Machine Learning Assignment 1: Supervised Learning Assignment

Vinayak Raja

GTID: vraja30

Abstract

We implement and optimize five different supervised learning algorithms (decision tree, boosted trees, support vector machines, K-nearest Neighbors, and Neural Networks) on two distinct datasets with distinct properties: *Customer propensity to purchase dataset and Telcom Customer Churn*. Performance for different models are a compared in the analysis. The analysis is done in Python using Scikit learn library and shared with code base.

What makes these datasets interesting?

Dataset 1: Customer propensity to purchase dataset

A website gets many site visitors every day, but only a small percentage of them buy from website. We can learn a lot from certain actions and interactions of a valuable customer and apply those learnings to re-target or reach out to prospective customers by **optimizing our Marketing Dollars.**

Objective: *Based on a shopper's actions on a website to identify the best prospects who are likely to purchase; using Machine learning predicting their propensity to buy a product.*

- In order to understand purchase propensity of site visitors, we can look at their interactions with website and what is common path to conversion.

- This will help in re-targeting these folks using custom offers as they will more likely to purchase a product from us.

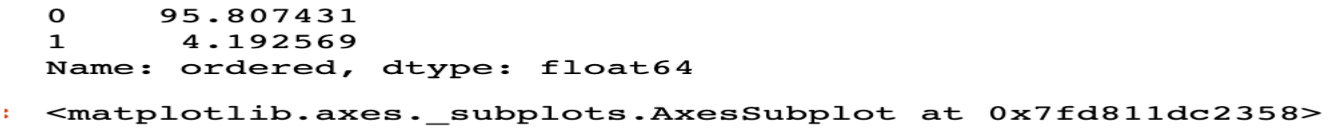
Data Info and source:

This data set represents a day's worth of visit to a fictional website. Each row represents a unique customer, identified by their unique UserID. The columns represent feature of the users visit (such as the device they were using) and things the user did on the website in that day. These features will be different for every website, but in this data a few of the features we consider are:

- **basket*add*detail:** Did the customer add a product to their shopping basket from the product detail page?
- **sign_in:** Did the customer sign in to the website?
- **saw_homepage**: Did the customer visit the website's homepage?
- **returning_user**: Is this visitor new, or returning?

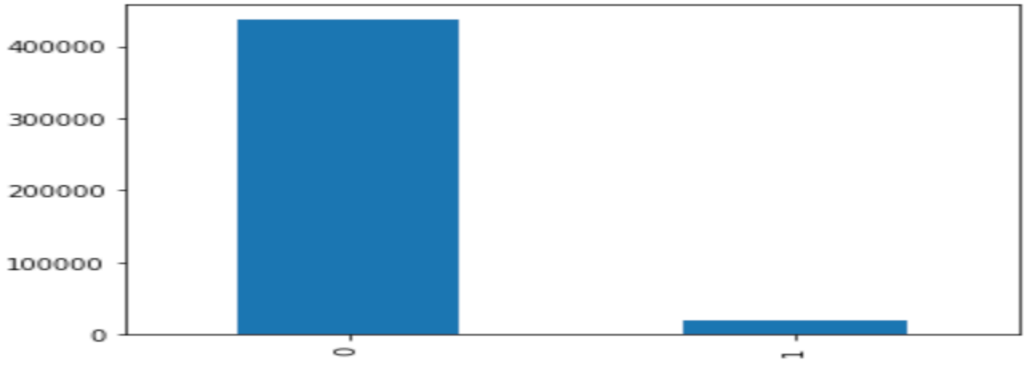
In this data set we also have a feature showing whether the customer placed an **order (ordered),** which is what we predict on.

Out of all site visitors only ~4% of total population made a purchase. This is high class imbalance problem and we can apply random under sampling to solve for this when we get into model building and using f-1 score for model evaluation.



```
0     95.807431
1      4.192569
Name: ordered, dtype: float64
```

```
: <matplotlib.axes._subplots.AxesSubplot at 0x7fd811dc2358>
```



Dataset 2: Telcom Customer Churn

Customer churn is the loss of clients or customers from a service. Churn is an important business metric for subscription-based services such **as telecommunications companies**. Due to the direct impact on revenues of the companies, subscription-based companies need to develop means to predict potential customer to churn. Therefore, finding factors that increase customer churn is important to take necessary actions to reduce this churn.

Objective: We will be using machine learning finding patterns in this data and predict likelihood of customers who are more likely going to churn.

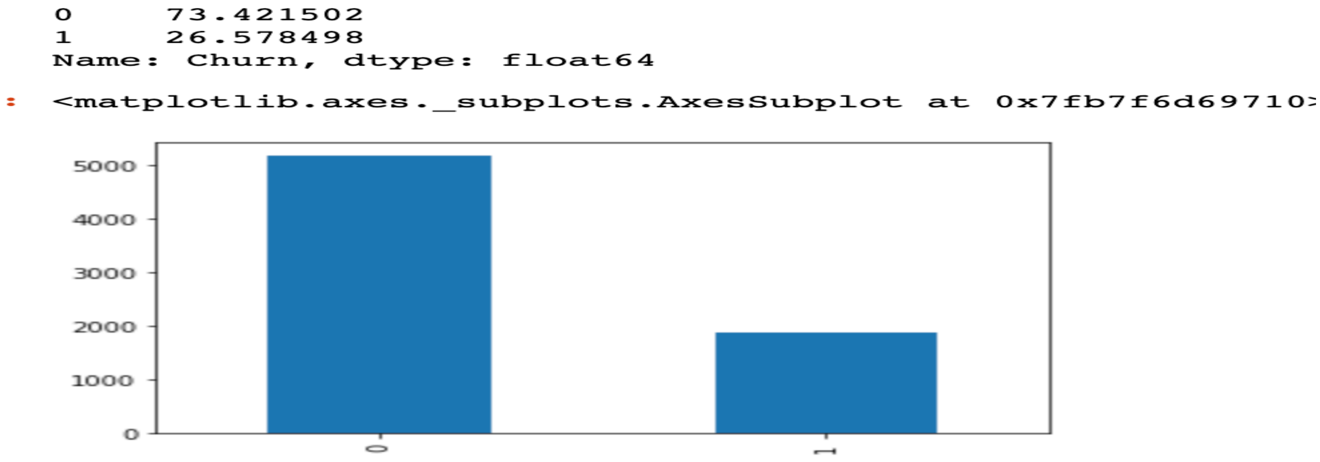
- This will help in reaching folks proactively who are going to churn and extend them an offer if they are valuable to the company.

