GEORGIA INSTITUTE OF TECHNOLOGY
Title: Machine Learning Homework1 – Supervised Learning

Author(s):
Weifeng Lyu

Instructor:
Prof. Charles Isbell,
Prof. Michael Littman

TABLE OF CONTENTS

1. Introduction

1.1 Problem

This project should implement five learning algorithms. They are:

(a). Decision trees with some form of pruning

The decision tree is one of the most commonly used classification techniques that a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g. whether a coin flip comes up heads or tails), each branch represents the outcome of the test, and each leaf node represents a class label (decision taken after computing all attributes).

(b). Neural networks

The neural networks used in this project is Multi-layer Perceptron (MLP), which is a supervised learning algorithm that such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules, they automatically generate identifying characteristics from the learning material that they process.

(c). k-nearest neighbors

In this project, k-nearest neighbors method is used for classification problem that defined as a non-parametric method used for classification. the output is a class membership. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small).

(d). Support Vector Machines

Support Vector Machines is the machine learning technique that given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. The algorithm separates categories divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

(e). Boosting

Gradient Boosting is the machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The algorithms optimize a cost function over function space by iteratively choosing a function (weak hypothesis) that points in the negative gradient direction

(f). testing

In this project, I implement two sets of data to test above five algorithms, one of them is true or false problem, the other is five-choice classification problem. By implementing five machine learning model to predict the result, the report will show the accuracy, error, learning curve, r2 score and computation time for better demonstrate how the algorithm performs differently in different dataset.

1.2 Computer Configuration

The following configuration has a strong correlation about the computation performance of each algorithm

Manufacturer: Dell, Model: Inspiron 7559 Signature Edition, Processor: Intel® Core™ i7-6700HQ CPU @ 2.60GHz 2.60GHz, Installed Memory (RAM): 8.00GB (7.88 usable), System Type: 64-bit Operating System, x64-based processor

1.3 Workflow Diagram

The workflow is designed for analyzing the accuracy scores and error based on different machine learning methods, these methods are Decision Tree, Neural Network, K-Nearest Neighbors, Support Vector Machine and Boosting. For each of machine learning methods, this experiment implements the algorithms provided by sklearn into two different kinds of classification problems, tunes meta parameters in order to obtain greater accuracy and analyze its performance. This report demonstrates many graphs about analyzing accuracy in comparison of meta parameters. These five machine learning algorithms have many parameters to tune so that the accuracy will be significantly different. To better view the effects but keep the readability of the graph, the implementation chooses two important parameters, they are

a. Decision Tree: (1). max_depth, (2). min_samples_split
b. Neural Network: (1). hidden_layer_sizes, (2). activation
c. K-Nearest Neighbors: (1). n_neighbors, (2). p
d. Support Vector Machine: (1). C, (2). kernels
e. Boosting: (1). n_estimators, (2). max_depth

The graphs will follow the same conventions to generate plots, using the first important parameter as x index, then the second parameter as legend, y axis as accuracy, also distinguish the accuracy between using cross validation or non-cross validation in addition to different line styles. Hopefully the project demonstration gives a good view of how the machine learning algorithm performance.

2. Dataset

2.1. Credit Card

This dataset has a volume of 30000 rows credit card client data, is acquired from UCI machine learning repository, https://archive.ics.uci.edu/ml/datasets/default+of+credit+card+clients. The dataset is divided into two subsets as the prediction result, one is with default payment, value is 1, the other is without payment default, value is 0. The following statements are the reason of choosing this dataset:

1. Credit card dataset has a great volume of data to process the machine learning method without much bias value.
2. All the data are numeric and easy to clean
3. The data generate a relatively good result for accuracy, around 80% and allow a better analysis of the results.

The difficulty is:

4. They have enough distinct attributes (which means many dimensions) to learn in the algorithms. For example, age is slightly different but has great impact on whether they have default payment, bill value for each has significant difference among the data but has not too much influence on whether they have default payment.

Overall, the credit card dataset is a standardized dataset to predict in machine learning method to produce reasonable and non-bias result.

2.2. Seasons Stats

This dataset has a small volume of data, a few above 15000 rows, acquired from Kaggle https://www.kaggle.com/drgilermo/nba-players-stats . The dataset is divided into five different subsets as the prediction results, representing which position the player plays in basketball game based on the stats they play in historical data. In addition, I pre-process this dataset with cleaning some rows and columns with null value, which are a small part of them, also cleaned the final output from C, PF, SF, SG, PG into 1,2,3,4,5, there are some trivial changes such as C/PF, this data will be turned into 1 since the information tells that the player can play basketball in center for the best performance. The following statements are the reason of choosing this dataset:

1. The volume of this dataset is great enough to perform machine learning analysis.
2. This dataset lasts in a long period, from 1988 till 2017, so it's representative and interesting to learn from basketball history.
3. The data is highly correlated, expected to generate meaningful result.

The difficulties are:

3. This dataset is 5-choices classification problem that might cause lower accuracy
4. This dataset requires small pre-processing and normalizations using google open-refine, and this process might negatively affect final prediction. About ¼ of data rows are deducted from the original dataset.

5. This dataset contain various types of attributes, not only numeric, but also float types of data, many of attributes are associated and have a better correlation in final prediction.
Overall this dataset is a standard long-period history data containing many attributes to learn. However, there could be more bias than credit card default payment, which leads to the fact that produce less accurate result.

3. Project Architecture

3.1. Python Implementation

This project implements in Python and sklearn, also with other toolkits such as Numpy and Pandas.

