Machine Learning Analysis – Compare kNN and Decision Tree Performances

Author: Qi Li

I. Description

Classification problem 1: Marketing

The goal of the analysis is to determine how different factors affect whether the marketing campaign of a Portuguese banking institution is successful or not. If successful, the bank's term deposit will be subscribed and it will have more stable deposits for further investments and thus increases its revenue. It's interesting because I receive a lot of marketing phone calls from banks and am very curious about what type of people will tend to say subscribe and how different factors like age will affect the subscription decision. Similar analysis could be applied to different marketing campaigns to reach their target customers more efficiently.

Classification problem 2: Wine

The goal is to analyze how physicochemical properties determine whether the wine is high-quality. It's interesting because for people who know little about wine, they can determine whether the wine is high-quality by checking the physicochemicals and decide whether to purchase the wine.

II. Detailed analysis with machine learning model

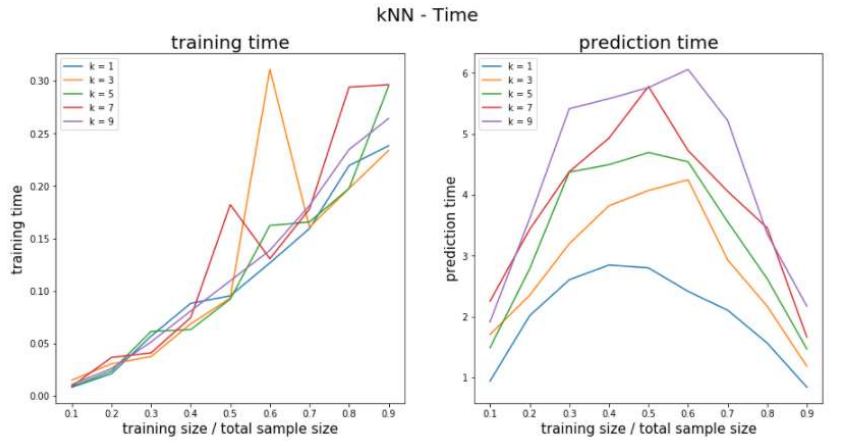
1. **K-Nearest Neighbors with various k**

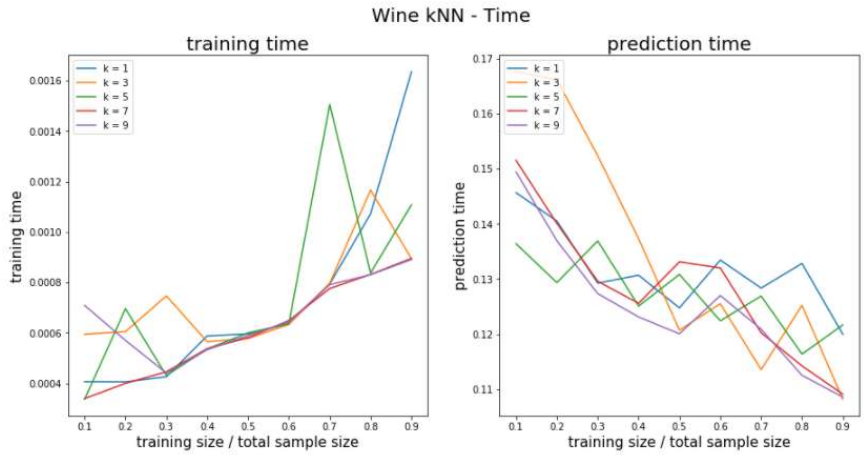
   Weighting nearest neighbors by inverse of Manhattan distance yields the best scores.

   Performance: Time

   As training size increases, training time increases, but prediction time increases then decreases. From marketing data increase of k doesn't affect training time but increases prediction time.

   Also We can tell from the graphs that training time is significantly lower than prediction time.

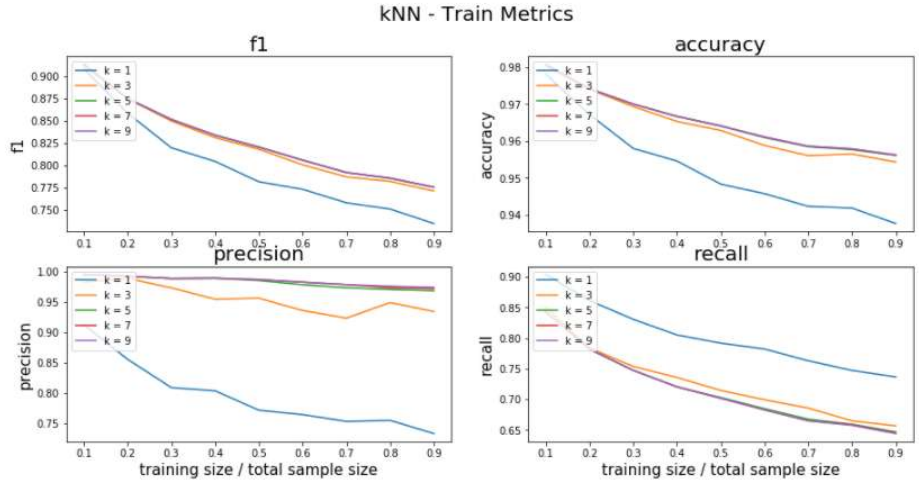


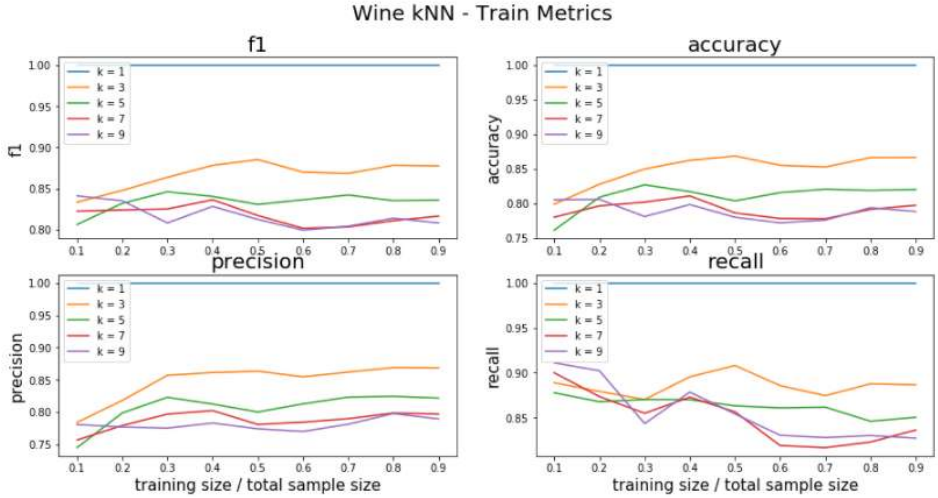


   Performance: Training

For marketing data, as training size increases, all metrics decrease for training data. As k increases, f1, accuracy and precision increases, but recall decreases.