Data info and source:

Each row represents a customer, each column contains customer's attributes described on the column Metadata. The data set includes information about:

- Customers who left within the last month – the column is called Churn
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies
- Customer account information – how long they've been a customer, contract, payment method, paperless billing, monthly charges, and total charges
- Demographic info about customers – gender, age range, and if they have partners and dependents

Out of all customers ~27% of total population churned. This is class imbalance problem and important to keep in mind for our modelling as skewness could lead to a lot of false negatives.



```
0     73.421502
1     26.578498
Name: Churn, dtype: float64
```

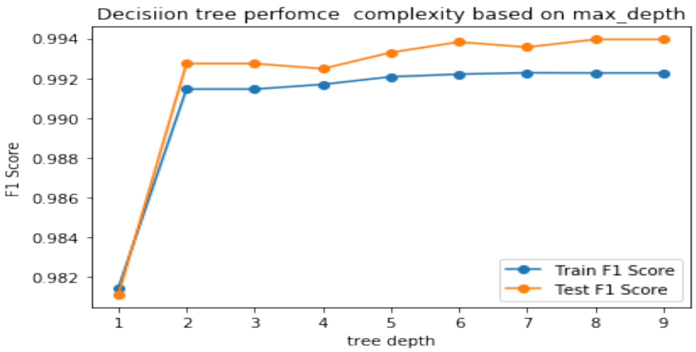
```
: <matplotlib.axes._subplots.AxesSubplot at 0x7fb7f6d69710>
```

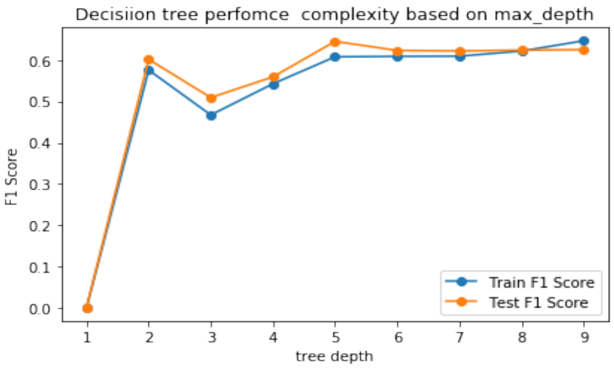
Modeling and Analysis Methodology:

- **Split the data:** The datasets are divided into two splits, where 80% of them are in training and 20% of them are kept in testing data(unseen data)
- **Random Under sampling for dataset1:** As we have minority class with only 4% representation, we decided to random under sampling for customer propensity dataset in order to not bias with Majority class.
- **One hot encoding:** We had only numeric variables based for dataset 1 but had a lot of categorical variables for Churn data, we applied one-hot encoding for this.
- **Evaluation metric:** Since both data are highly skewed and its class imbalance problem, we used **f-1 macro** as evaluation metric instead of accuracy in this classification task.
- **Hyperparameter Tuning:** We did Grid search on different hyperparameters for various algos to find best HP and evaluated on unseen test data
- **Analysis:** We looked at learning curve to understand how our model preformed and if they were overfitting and underfitting. Also, we looked at query time and training time to understand differences between different algorithms

Decisions Tree

Decision tree tend to overfit and one of the important parameters we should be using in order to prune the decision tree is **`max_depth`.** We also looked at **`min_sample_leaf`** the minimum no. of sample required to be a leaf node.

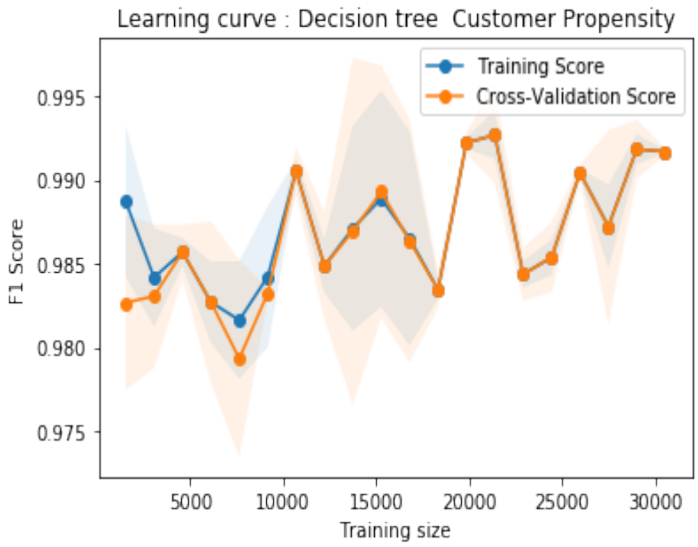


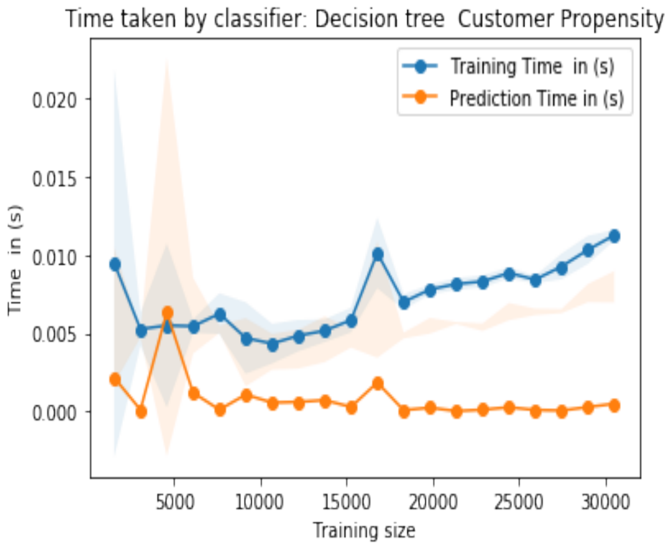


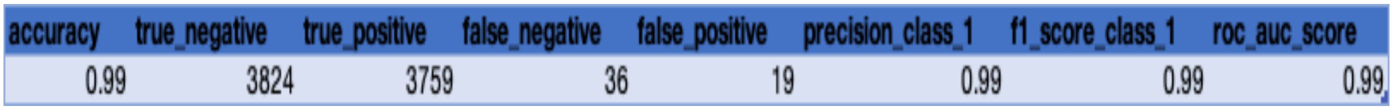
Model performance for dataset 1(left) and dataset 2(right), based on max_depth size. Its visble that as max depth increases model performance goes down.

Grid Search

Dataset 1: We found **2 as best max_depth and 153 as min_sample_leaf** during our grid search and used these parameters to evaluate our test data. **The learning curve shows that with more examples our DT learns better and there is very nice layover of training validation curves.**