(a). Parameter Tuning

Parameter Tuning aims to find the best parameter combination to obtain higher accuracy prediction. This is implemented for loops in the number of parameters then generate graphs to demonstrate the analysis.

(b). Complexity Analysis

Complexity Analysis aims to find the computational time of each algorithms in different classification problem, different parameters tuning, not including cross validation.

(c). Cross Validation Effectiveness

Cross Validation Effectiveness aims to find whether cross validation can improve the performance or not, why cross validation can improve or reduce the accuracy.

(d). Learning Curve

The Learning curves aim to evaluate the model performance, usually denoted by the risk, cost or score versus the size of training set and test set. The learning curve indicates how much data the machine learning method might need to optimally train the model.

(e). Model Comparison

The model comparison aims to find the best algorithm to predict certain dataset by comparing their error, r2 score and most importantly, accuracy and computational time.

3.2. Project Demonstration and Sample Result

(a). Decision Tree

Decision Tree is one of the most popular machine learning algorithms greatly applied to classification problem, its purpose is to approximate discreted-valued target functions. decision tree learning is generally best suited to problems. Now let's implement into my datasets and see how well it can achieve.

Decision Tree are suitable for the following data analysis:

1. Instances are represented by attribute-value pairs.
2. The targetfunction has discrete output values.
3. Disjunctive descriptions may be required.
4. The training data may contain errors. Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
5. The training data may contain missing attribute values. Decision tree methods can be used even when some training examples have unknown values.

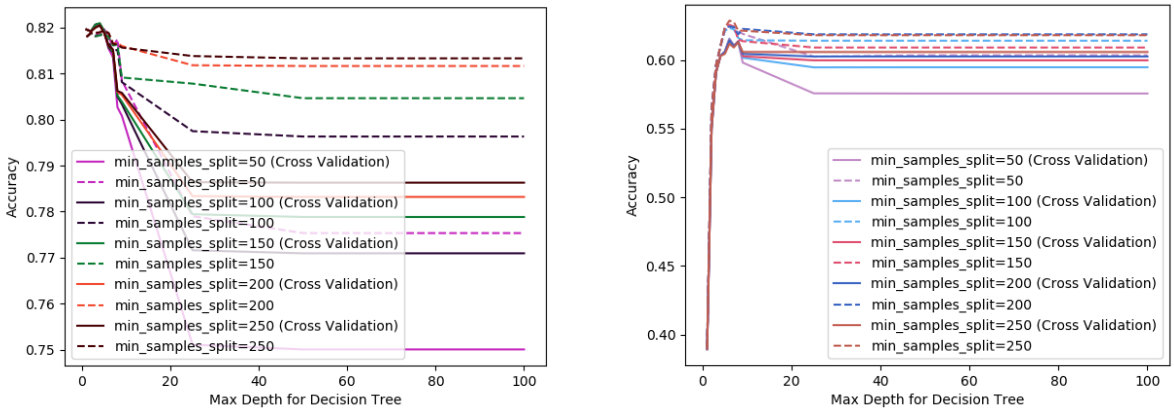


Figure1. Parameter Tuning for Decision Tree (Left) for Credit Card, Parameter Tuning for Decision Tree (Right) for SeasonsStats

Max_Depth controls the maximum depth of the tree that will be created. It can also be described as the length of the longest path from the tree root to a leaf. Min_Samples_Split is the minimum number of values in a node that must exist before a split is attempted.
From what we can see in the graph, when the experiment adjusts max_depth to high value, the accuracy reduces, that's because over-fitting occurs. When the tree is designed so as to perfectly fit all samples in the training data set. Thus it ends up with branches with strict rules of sparse data, it affects the accuracy when predicting samples that are not part of the training set. To overcome overfitting, there are three methods, Prune the tree after building to remove unneeded leaves, or use ensembling to blend the predictions of many trees, also restrict the depth of the tree while you're building it. From parameter tuning perspective and ROC AUC Score (Not plotted), minimum_sample_splits should remain low values to avoid overfitting.

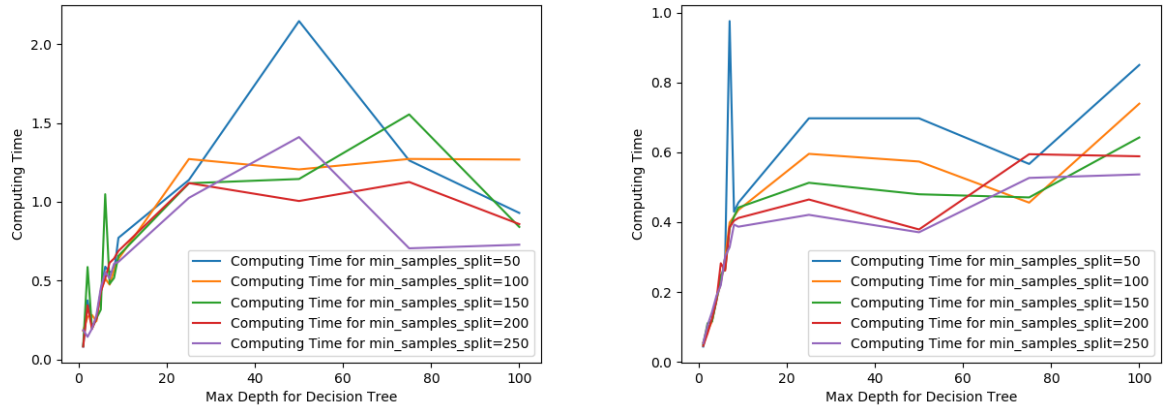


Figure3. Computation Time for Decision Tree (Left) for Credit Card, Computation Time for Decision Tree (Right) for SeasonsStats

From analyzing the computation time, it can be concluded that decision tree algorithm is simple and fast-computed, the lower min_samples_split is, more computation time required, the higher max_depth is also a key factor to cause longer computational time, the running time analysis for Decision Tree is O(Log(N)) in terms of the depth.