For wine, k=1 always yields the best result.

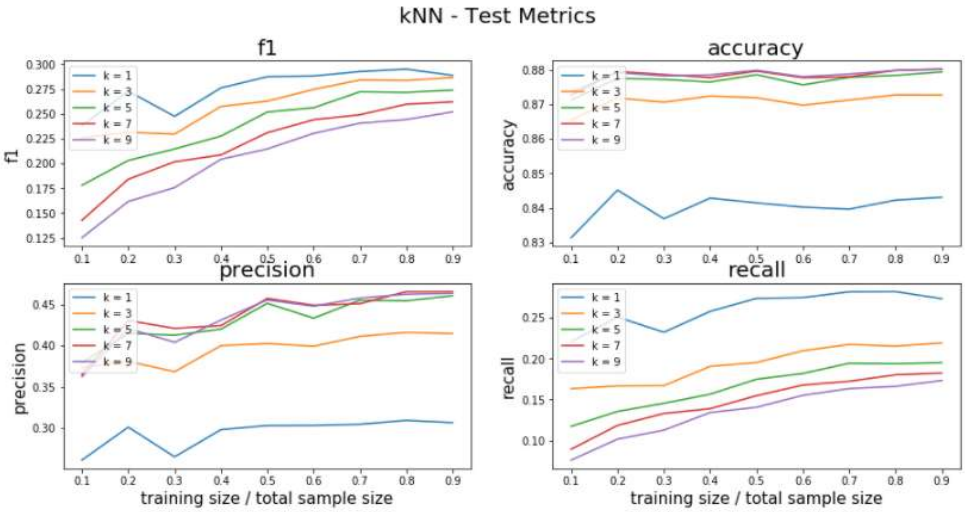


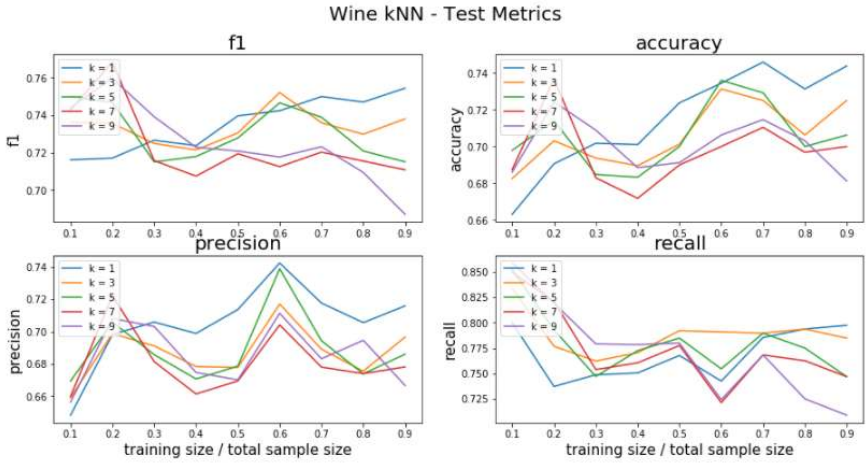


Performance: Test

For marketing, as training size increases, all metrics increase then stable. As k increases, accuracy and precision are higher, but f1 and recall are lower.

But for wine, k=1 yields the best f1, accuracy and precision when training size is large.





2. Decision Tree with various max depth

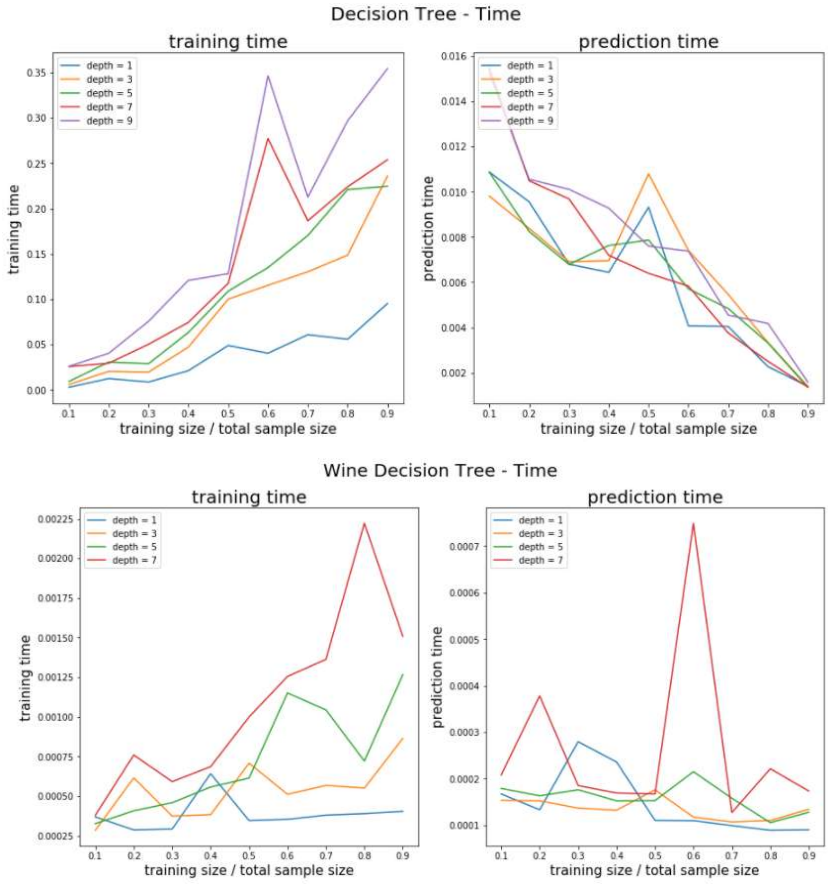
- Criterion: gini impurity instead of entropy because entropy is significantly more complex than gini computationally and can take a lot of time in training
- Splitter: the strategy is random instead of best because it's computationally simple
- Min_samples_leaf: the minimum number of samples required to be at a leaf node. I choose 1 because according to comparison, 1 is significantly better than choosing 2.