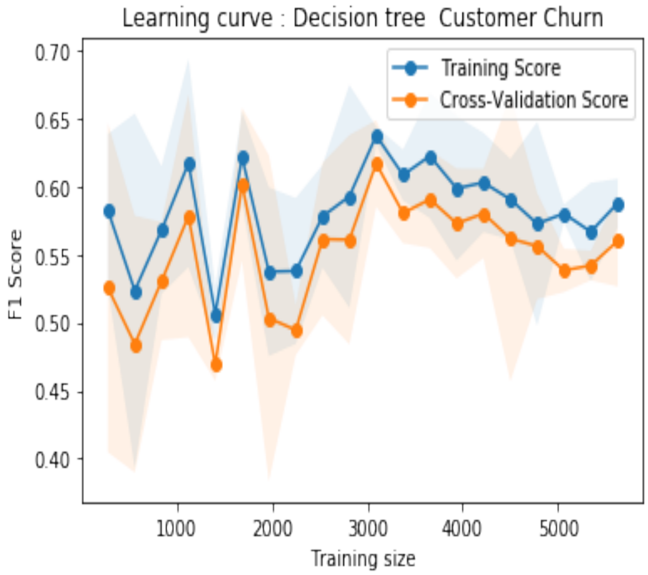


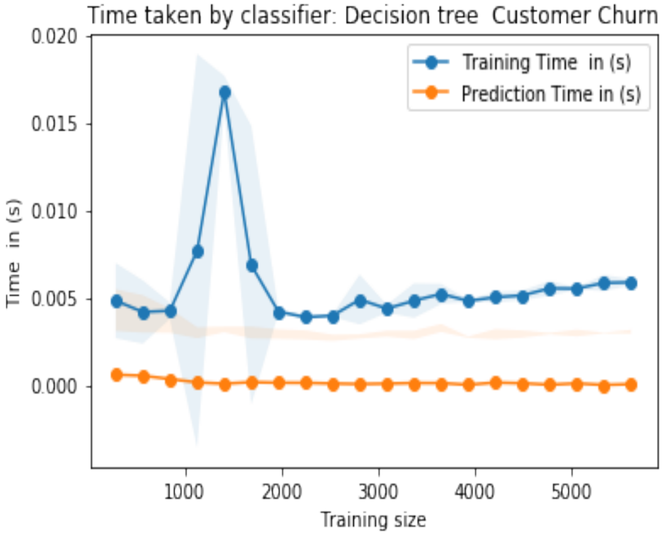


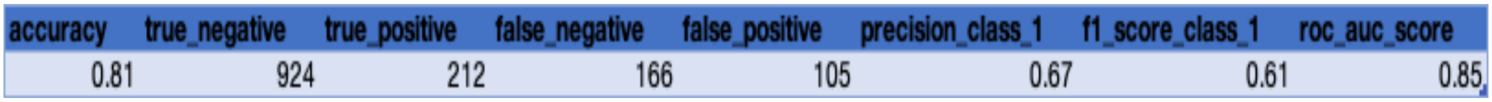


accuracy	true_negative	true_positive	false_negative	false_positive	precision_class_1	f1_score_class_1	roc_auc_score
0.99	3824	3759	36	19	0.99	0.99	0.99

Dataset 2: We found **7 as best max_depth and 41 as min_sample_leaf** during our grid search and used these parameters to evaluate our test data. The learning curve shows that with more examples our DT learns performance remains same and its very good fit on the data.







accuracy	true_negative	true_positive	false_negative	false_positive	precision_class_1	f1_score_class_1	roc_auc_score
0.81	924	212	166	105	0.67	0.61	0.85

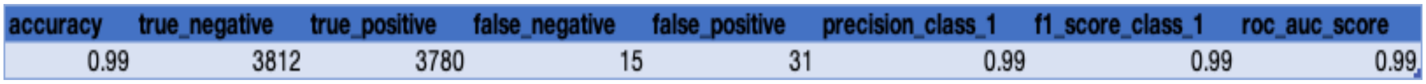
Gradient Boosted Trees

We used Gradient boosted trees using Sklearn. It builds the model in a stage-wise fashion and idea is to to improve upon the predictions of the multiple weak learners. We used similar hyperparameters as decision tree but optimized on two additional parameters `n_estimators`: number of weak learners and `learning rate` to find the best fit model that can minimize our error.

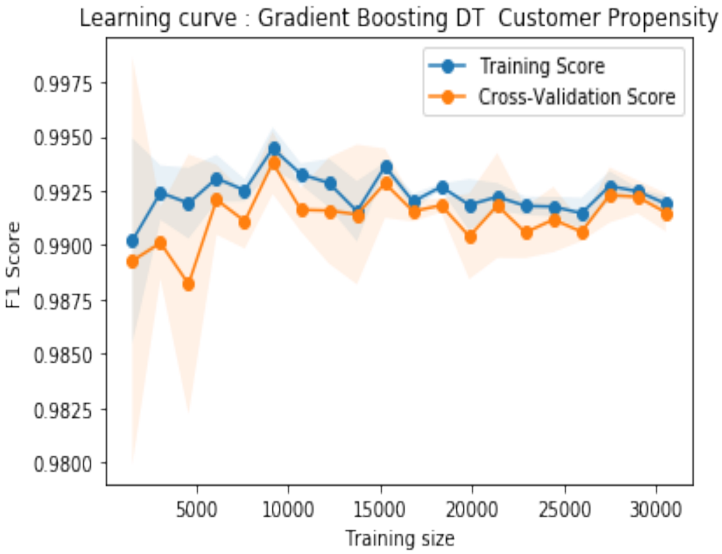
Dataset 1:
After performing grid search these were the HP that maximized our overall evaluation metric f1-score.

Best parameters are:
{'learning_rate': 0.1, 'max_depth': 3, 'min_samples_leaf': 298, 'n_estimators': 200}

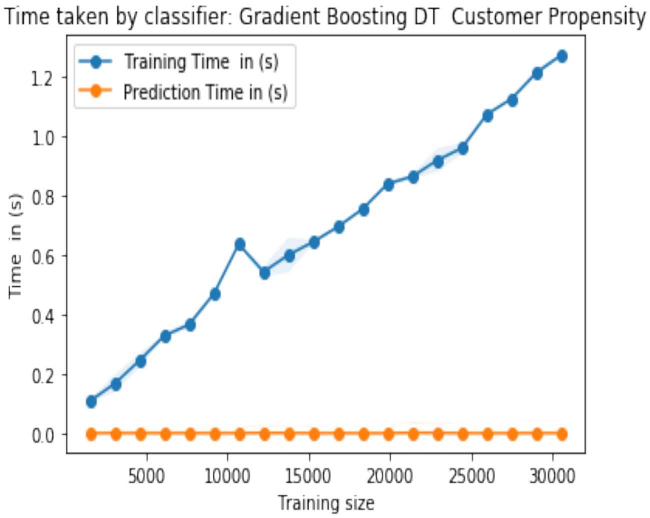
The learning curve shows that with more examples our gradient boosted tree learns better and minimizes overall error rate.



accuracy	true_negative	true_positive	false_negative	false_positive	precision_class_1	f1_score_class_1	roc_auc_score
0.99	3812	3780	15	31	0.99	0.99	0.99



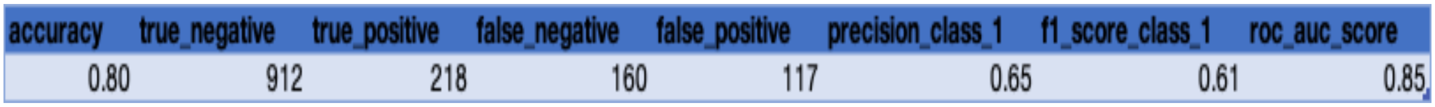
Learning curve : Gradient Boosting DT Customer Propensity



Time taken by classifier: Gradient Boosting DT Customer Propensity

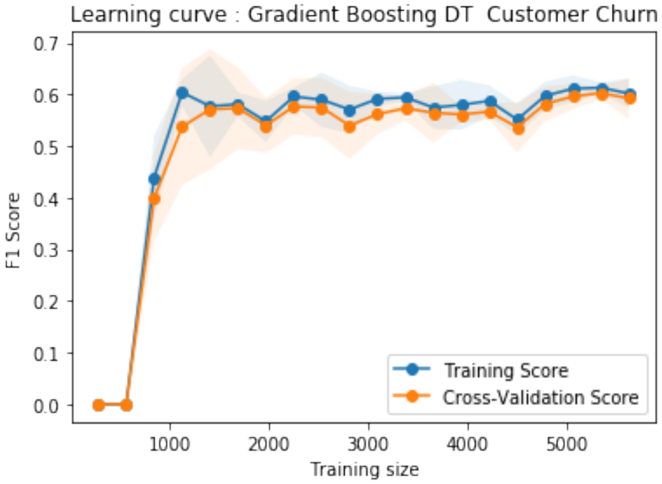
Dataset 2:
After performing grid search these were the HP that maximized our overall evaluation metric f1-score.