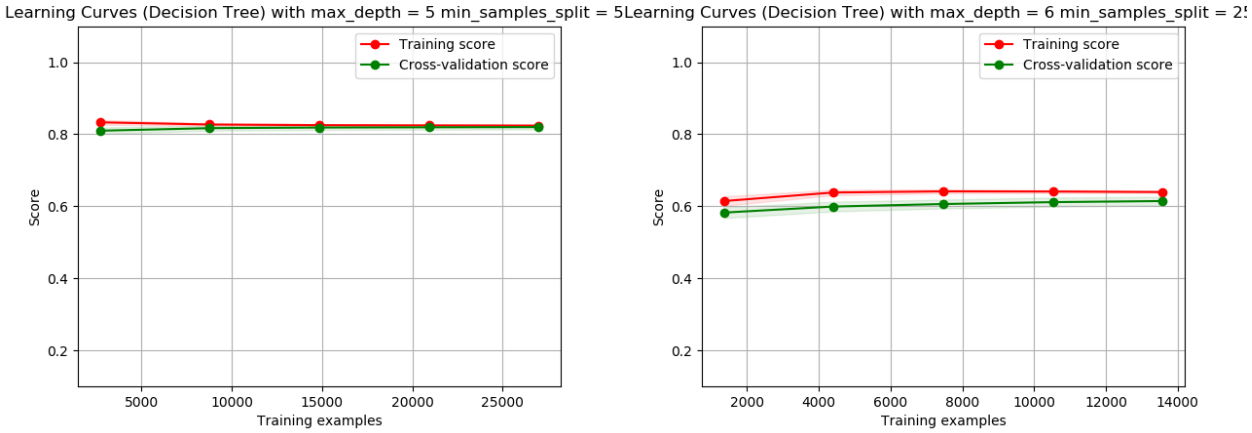


Figure4. Learning Curves for Decision Tree (Left) for Credit Card, Learning Curves for Decision Tree (Right) for Season Stats

From what we can see in learning curve, the score are normally converged into training score in two dataset. However, in Season Stats data, there is a larger gap between training score and cross-validation score. Therefore, when decision tree encounters more complicated dataset, it becomes more difficult to converge accuracy and training score.

(b). Neural Networks

Neural network learning methods provide a robust approach to approximating real-valued, discrete-valued, and vector-valued target functions. The backpropagation algorithm is widely used in neural networks for many predictions and achieve many successes.
Neural Network are suitable for the following data analysis:
1. Instances are represented by attribute-value pairs.
2. The target function output may be discrete-valued, real-valued, or a vector of several real- or discrete-valued attributes
3. The training data may contain errors.
4. Long training times are acceptable.
5. Fast evaluation of the learned target function may be required.
6. The ability of humans to understand the learned target function is not important.

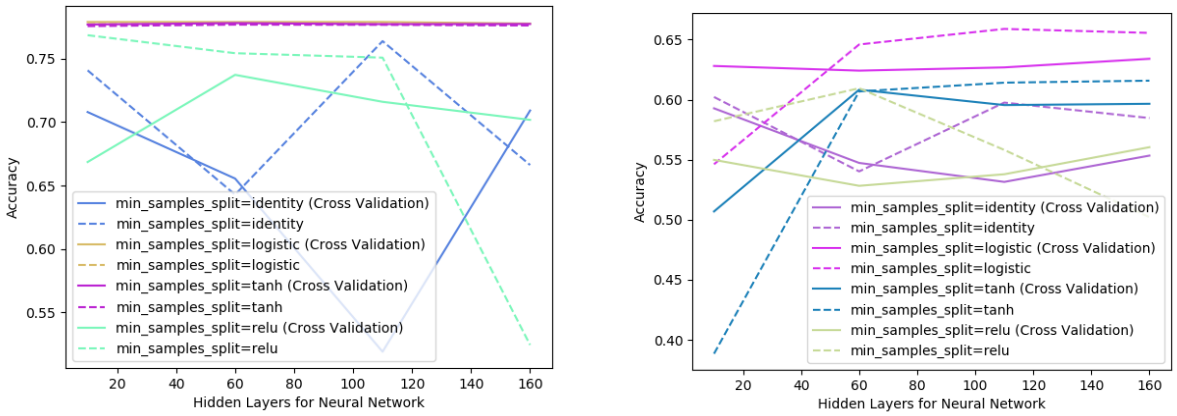


Figure5. Parameter Tuning for Neural Networks (Left) for Credit Card, Parameter Tuning for Neural Networks (Right) for Season Stats

In comparison of other algorithm, Neural Networks perform better in NBA Season Stats, but poorer in Credit Card Default. The reason is that ANN learning methods are quite robust to noise in the training data.
Layers in a neural network is an important factor to set the number of trainable parameters in the network, which represents the number of neurons in the ith hidden layer. From observation, it can be found that there is an optimal number of hidden layers in different activation, for Credit Card, they are around, 60 for identity, and coincidently the optimal hidden layer of identity is also around 60 for Season Stats. By analysis, it seems that more hidden layers do not harm the accuracy, because MLP classifier will converge still, but less hidden layers will prevent it from converging. With the calculation of the error rate, it is more convinced to select the ideal hidden layers with lower error rate. However, the effect of the activation function is the key to improve the accuracy, the logistic model perform the best among other models. If every activation function allows similar accuracy, all of them have their particular use-case which allow to get better accuracy or shorter training time. MLPClassifier trains iteratively since at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting.