Performance: Time

As training size increases, training time increases, and prediction time decreases.

As max depth increases, training time increases, higher max depth generally takes more time to predict than lower max depth.

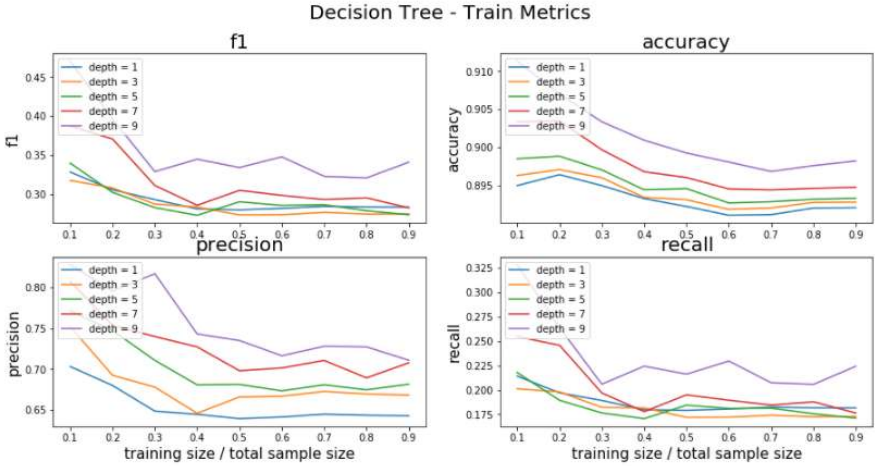
Also we can see prediction time is significantly smaller than prediction time.

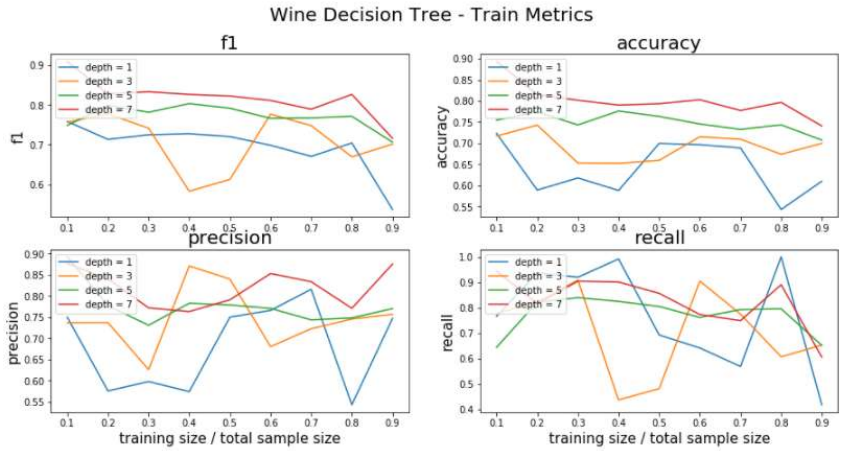


Performance: Training Metrics

For marketing data, as training size increases, all 4 metrics decreases, but becomes stable later. When max depth increases, all 4 metrics is higher and better for training data.

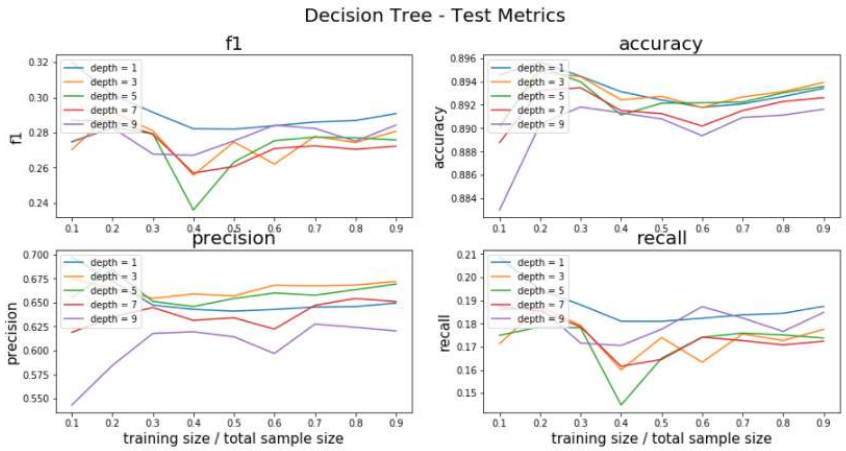
For wine data, it's more volatile but higher depth generally yields higher f, accuracy, precision and recall, but increasing training size is not improving the metrics.

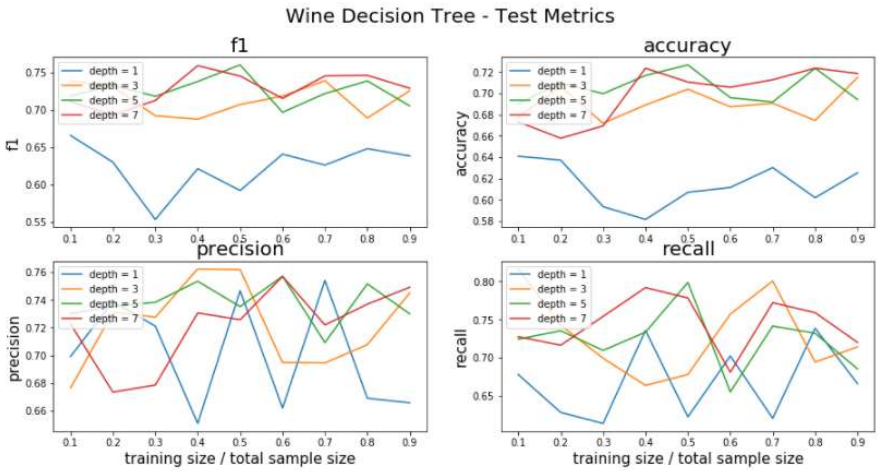




Performance: Test Metrics

- Accuracy: higher depth, lower accuracy for marketing data. But for wine data, higher depth, higher accuracy.
- F1, precision, recall: Only when depth is much larger, precision is lower, recall is lower, and f1 is lower.