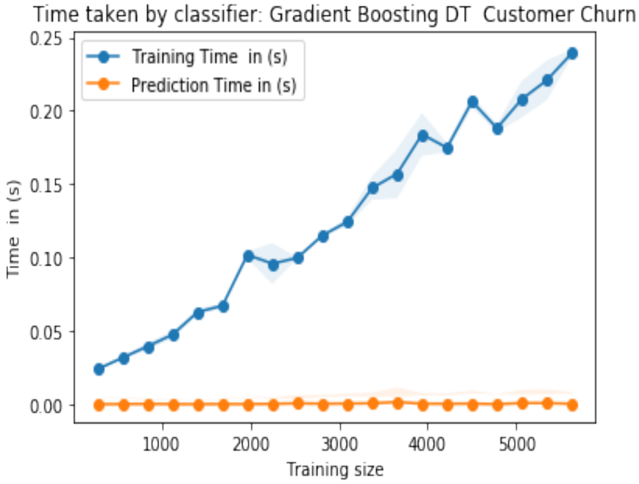
Best parameters are:
{'learning_rate': 0.05, 'max_depth': 4, 'min_samples_leaf': 201, 'n_estimators': 80}



accuracy	true_negative	true_positive	false_negative	false_positive	precision_class_1	f1_score_class_1	roc_auc_score
0.80	912	218	160	117	0.65	0.61	0.85

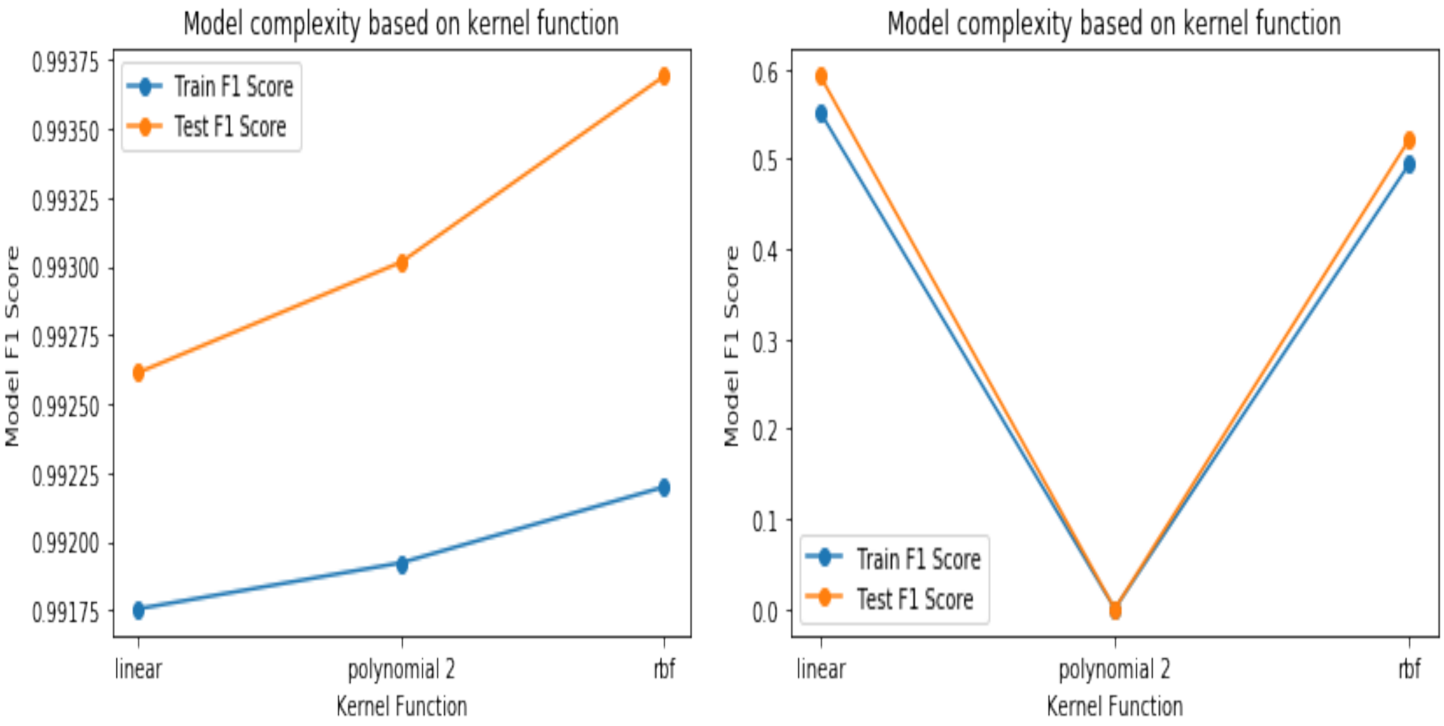
The learning curve shows that with more examples our gradient boosted tree learns better and minimizes overall error rate.





Support Vector Machines

We looked at multiple Kernels for SVM: kernel function: linear, polynomial, rbf (radial basis function)



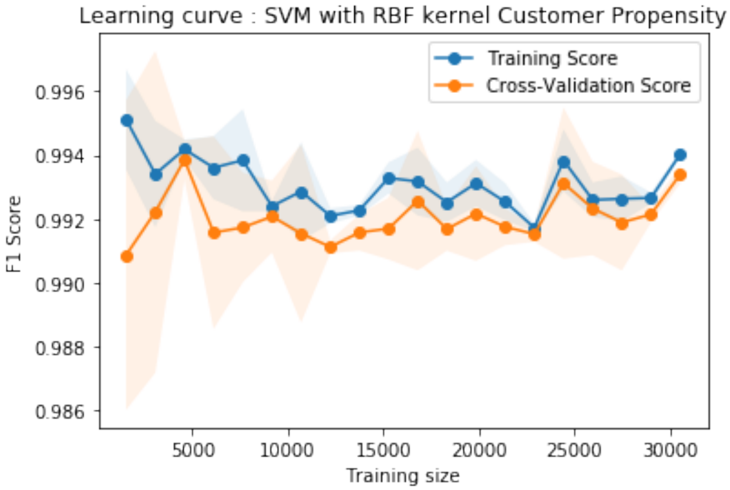
Model performance for dataset 1(left) and dataset 2(right), using different Kernel. Rbf beats in dataset1 and isat par with linear kernel for dataset 2.