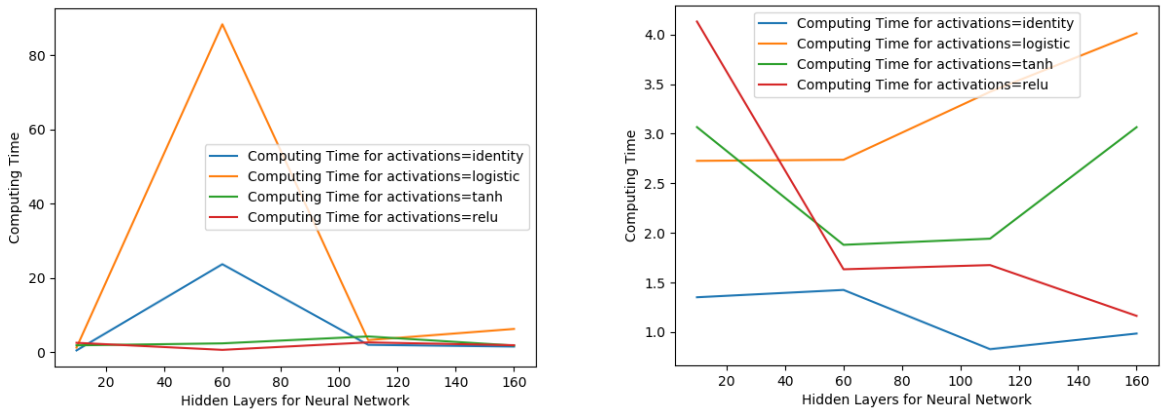


Figure6. Computation Time for Neural Networks (Left) for Credit Card, Computation Time for Neural Networks (Right) for Season Stats

From the above figure, it can be concluded that most of computation time is contributed from activations but not hidden layers value. Compared to other algorithms, MLP classifier is relatively fast by taking advantage of multi-threading.

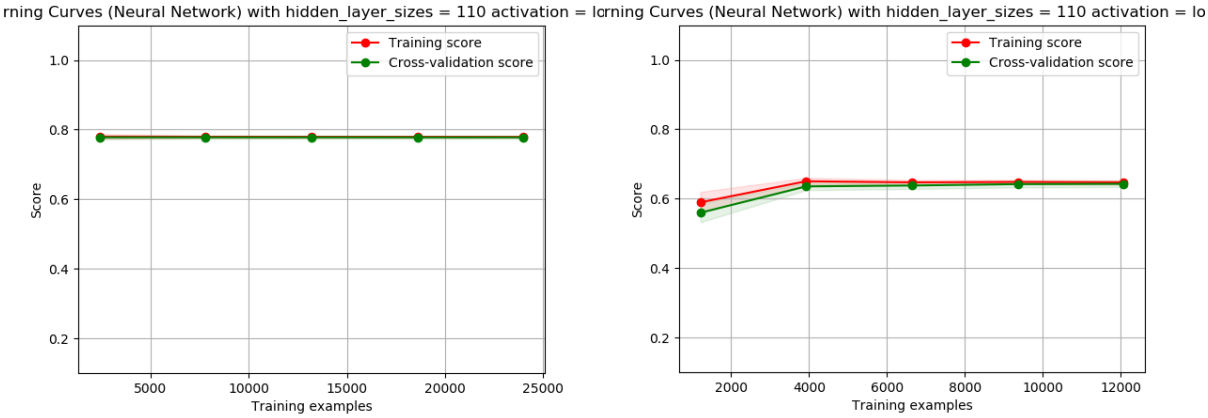


Figure7. Learning Curves for Neural Networks (Left) for Credit Card, Learning Curves for Neural Networks (Right) for Season Stats

One of interesting facts is that more training samples does not significantly affect the accuracy score from the observation above, which may lead to the conclusion that the training data is fast self-learned and not large volume required.

(c). K Nearest Neighbors

K-Nearest Neighbor machine learning method provides an easy-understood and intuitive concept to new learners. KNN can be used for both classification and regression predictive problems. KNN intends to find the nearest objects to learn the similarity of them, so it can predict new object. KNN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The KNN algorithm is among the simplest of all machine learning algorithms, with ease to interpret output, less calculation time, great predictive Power

