	0.3457185039370078
AUC ROC	0.5651496229639283

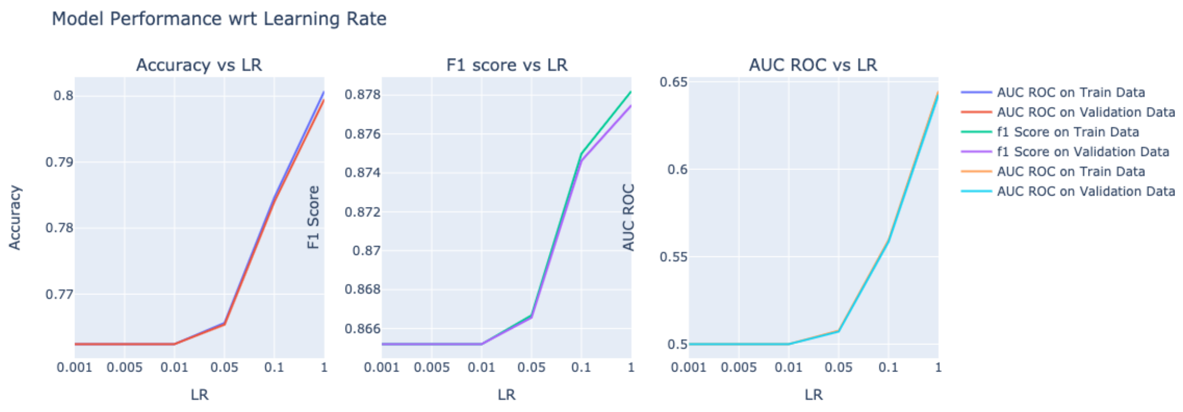
	Confusion Matrix	
0	7405	680
1	20592	5620

	Precision	Recall	F1-Score	Support
0	0.26	0.92	0.41	8085
1	0.89	0.21	0.35	26212

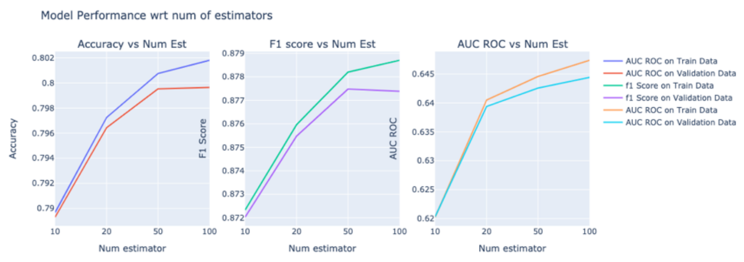
On the unseen data, the LinearSVC has pessimal precision on class 0 whereas precision on class 1 is acceptable which can be easily understood by looking at decision boundaries that the model has learned. It is clearly visible that the decision function values are overlapping for both the classes. Adjusting the decision boundary value would help in maintaining balance between precision and recall.

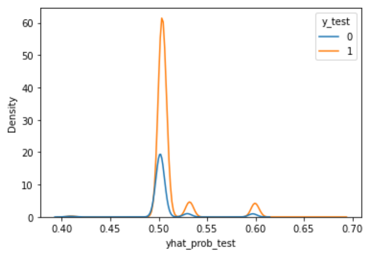
Analysis of AdaBoost on BNP Paribas Cardif Claims Management

AdaBoost is a boosting algorithm which fits multiple weak classifiers and combines their output into a weighted sum that represents the final output of the boosted classifier. We used DecisionTree with max_depth = 1 as weak classifiers. The parameter num_estimators represents number of weak classifiers to fit sequentially.
We first tried tuning the learning rate parameter keeping base classifier as DecisionTrees with max_depth of 1 and num_estimators = 50.



We found that learning_rate value of 1 exceeds performance than rest of the other values. The possible reasoning behind it is DecisionTrees are weak classifiers, in order to reduce bias the AdaBoost was reducing bias in consecutive steps taking in consideration the overall bias reduced by consecutive DecisionTrees than shrinking output of each weak classifier. We then varied num_estimators keeping learning_rate = 1, we found that the validation error plateau after 50 estimators though the train error was decreasing thereby increasing accuracy and f1 score.