3. **Boosting**

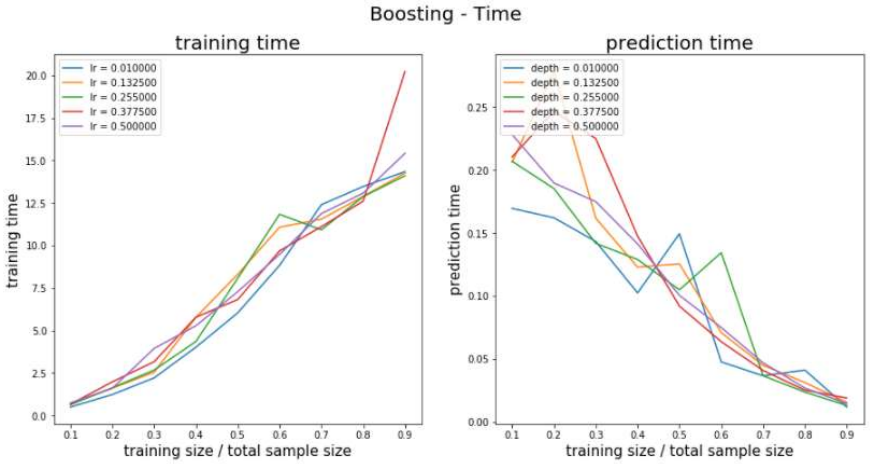
To build the boosting decision tree, I introduce the below parameters:

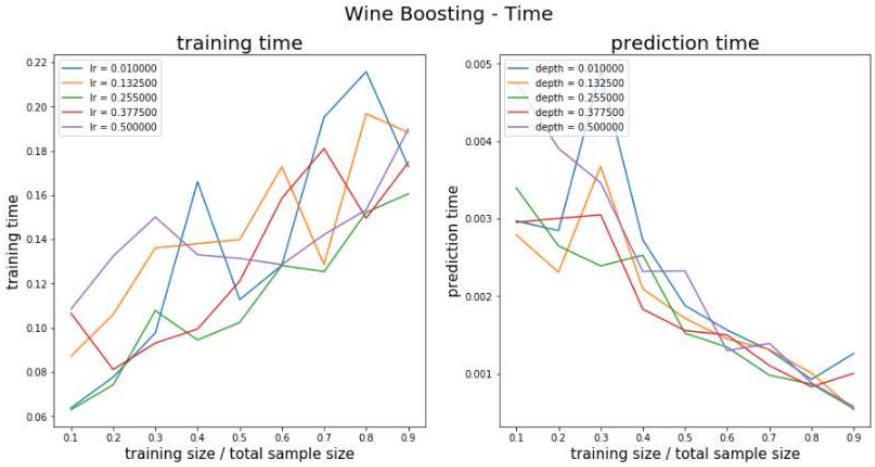
- Max_depth: on the analysis above on decision tree, I choose 1 for marketing and 3 for wine.
- N_estomator: number of trees to add in this model. Using 300 yields the best scores.
- Learning_rate: Contribution of each tree. Below lr in graph label represents learning rate.

Performance: Time

As training size increases, training time increases and prediction time decreases.

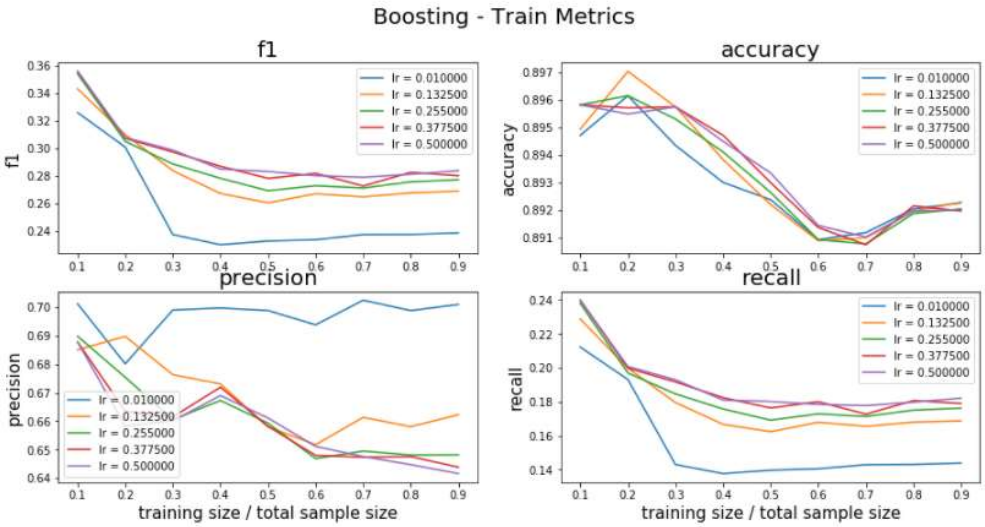
As learning rate increases, there is no significant differences in the relationship.

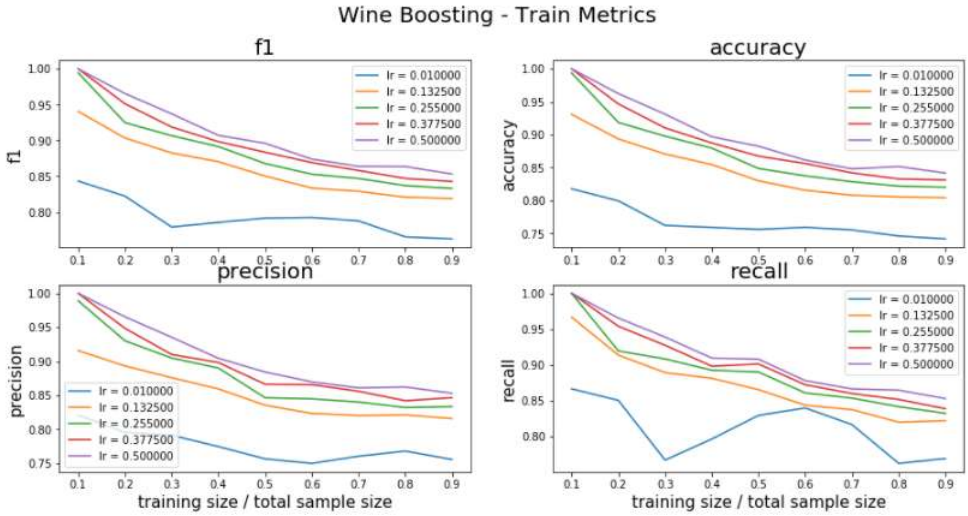




Performance: Training Metrics

- F1: As training size increases, f1 drops first and maintains a steady value. As learning rate increases, f1 increases as well.
- Accuracy: As training size increases, accuracy's main trend is dropping. For marketing it's hard to tell the difference but for wine date, higher learning rate yields higher accuracy.
- Precision: As training size increases, precision has a tendency to decrease. For marketing data lower learning rate is better but for wine higher is better.
- Recall: As training size increases, recall decreases then becomes stable. As learning rate increases, recall increases.