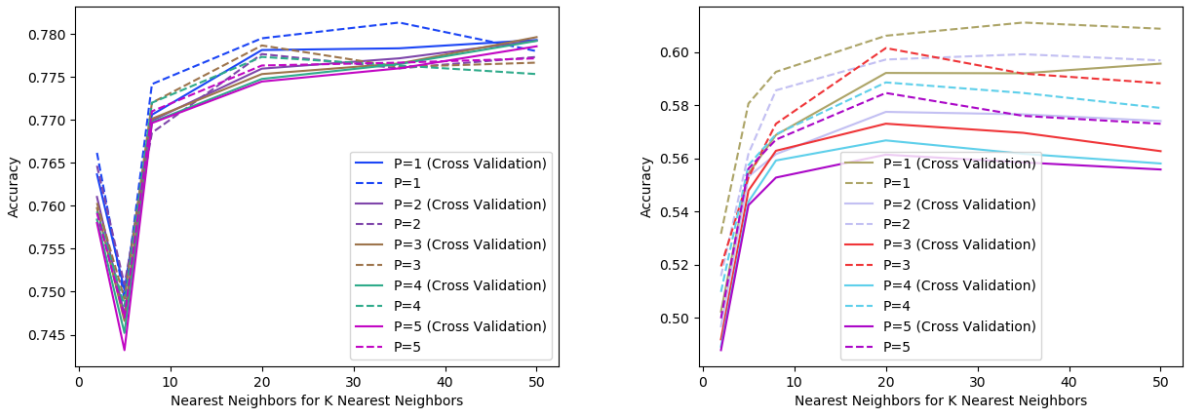


Figure8. Parameter Tuning for K Nearest Neighbors (Left) for Credit Card, Parameter Tuning for K Nearest Neighbors (Right) for Season Stats

K-Nearest Neighbor performs moderately among selected algorithms on both dataset, as we observe the dataset, there could be a good choice to tune P because Power parameter for the Minkowski metric, as different types of calculations for the distance so by understanding different kinds of attributes in the dataset, we can acquire better predictions by increasing P. Evidently, P=1 is definitely the best parameter choice for less computation. In addition, the selected datasets do not have many dimensions, or we can say most of them have more like a direct minus correlation rather than distance, for example, the number of rebounds in basketball game can straightly indicate this player is Center or Guard. Unlike, the prediction on map requires extra P=2, 3, 4, 5, 6 level computation, so P does not play a significant impact. As expected the quality of the classification improve with the addition of new training example and do not seem to overfit.

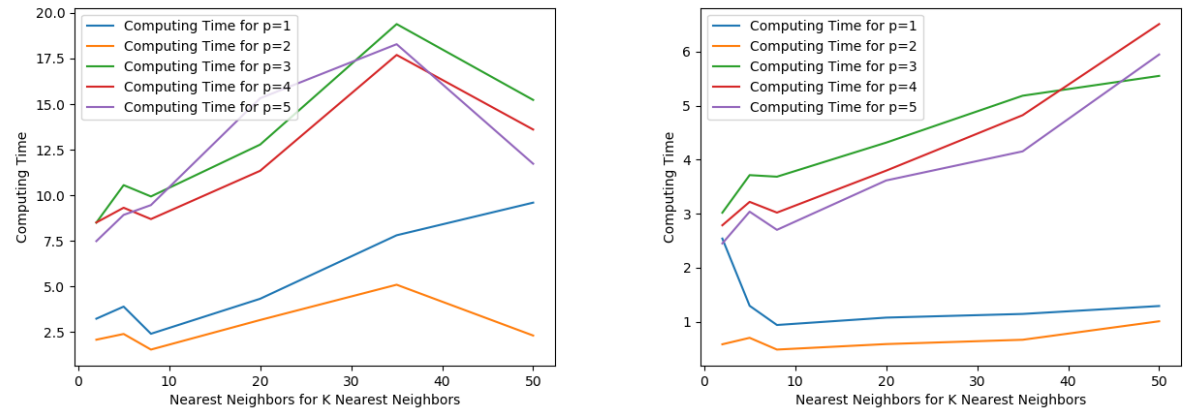


Figure9. Computation Time for K Nearest Neighbors (Left) for Credit Card, Computation Time for K Nearest Neighbors (Right) for Season Stats

Regarding the computational time, KNN also perform moderately among other algorithms. The difference of computation time can be easily discovered and justified. p = 1, this is equivalent to using manhattan_distance (l1), and euclidean_distance (l2) for p = 2. Therefore, more dimensions will involve more expensive multiplications and square root. For the value of n_neighbors, it can be illustrated from KNN model that it takes O(N) running time for unsorted objects and O(logN) for sorted items.

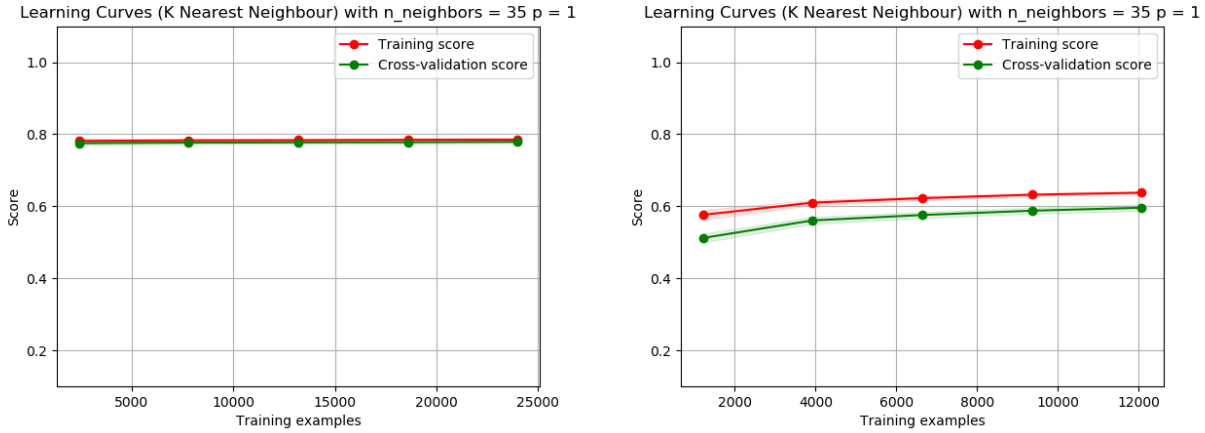


Figure10. Learning Curve for K Nearest Neighbors (Left) for Credit Card, Learning Curve for K Nearest Neighbors (Right) for Season Stats

Regarding KNN algorithm learning, I find that as training samples increase, KNN does converge with training data in Credit Card case but not SeasonsStats case. Therefore, it probably means that SeasonsStats have more classification choices, significantly reduce the nearest neighbor attributes, also more complicated data and various kinds of attributes make KNN prediction difficult to obtain a good result.