Overall the model achieve f1 score of 0.8553 on the unseen data.

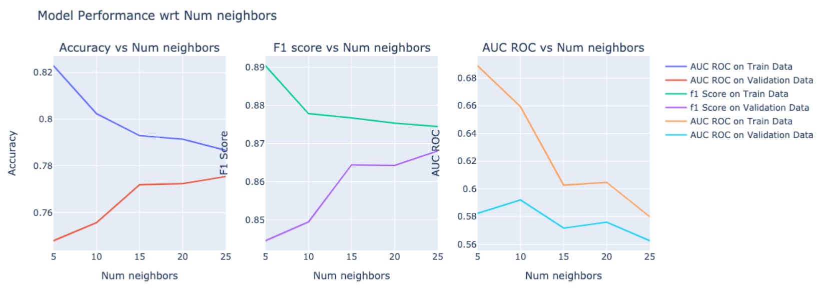
Accuracy	0.7625156719246581
F1 Score	0.8553825393725253
AUC ROC	0.5875878501219729

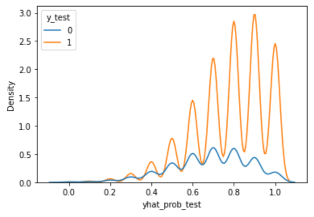
	Confusion Matrix	
0	2064	6224
1	1921	24088

	Precision	Recall	F1-Score	Support
0	0.52	0.25	0.34	8288
1	0.79	0.93	0.86	26009

Analysis of KNN on BNP Paribas Cardif Claims Management

As expected the KNN model is overfitting the training data for lower values of neighbours. As the number of neighbours increased the training accuracy followed a downward trend whereas accuracy on validation data improved. By looking at different metrics num_neighbors = 20 seems to be optimal value for this dataset.





The KNN model achieved overall accuracy of 0.7385 on the unseen data. Looking at distribution of predicted pr obabilities, it seems that probability distribution of class 0 and class 1 overlapped thus KNN model achieved a lo wer value of precision for class 0. The below table shows KNN model performance.

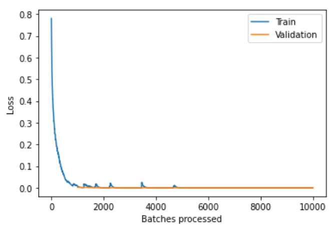
Accuracy	0.738519404029507
F1 Score	0.8396624472573839
AUC ROC	0.5621603670535862

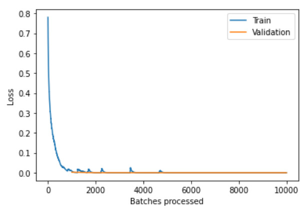
	Confusion Matrix	
0	1847	6242
1	2726	23482

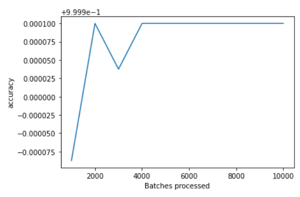
	Precision	Recall	F1-Score	Support
0	0.40	0.23	0.29	8089
1	0.79	0.90	0.84	26208

Analysis of Neural Network on BNP Paribas Cardif Claims Management

For neural net we didn’t apply target encoding on categorical predictors. We used the neural network architecture proposed by fast.ai where categorical predictors are converted into numerical by means of entity embeddings.