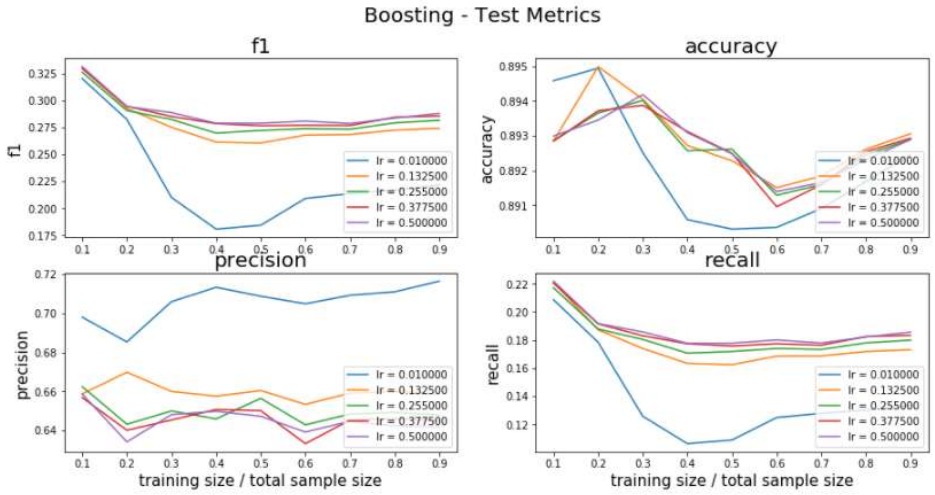
Performance: Test Metrics

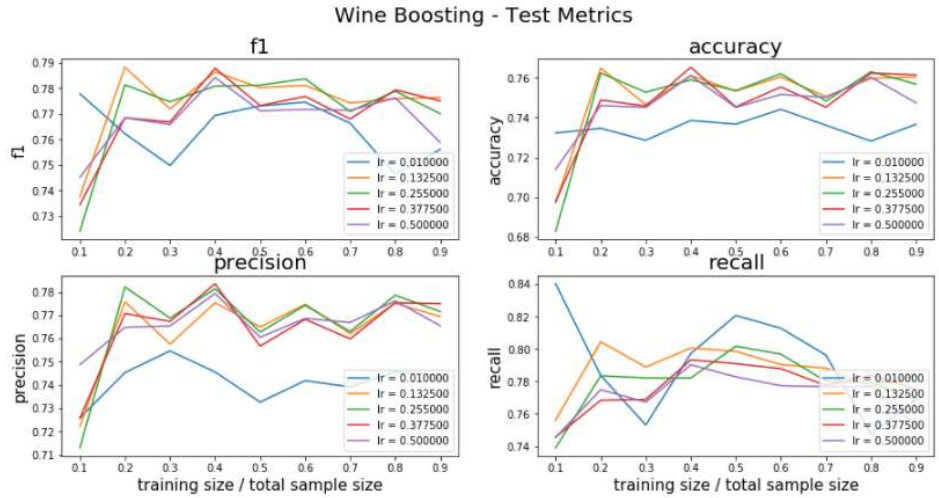
For marketing data:

- F1 and recall: As training size increases, f1 and recall drops first and maintains a steady value. As learning rate increases, f1 and recall increases as well but increases more and more slowly.
- Precision: As learning rate increases, precision decreases.
- Accuracy: It bumps with training size and do not have a trend. When learning rate is lowest, accuracy is lowest. Other learning rates have similar accuracy.

For wine data:

Training size doesn't have apparent impact. Higher learning rate usually yields better results.





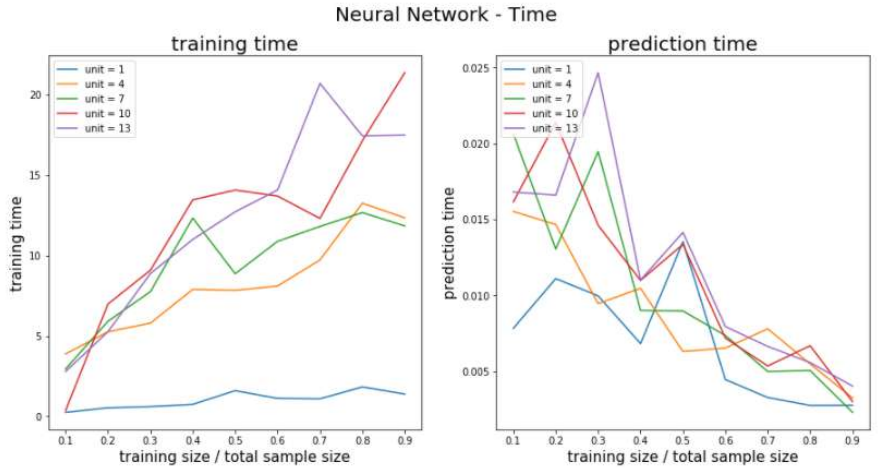
4. **Neural Networks**

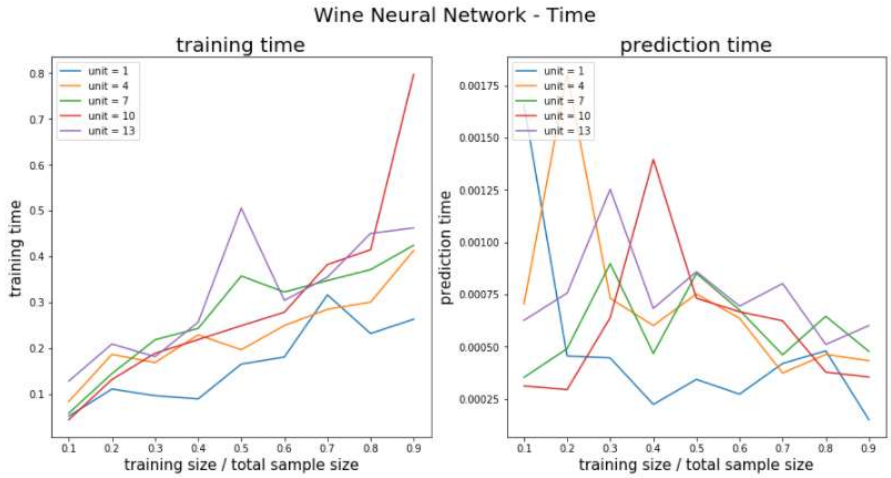
- Activation: tanh, the hyperbolic tan function
- Solver: sgd, which refers to stochastic gradient descent
- Random_state: 1 because this runs the fastest
- Hidden_layer_size: I change the parameter to compare how the number of hidden units will affect the results. The label unit in below chart refers to hidden unit.