(d). Support Vector Machine

Support Vector Machines are powerful tools, but their compute and storage requirements increase rapidly with the number of training vectors. Support Vector machine is highly effective in many dimensional spaces, also where number of dimensions is greater than the number of samples. It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient. Different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels. However, if the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial. It does not directly provide probability estimates.

Since its computation time is related to sample size and features exponentially, it is time consuming to compute the prediction by training all the values, I determine to randomly drop rows down to 5000 rows to compute.

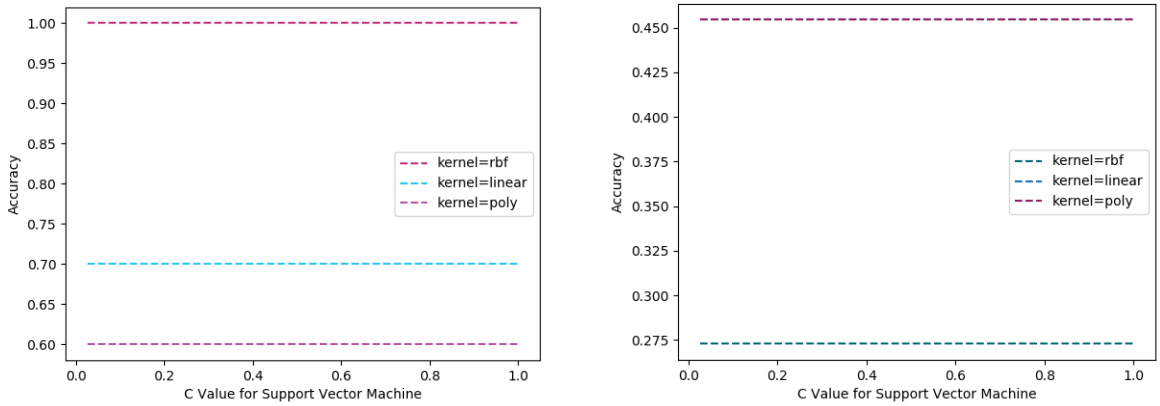


Figure11. Parameter Tuning for Support Vector Machine (Left) for Credit Card, Parameter Tuning for Support Vector Machine (Right) for Season Stats

This parameter tuning basically gives overview about how kernel mainly affect the accuracy, rbf kernel seems not provide valid result. Also, C, as Penalty parameter C of the error term does not play any effect on prediction accuracy in these two datasets because the training model is heavily based on the complicated dataset, it will not perform any penalty for those values. Also, we can notice that accuracy for SeasonsStats is a way lower than Credit Card, something like rbf, more like a guessing algorithm, gets only 1/5.

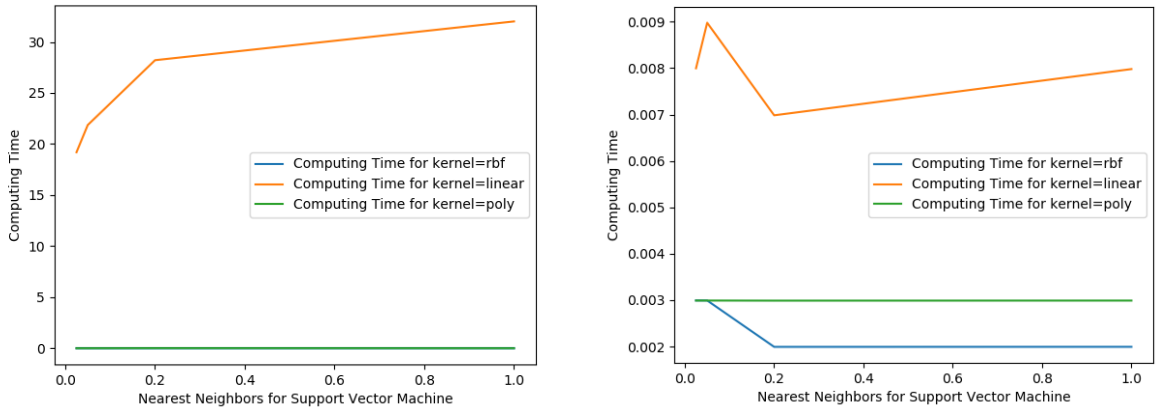


Figure12. Computation Time for Support Vector Machine (Left) for Credit Card, Parameter Tuning for Support Vector Machine (Right) for Season Stats

Support Vector Machine's computational time is based on training sample size, The QP solver used by this libsvm-based implementation running time analysis scales between O(n_features*n_samples^2) and O(n_features*n_samples^3) depending on how efficiently the libsvm cache is used in practice. In order to perform a valid result, the training samples are randomly deducted, from the graph we can see computational time is positively proportional to C in linear, reflecting 'linear', others remain same as C increased.