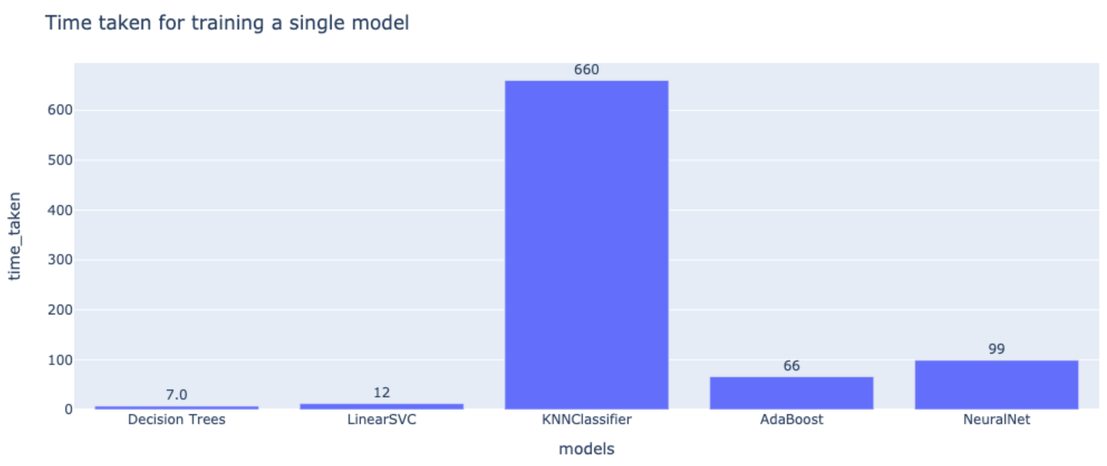






Runtime comparison

All the algorithms used single core, as expected DecisionTree classifie took the least time whereas time taken by KNN is highest. For Neural net we ran only 10 epochs.