Performance: Time

As training size increases, training time increases and prediction time decreases.

For marketing data, as size of hidden units increases, training time and prediction time increases. For wine data, training time increases as well but prediction time is bumpy but generally shows a trends of decreasing.

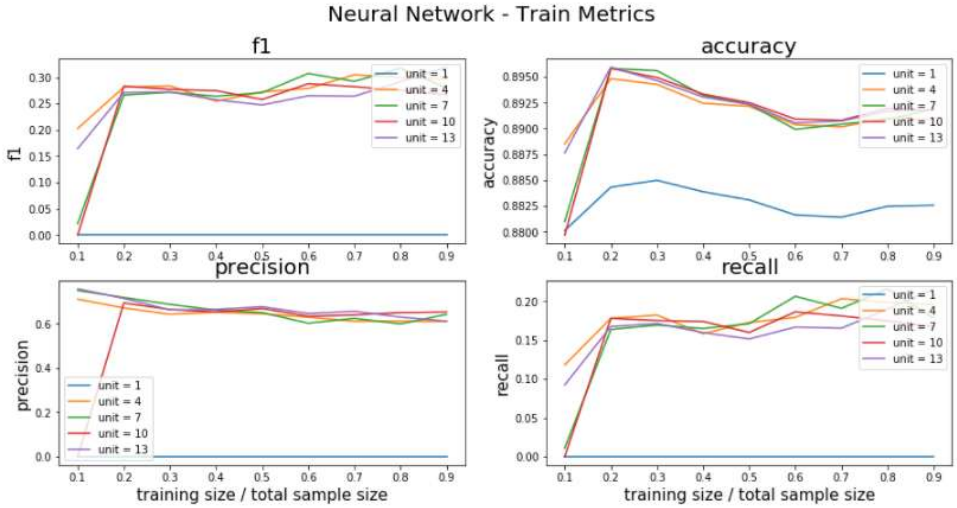


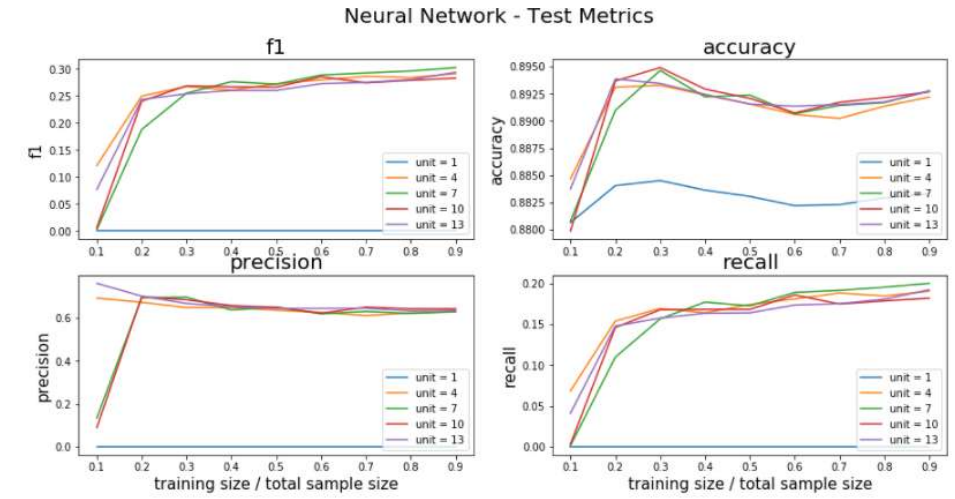


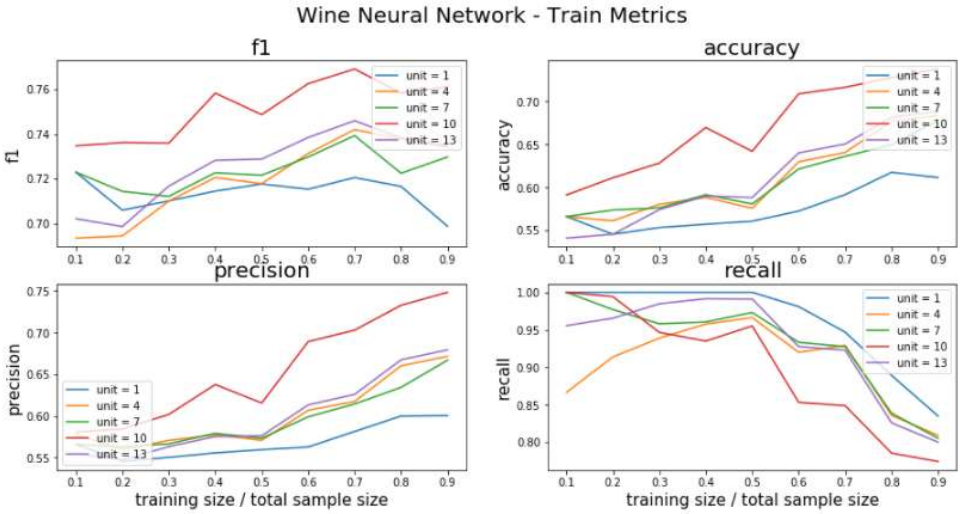
Performance: Training Metrics AND Test Metrics