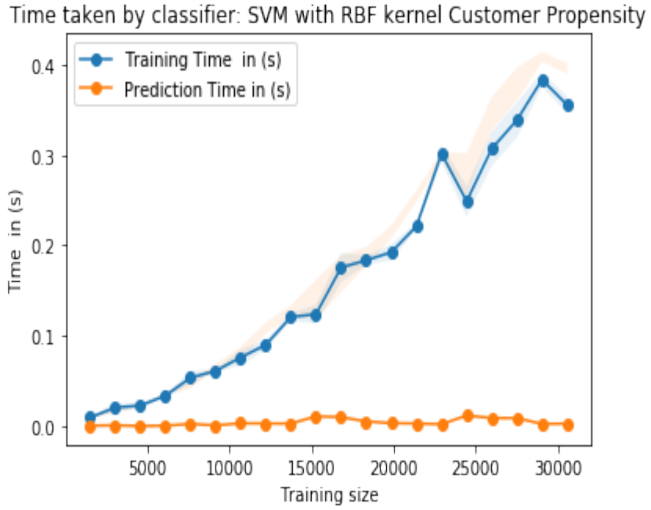
As we can see the 'RBF' kernel performs best for both the data set, we use this as our bese kernel and perform Grid grid srech on two Hyper parameters 'C' and 'gamma'for a nonlinear support vector machine (SVM).

C is a regularization parameter that controls the tradeoff between the achieving a low training error and a low testing error that is the ability to generalize your classifier to unseen data.

Dataset 1:
We get very nice learning curve with SVM compared to Decision three and GBT where training and validation curve were hugging each other.

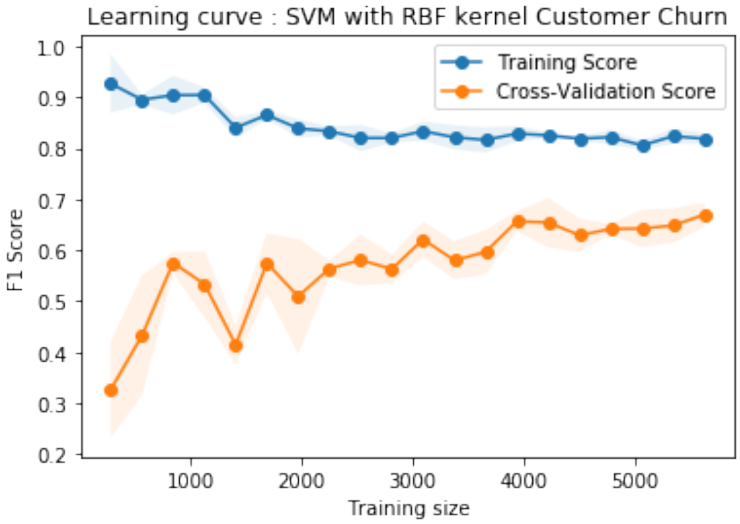
best parameters are:
{'C': 1, 'gamma': 1}

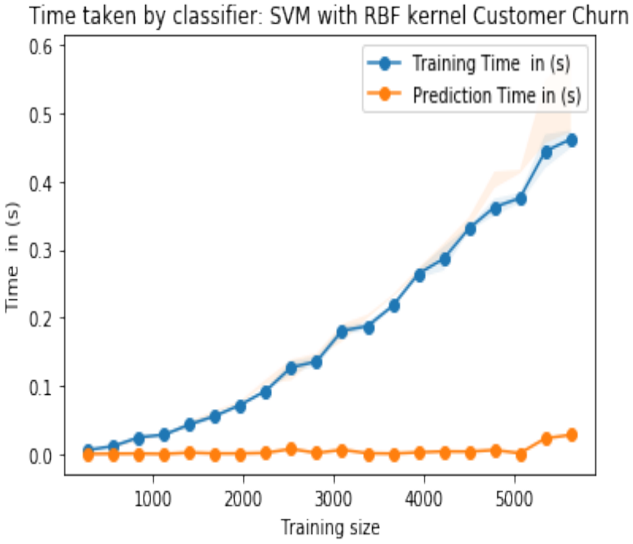




Dataset 2:
We see grid search produces similar values for HP for dataset 2. Compared to Gradient boosting we see a case of overfitting here for SVM model but overall f1-score is much higher compared to other models. It means SVM is able to draw better hyperplane but fail to generalize a bit on test data.

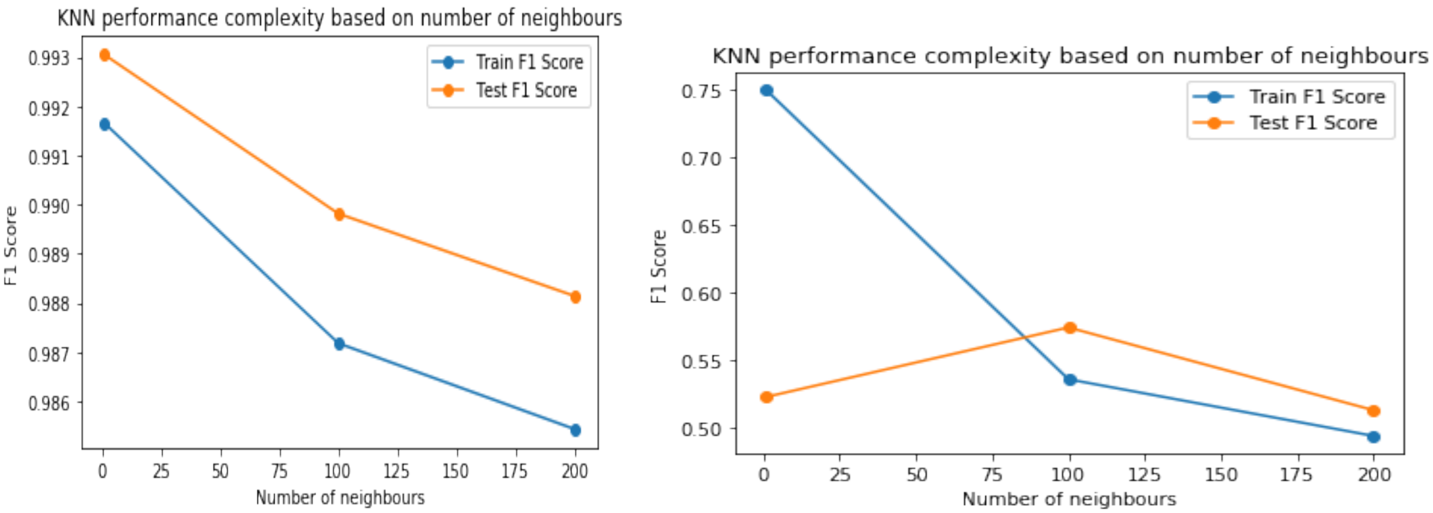
best parameters are:
{'C': 1, 'gamma': 1}





k-Nearest Neighbors

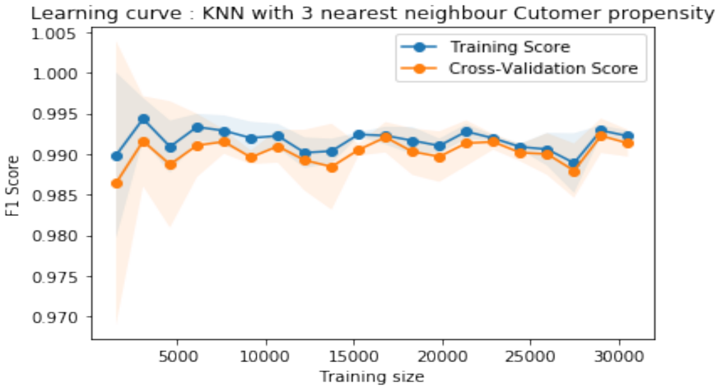
We have only one hyperparameter to train on which k meaning, this is one of the simple algorithm.