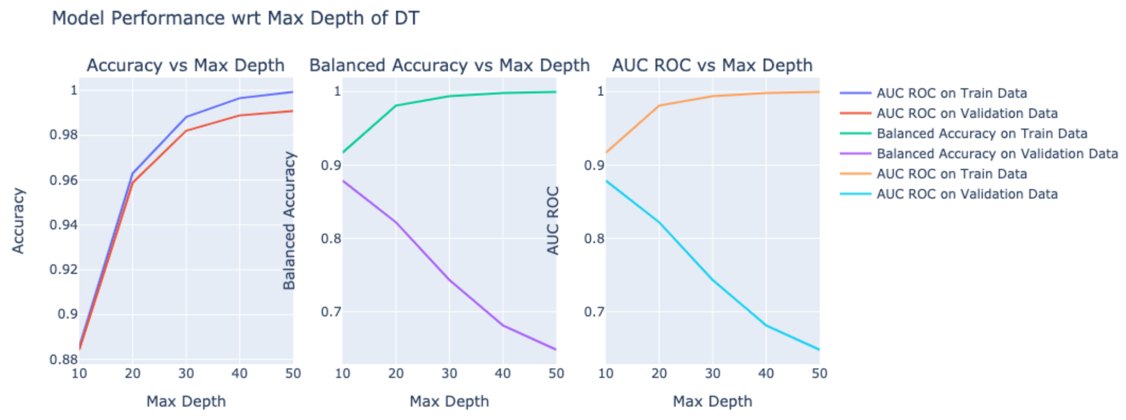


Dataset: Backorders Prediction

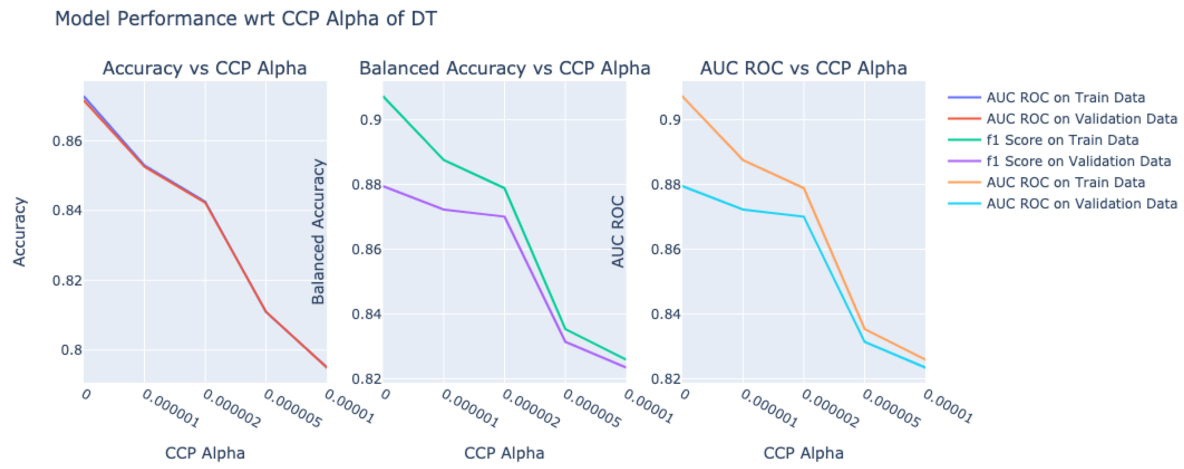
A backorder is an order for a good or service that cannot be filled at the current time due to a lack of available supply. The item may not be held in the company's available inventory but could still be in production, or the company may need time manufacture the product. The backorder is an indication that demand for a company's product outweighs its supply. They may also be known as the company's backlog. This particular dataset contains backorder information of different products, the target variable “went_to_backorder” is highly skewed dataset with minority class only representing 0.006690% of the samples. This dataset is interesting to evaluate performance of different classifiers on highly skewed datasets. The purpose of the exercise is to evaluate and understand performance of different classifiers, it is not meant to apply specific pre-processing steps such as under-sampling and/or over-sampling and then compare performance. The dataset has numerical and categorical predictors, during pre-processing the binary predictors containing “Yes” and “No” values were converted into numerical 1 and 0. The missing values in the numerical predictors were represented by -1. The target variable is label encoded. Since, the target variable is extremely skewed we used class_weight="balanced" in all classifiers to take imbalance target into account.

Analysis of Decision Tree on Backorders Prediction

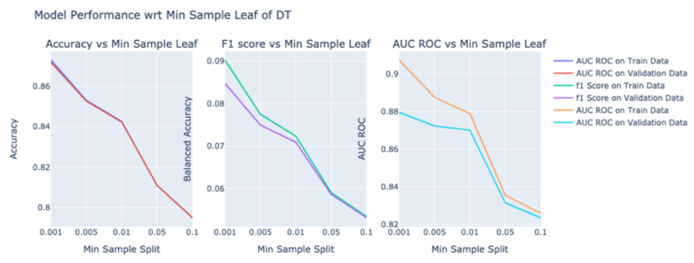
We used balanced accuracy as the metric to compare the performance of different classifiers. For the decision trees, we tried to tune max_depth, pruning parameter and max_samples_split hyperparameters. It is evident from the below figure, with increasing value of max_depth accuracy also starts increasing as decision tree started to overfit the data, balanced accuracy and auc roc score started to decrease on validation for the same reason. The max_depth of 10 was found to be optimal.

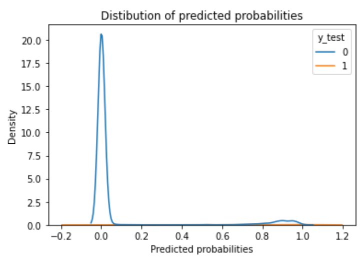


After tuning max_depth, we tuned pruning param ccp alpha. The optimal value of ccp value was found to be 0.



For the max_depth of 10, decision didn't need any type of pruning. Hence we got ccp_alpha = 0 as optimal value. Thereafter, we tuned min_sample_split whose optimal value found out to be 0.001. With increasing value of min_sample_split, the decision tree would become more shallow hence overall bias on train and validation dataset would be higher which is evident from below figure.





Performance of decision tree with max_depth = 10, ccp_alpha = 0 and min_sample_split = 0.001. Looking at distribution of predicted probabilities it is quite visible that model has done a good job in predicting samples of class 0 but it struggled to classify samples with class = 1.