(e). Gradient Boosting

Gradient Boosting is a machine learning ensemble meta-algorithm for primarily reducing bias, and also variance in supervised learning, and a family of machine learning algorithms that convert weak learners to strong ones. GBM is a highly popular prediction model among data scientists, it allows for the optimization of arbitrary differentiable loss functions. The algorithm builds upon other algorithms (weak learners) to optimize the prediction. Boosting is very similar to classical decision tree, more like an enhanced version of decision tree.

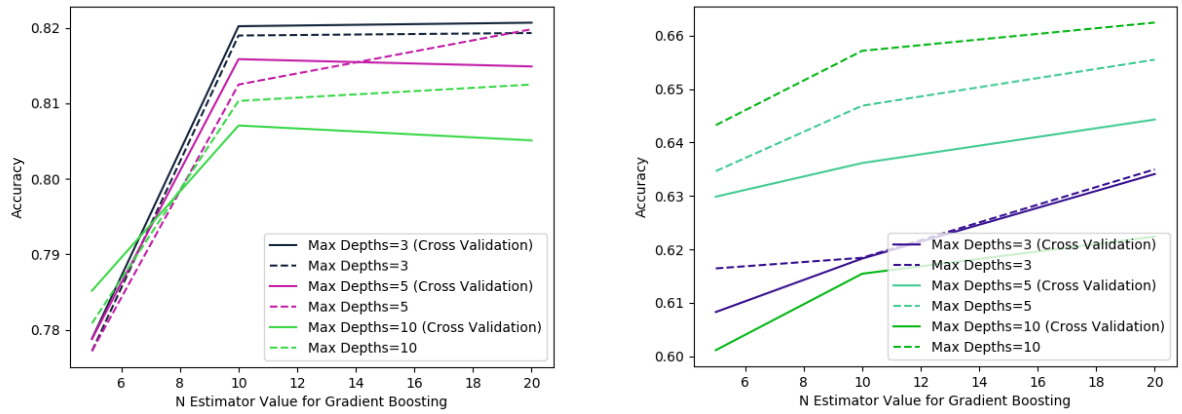


Figure14. Parameter Tuning for Gradient Boosting (Left) for Credit Card, Parameter Tuning for Gradient Boosting (Right) for Season Stats

n_estimators and max_depth are two factors as parameters that we explore. n_estimators is the number of boosting stages to perform. Gradient boosting is fairly robust to overfitting so a large number usually results in better performance. max_depth is maximum depth of the individual regression estimators. The maximum depth limits the number of nodes in the tree. This parameter can be tuned for achieve the best value depends on the interaction of the input variables. It can be concluded that more weak estimators can contribute into better accuracy performance. However, there could be some overfitting after n_estimator reaches above 10, but for SeasonsStats data has better tolerance for n_estimator because its features need more weak learners to compute, aggregate into a good model.

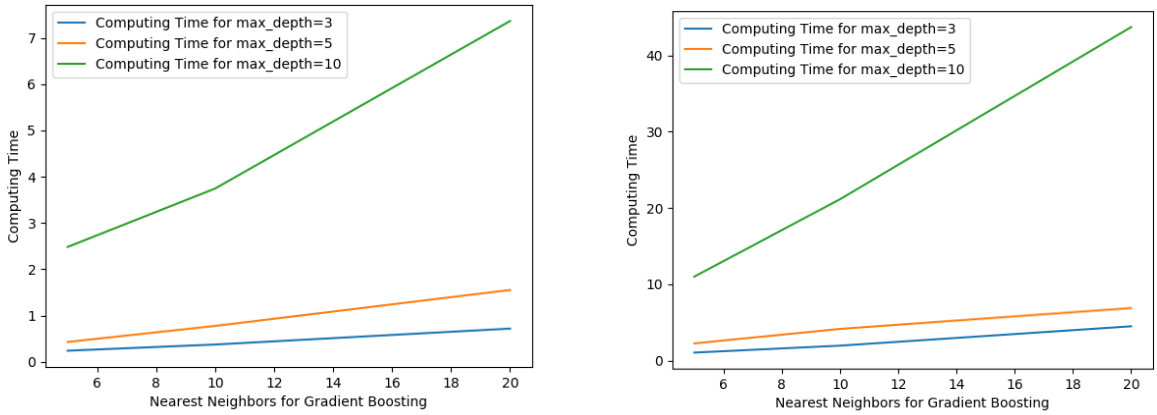
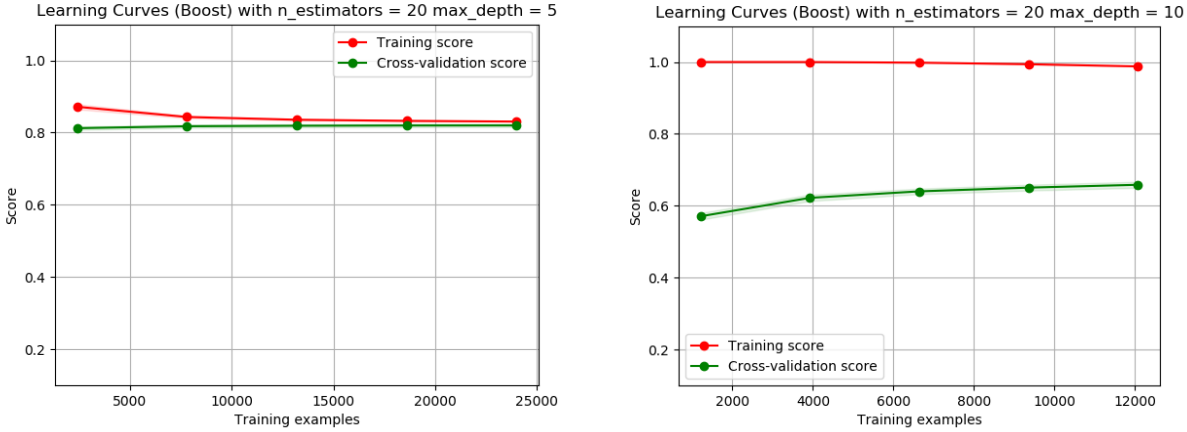