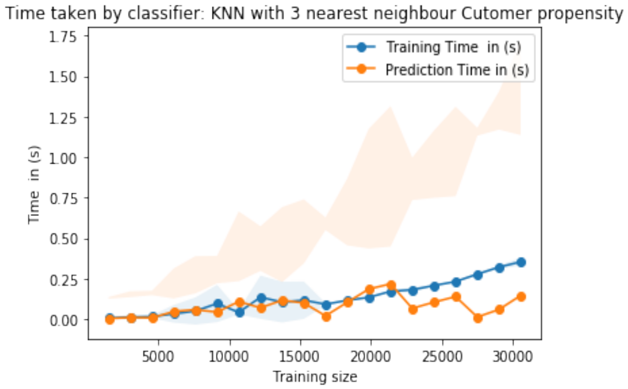


Model performance for dataset 1(left) and dataset 2(right), using different K

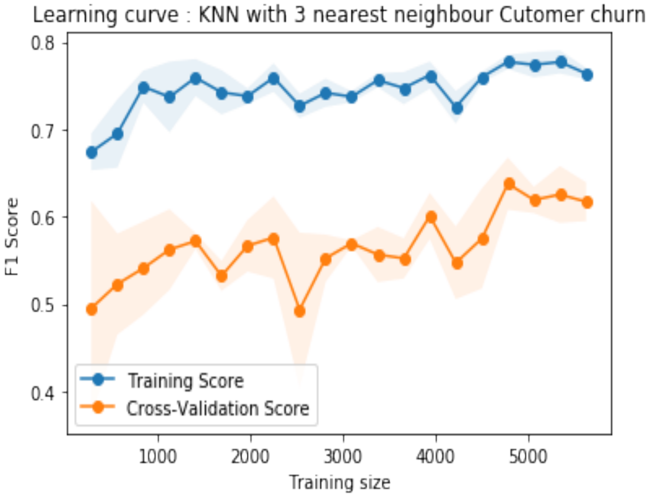
Dataset 1:

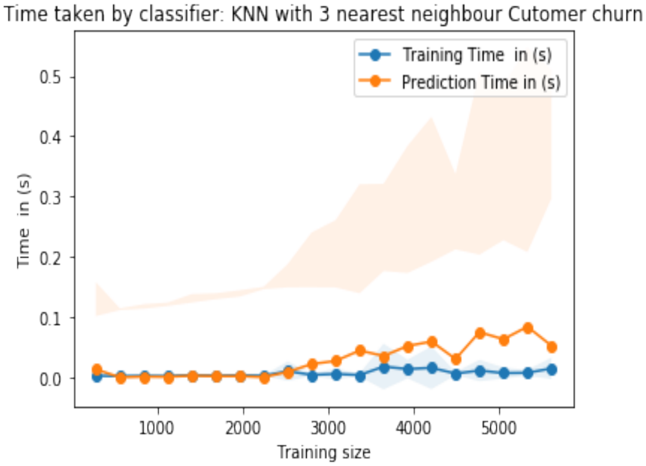
Overall we see similar trend for dataset 1 , as model is able to learn really well in all the othere models the perfomance is almot simlar on test data as well.





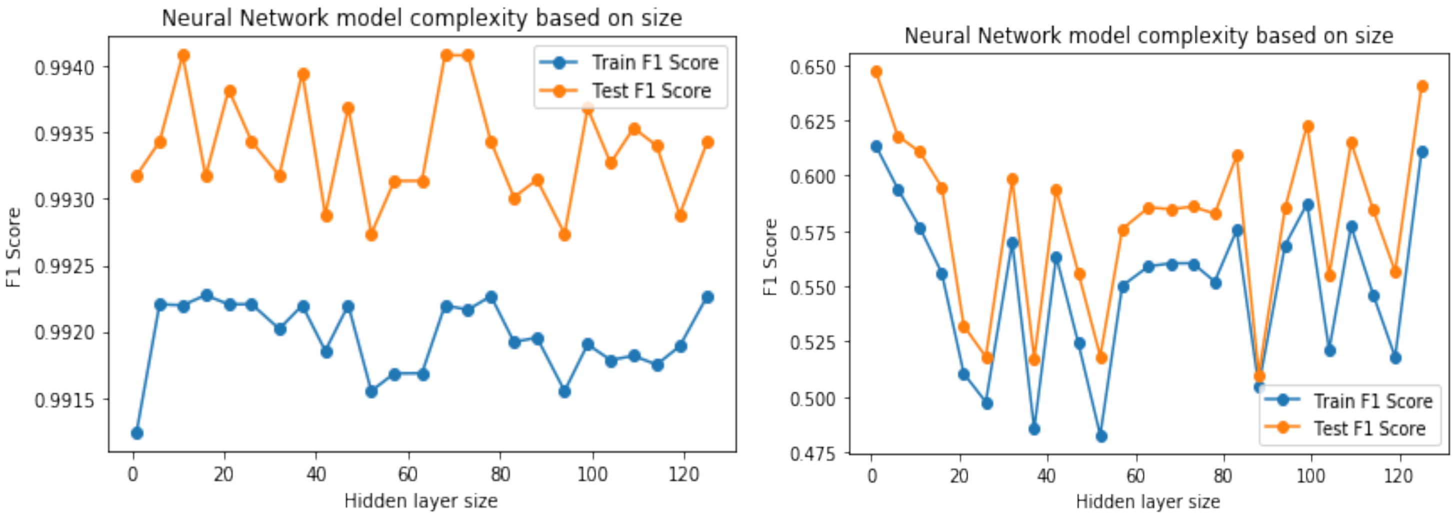
Dataset 2: Model goes through a lot of overfitting, training f1 score is high but its not able to generalize well.





Neural Network:

There are a lot of hyperparameters that can be trained on Neural network, but we focused on number of hidden units



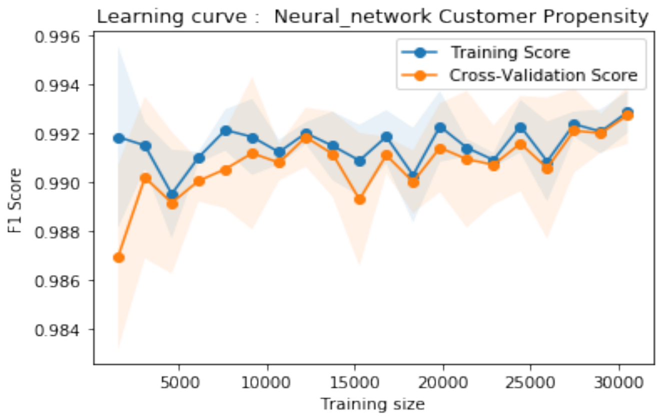
Model performance for dataset 1(left) and dataset 2(right), based on size of hidden layer. There's seems to be a lot of variation in this and best way to find the best hyper parameter will be to do grid search.