Accuracy	0.872126045209120
Balanced Accuracy	0.891074895874115

	Confusion Matrix	
0	464360	38610
1	478	2910

	Precision	Recall	F1-Score	Support
0	1.00	0.92	0.96	502970
1	0.07	0.86	0.13	3388

AUC ROC	0.891074895874115

Analysis of SVC Backorders Prediction Dataset

The performance of LinearSVC is worse than the random classifier, the possible reason behind this is data may not linearly separable at all. We have to forcefully choose “linear” kernel as SVM with “rbf” kernel was taking a lot of time to train.

Accuracy	0.9932557597589058
Balanced Accuracy	0.4999731594329681
AUC ROC	0.4999731594329681

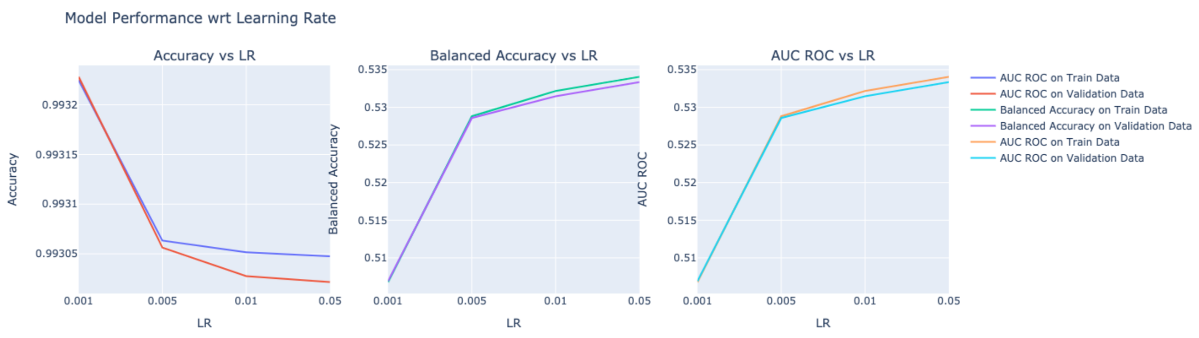
	Confusion Matrix	
0	5e+05	27
1	3388	0

	Precision	Recall	F1-Score	Support
0	0.99	1.0	1	502970
1	0	0	0	3388

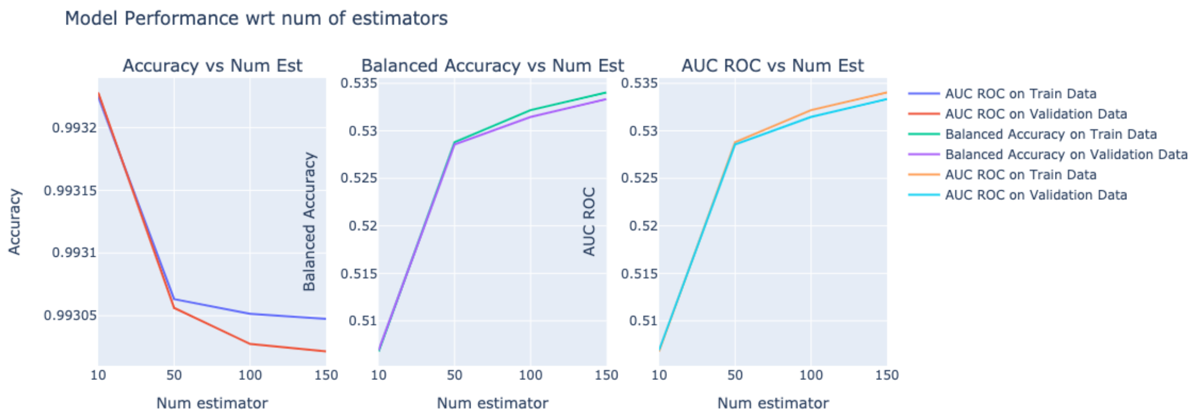
We also tried to tune regularization param C, however it doesn’t make sense to apply regularisation when the overall bias is very low.