Figure15. Computation Time for Gradient Boosting (Left) for Credit Card, Parameter Tuning for Gradient Boosting (Right) for Season Stats

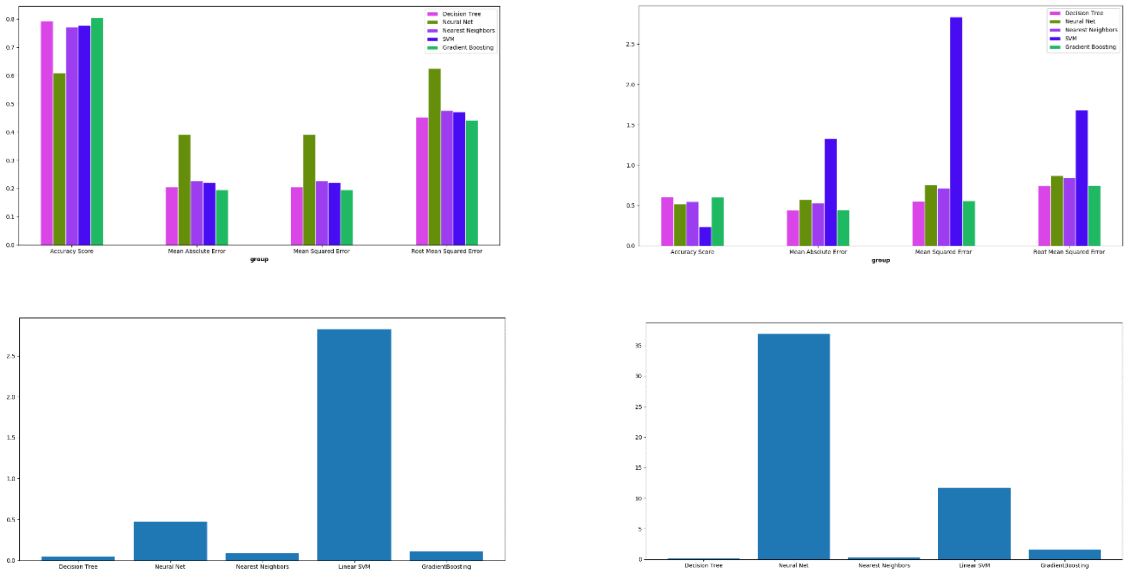
The computation time for Gradient Boosting is very long, especially when max_depth is increased, SeasonsStats also need more computational time than Credit Card, which can be explained that there are more features to learn and be established weak learners based on those features.



(f). Cross Validation

Regarding cross validation, it does not play any positive improvement on accuracy, it generates more computational legacy. Most of predictions without cross validation perform better since in the train_test_split, using 80% of data for training (test_size=0.2) only takes a single time, in cross_val_score the data will be splitted into 90% train (cv = 10) 10 times (Each time 9 folds will become the train and remaining 1 as test), so 90% training data is more than 80% so the accuracy should not decrease. The train_test_split will not stratify the splits by default, but it will do in cross_val_score. Stratification is keeping the ratio of classes (targets) same in each fold. Most probably, its happening that the ratio of targets is not maintained in the train_test_split which leads to over-fitting of the classifier and hence this high score.

(g). Model Comparison



From all performance analysis above, I select the best parameters from all experiment above, then run performance test, so we can draw a conclusion that in terms of accuracy, Gradient Boosting algorithm ranks no.1 for the accuracy. Decision Tree is the most effective algorithm, SVM is the most time-consuming algorithm because its running time is based on training sample, after the training sample is deducted, for similar features SVM still requires more computation effort, but for various feature dataset, neural network takes more time to learn and predict. Since we have deducted training size, the R2 score is computed but not plotted here, they are mostly negative.

5. Conclusions, Discussion and Improvement

In summary, this project implements five classical machine learning techniques, analyzes their meta parameters contribute a major part for better performance, basically it gives an overview for me to learn how machine learning works. Tuning the parameters takes much effort and requires many tests. It also relies on solid understandings of dataset and mathematical background of the algorithms.

Data is really the heart of machine learning, making a computer generalize from a very few amounts of data is a difficult task. Many data-cleaning and preprocessing are required to be done before putting those data into algorithms. Implementing best-fit algorithms into different datasets is a tough machine learning technique, also as known, associated with many crossed study fields.

To obtain the best accuracy but not overfitting, there are many matrixes result that can be optimized. All the machine learning methods are not equal on the land of data, but most of them have similarity in terms of machine learning perspective. The biggest challenge is to enhance the computational complexity of their execution. some results presented here have been computed in hours, if not in days so that there has to be some data cutting in place. Thus, as always machine learning is a compromise between data, time and computational power. Of courses, many algorithms have its strength and shortcomings. This homework starts leading me to explore a new mathematics and statistical world of predicting with machine learning techniques using real data, and better understand how they work, their advantages and disadvantages, resolve real-world problems with optimal solutions more effectively.