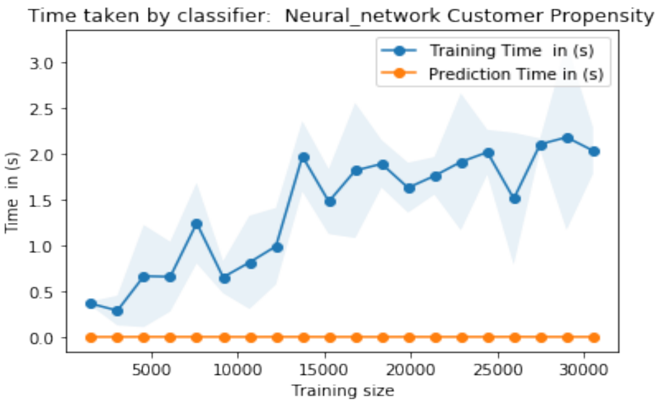
We look at two hyper parameter for neural network , 'hidden_layer_sizes' and 'learning_rate' to optimizae our neural network

Dataset 1:

Overall Neural network takes a lot of time to train and performance wise its at par with other algorithm for data set 1. Looking at Learning curve, we don't see a problem of overfitting which is the case with almost every algorithm in dataset 1.

best Hyperrameters after 3-fold cv in grid search:
{'hidden_layer_sizes': 50, 'learning_rate_init': 0.01}

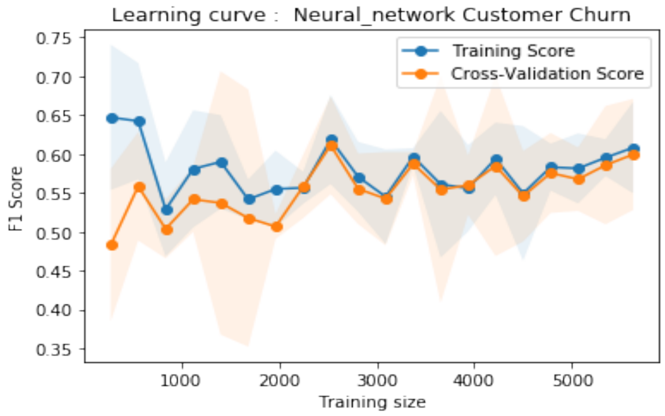


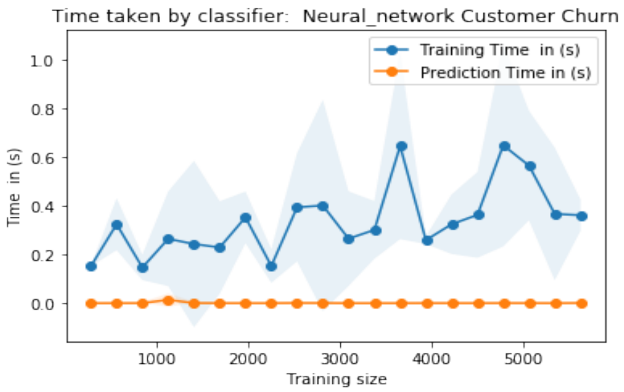


Dataset 2:

We see good performance with Neural network compared to KNN and SVM and with more data overfiiting seems to reduce for this algorithm.

best Hyperrameters after 3-fold cv in grid search:
{'hidden_layer_sizes': 60, 'learning_rate_init': 0.01}



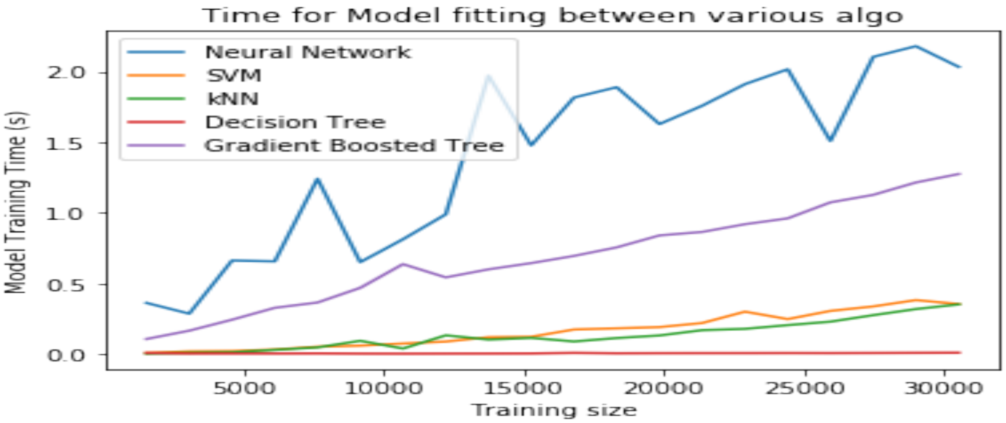


MODEL COMPARISON:

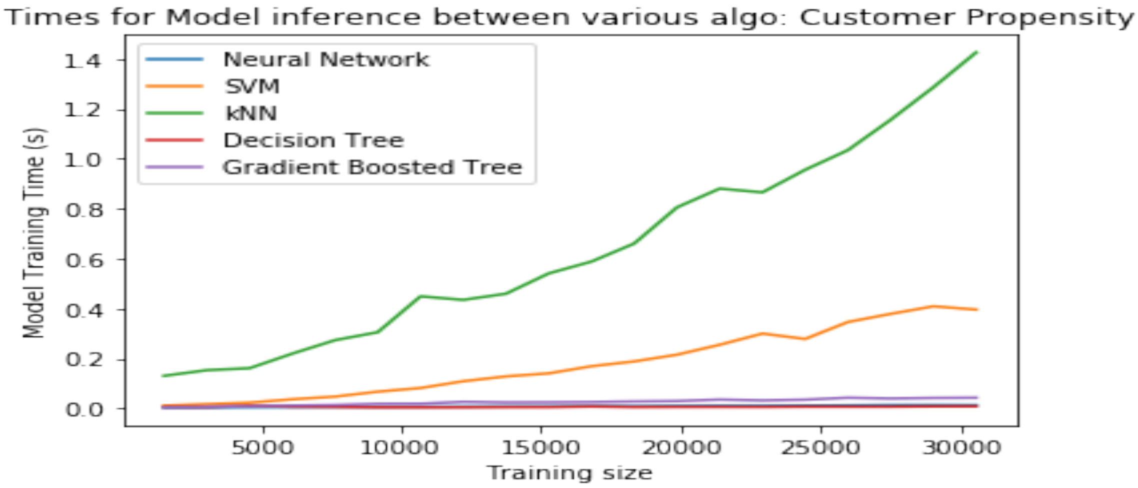
We primarily looked at three things to evaluate and compare modes, a. Training time b. Inference time c. Model metric on 3-fold Cross validation data. Based on this we created this rubric to help understand better.