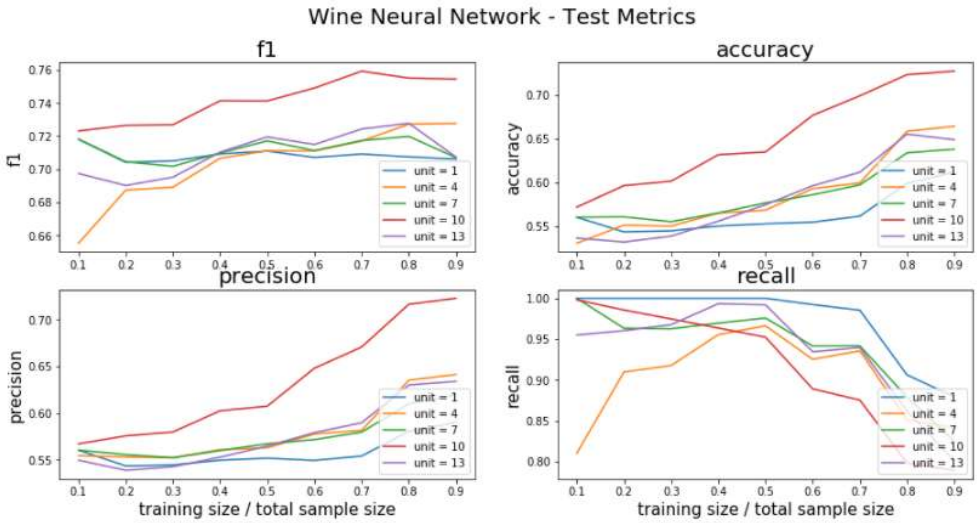
For both training and testing in marketing data, and for all 4 metrics, when training size increases, they increase a lot at first become relatively stable. Increase of hidden unit size increases all metrics, but after some level they do not distinguish.

For both training and testing in wine data, accuracy, f1 and precision increases with training size but recall the opposite. Increase of hidden unit size increases all metrics but 10 is the best and an increase further decreases the scores.









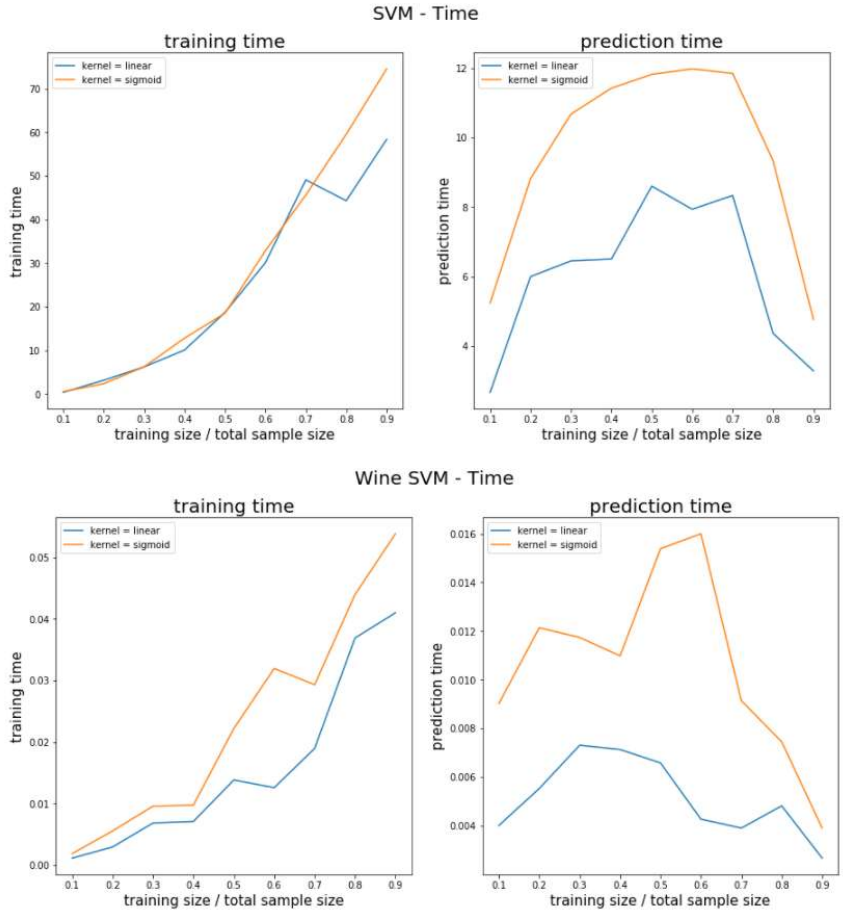
5. **Support Vector Machines**

To build neural networks, I choose 2 kernels, linear and sigmoid. Linear refers to using a linear hyperplane while sigmoid is similar to a 2-layer perceptron model of neural network.

Performance: Time

For training time, as training size increases, it increases for both kernels. But when training size is large enough, sigmoid kernel takes more time than linear.

For prediction time, as training size increases, it increases and then decreases. Sigmoid is always much higher than linear because it's computationally more complex.



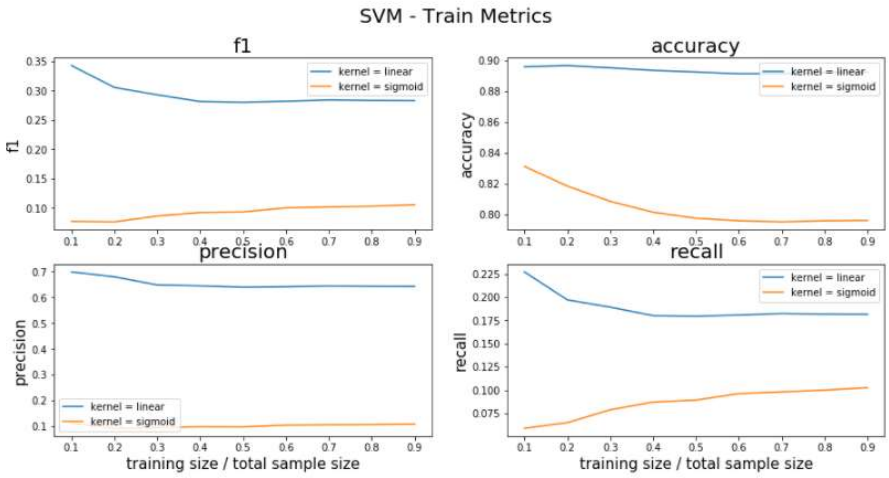
Performance: Training Metrics

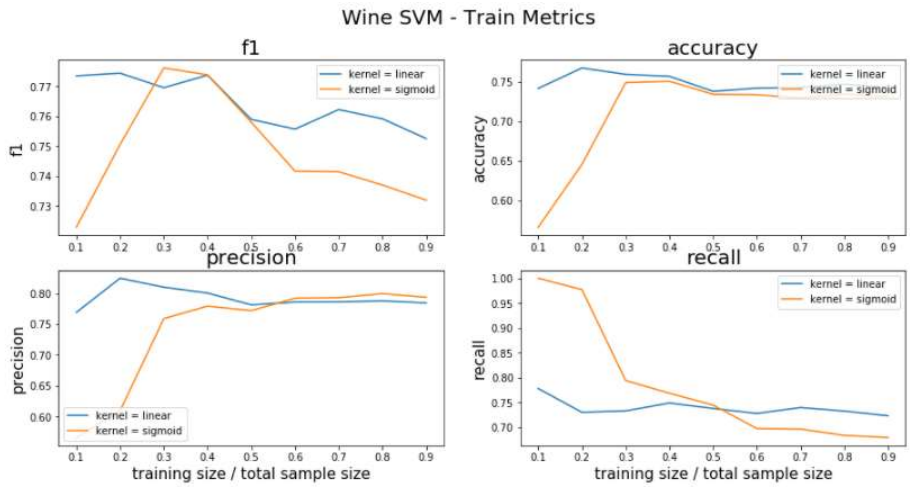
For marketing data:

For all 4 metrics, linear is much higher and therefore better than sigmoid. As training size increases, linear kernel has slight decreases and becomes stable, sigmoid slightly increases and becomes stable.

For Wine data:

As training size increases, metrics value decreases. Recall is better for sigmoid when training size is small.

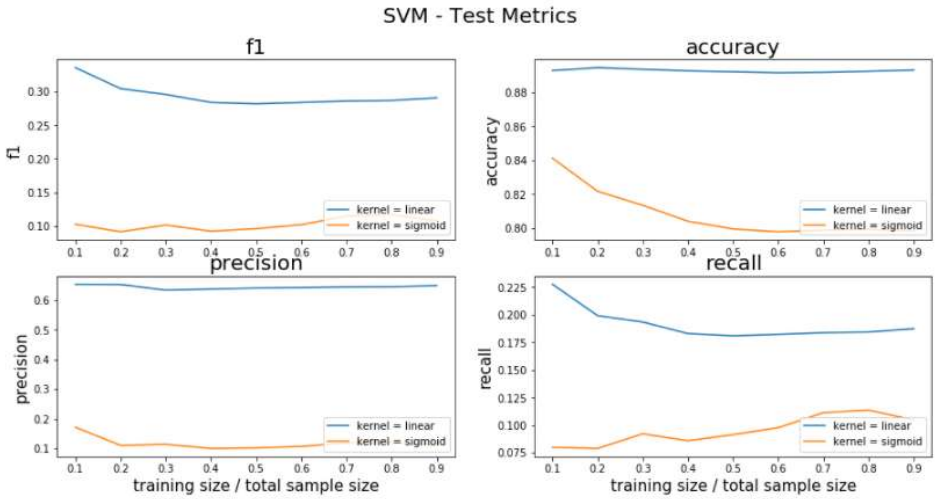


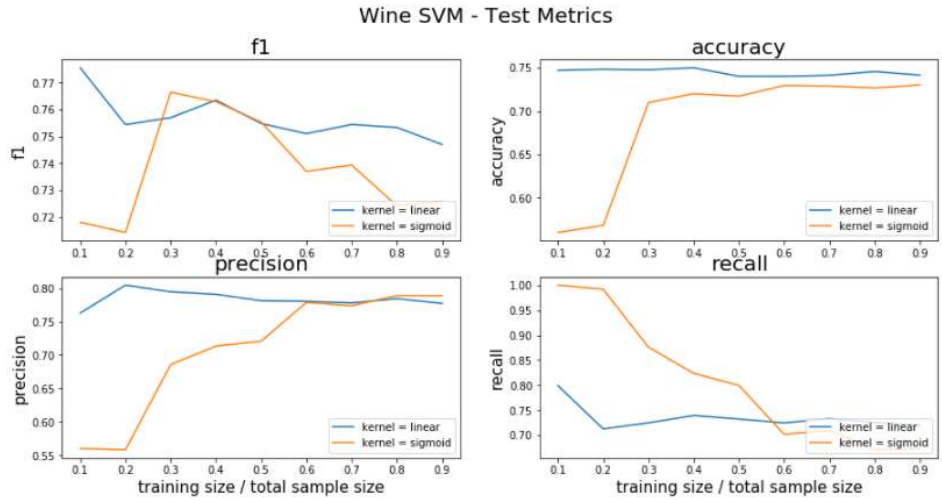


Performance: Test Metrics

For f1, accuracy and precision, linear is higher than sigmoid.

For recall, in marketing dataset, linear is much higher, but in wine dataset, sigmoid is higher when training size is small.





III. Comparison across models

1. **Time**

   Training time is lowest for kNN, and then decision tree. Neural network and boosting have around the same training time in my choice of learning rate and size of hidden units. SVM is the highest.

   Prediction time is highest for for SVM, then kNN, then neural network and boosting, and lowest is decision tree.

   The timing results make sense because kNN is a lazy algorithm which saves the calculation during prediction so predicting takes much longer than training. Others finish modeling in the training period so it takes them much longer to train than to predict. Boosting is based on decision tree and have multiple decision trees so it will take longer to train and predict. SVM takes the longest time to train and predict because it's the most computationally complex.

2. **Cross validation**

   Due to the **introduction of cross validation**, when training size becomes large enough, the metrics are pretty stable because they have enough data to cross validate.

3. **Best algorithm**

   I define best as the algorithm that runs the fastest, and also have the highest f1 and accuracy scores. Recall and Precision are not the major criterion here because they are included in f1 calculation.

   For marketing data, decision tree is the best algorithm, within a short training and prediction time, it yields the highest f1 and accuracy results. Boosting is a bit less favorable because with similar metrics, the time it spends is larger.

   For wine data, boosting is the best algorithm, although it's having a slightly longer training and prediction time than decision tree, it has better f1 and accuracy results than all other algorithms.

4. **Improving performances**

   Currently I am using GridSearchCV to pick the best parameters for each algorithm.

   - kNN: it differs for different problems, but if we select uniform weights and also lower number of neighbors we can get lower training and prediction time as part of performance.
   - Decision tree: I've selected the best parameters and it turns out that max depth for each problem will be very different and should be tested before nailing down this parameter.
   - Boosting: there is a tradeoff between learning rate and n_estimators, I could use more combinations to figure out the best performance.
   - Neural network: Can introduce a reasonable alpha to avoid overfitting or underfitting. Also learning rate can be adaptive (meaning keeps the rate constant as long as training loss is decreasing but will decrease if not) so we have flexible learning rates when the training is not going well.
   - SVM: can implement a lower C to reduce error