Analysis of AdaBoost Classifier on Backorders Prediction Dataset

We applied AdaBoost classifier with base classifier as DecisionTreeClassifier with max_depth = 1, we first performed grid search to find optimal learning rate keeping num_estimators = 100. The optimal value we got for learning_rate is 0.05.



Keeping learning_rate = 0.05, we tried to find different values of num_estimators. The best value of estimator that we have got was 100.



Accuracy	0.855550815825957
Balanced Accuracy	0.850478050580261
AUC ROC	0.850478050580261

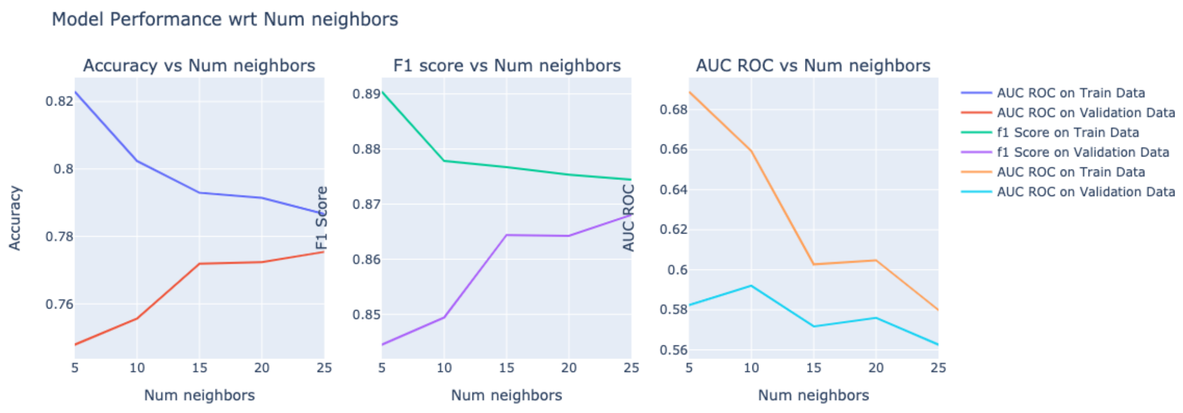
	Confusion Matrix	
0	430351	72619
1	524	2864

	Precision	Recall	F1-Score	Support
0	1	0.86	0.92	502970
1	0.04	0.85	0.07	3388

The AdaBoost classifier achieved balanced accuracy of 0.85 which is less than Decision tree classifier which is surprising. One possible reason for this behaviour is that the subsequent decision stumps are not contributing in the decision making on AdaBoost.

Analysis of KNN classifier on Backorders Prediction Data

The KNN took more than 6 hrs to train a single model on 5 fold cross validation for number neighbours [5, 10, 15, 20, 25]



Accuracy	0.845550815825957
Balanced Accuracy	0.840478050580261
AUC ROC	0.840478050580261

	Confusion Matrix	
0	420346	73615
1	524	2864

	Precision	Recall	F1-Score	Support
0	1	0.86	0.92	502970
1	0.04	0.85	0.07	3388

Improvements:

The BNP Paribas dataset has 131 predicted some of which are highly correlated to each other whereas some features are exhibiting some data distribution for both the classes. Feature selection algorithms would have helped us to only use high quality features by dropping noisy features. The categorical predictors were encoded using target encoding with Laplace prior instead entity encoding could have been tried to represent categorical data. The biggest challenge we faced was runtime of SVM and KNN. We were hesitant in applying SVM on large dataset knowing the fact that SVM doesn't scale. The performance of the KNN would have been drastically improved if we would have moved away for sklearn implementation to Facebook's faiss library

On the Backorders prediction dataset, under-sampling the majority class would have made a lot easier to train different classifiers. Since dataset was highly imbalanced, probability thresholding would have helped to main trade-off between precision and recall. The neural network performed horribly on both the datasets overfitting and predicting only a single class even with playing with embedding size and layers. We believe that by tweaking the neural networks objective function to consider high imbalance nature of data (objective functions like focal loss could help).