Dataset 1

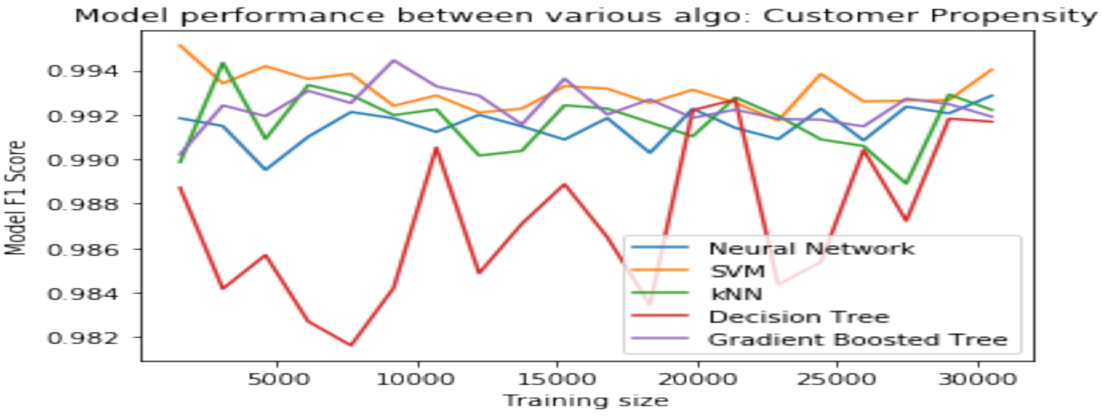
Model Fitting time: Neural network takes the most time to train followed by gradient boosted trees.



Model Inference time: KNN is most time consuming followed by SVM

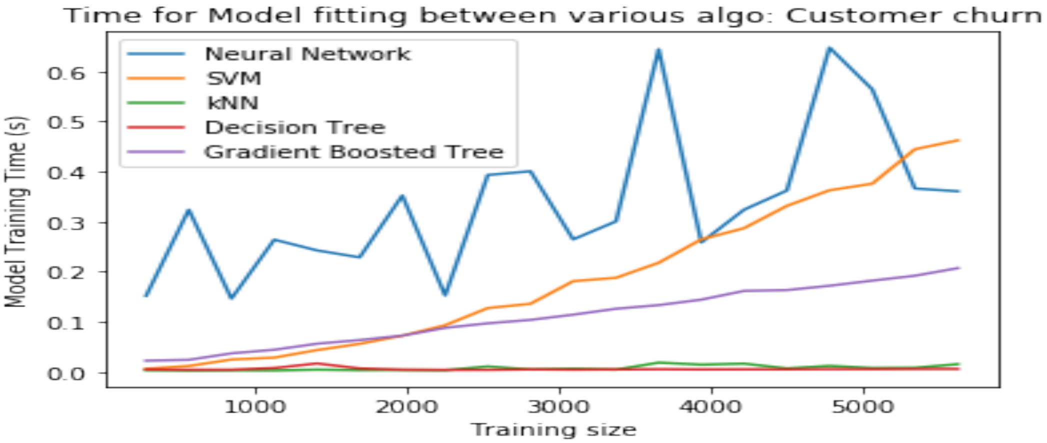


Model Performance f1 score: For dataset 1, all algorithms perform equally well but Decision tree is an outlier here in the bunch.

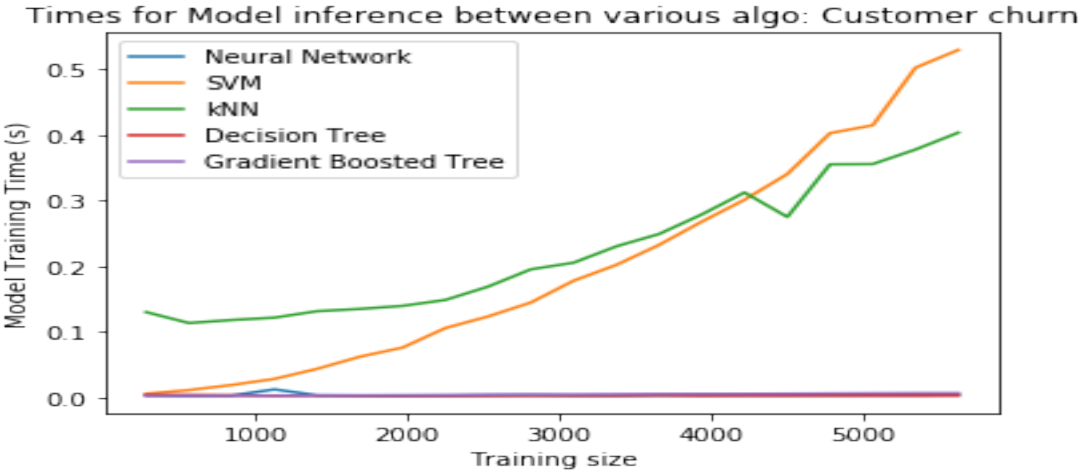


Dataset 2:

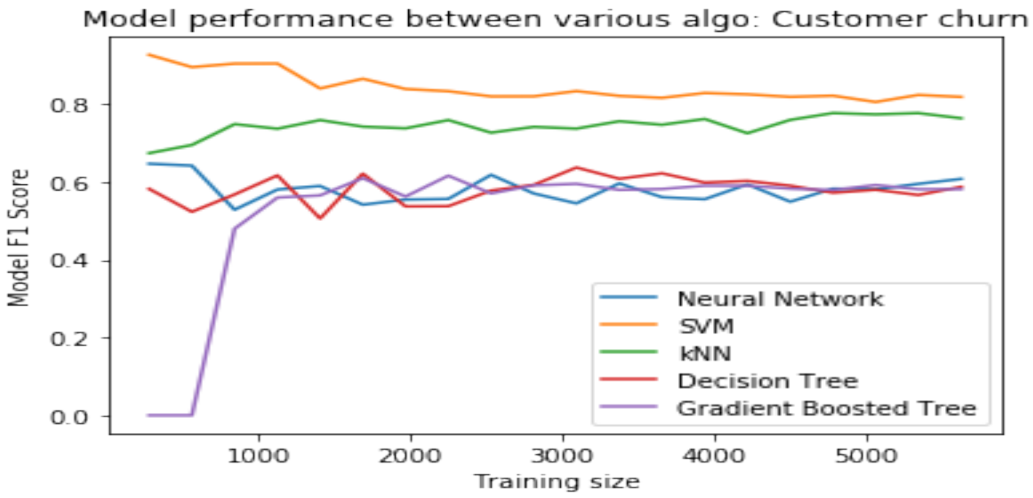
Model Fitting time: Neural network takes the most time to train followed by gradient boosted trees.



Model Inference time: KNN and SVM both have high inference time



Model Performance f1 score: As seen above that SVM gets f1 score on the churn dataset compared to other dataset, we may need to add more to improve predictive performance of other algorithms



Based on above analysis, we can conclude the following table that helps us to form baseline:

Algorithm	**Evaluation metric performance**	**Training Time**	**Inference time**	**Model complexity**
KNN	Low/Medium	Quick	Slow as K goes up	Not complex
Decision trees	Medium	Quick	Quick	Simple but bit more complex compared to KNN
SVM	High	Slow	Can be slow based on kernel	Depending on kernel, can be complex
Gradient boosting	High	Slow	Quick	Complex
Neural networks	high	**Slow**	**Quick**	Very complex(blackbox), especially DNN

References

1. **Plotting learning Curve:** https://scikit-learn.org/stable/auto_examples/model_selection/plot_learning_curve.html
2. **SVM kernels:** https://data-flair.training/blogs/svm-kernel-functions/#:~:text=SVM%20Kernel%20Functions,it%20into%20the%20required%20form.&text=These%20functions%20can%20be%20different,(RBF)%2C%20and%20sigmoid.