Machine Learning Supervised Learning Analysis

This paper involves the analysis of applying machine learning algorithms to two data sets in an attempt to classify them with a high degree of accuracy. The first data set is a set of personal information which we use to classify a person as having a term deposit, or not having a term deposit . The second set takes measurements of tumors on breasts and classifies them as benign or malignant. Both datasets were picked because they work really well with certain algorithms over others. The analysis will involve comparing algorithms, analysis of how each algorithm performs, and assertions about the data as to why certain algorithms work better. I will attempt to provide the best performance from each algorithm on the dataset to give the most fair comparison

Banking Deposit Classification

The first problem I picked was to take information about bank customers and a marketing campaign and classify them as either getting or not getting a term deposit.. I picked this problem for a number of reasons. One reason is that I recently quit a banking job as an appraisals statistician. Most of my job was to apply regression, over and over again, to different markets to see how they are fairing.I became tired of regression. I wanted to find a dataset which didn't have much linearity to it, that was related to banking. This data seemed very categorical, which meant that it would be hard to run predictions based on regression. I pulled the data from Kaggle (https://www.kaggle.com/janiobachmann/bank-marketing-dataset) because they had it already in a clean, workable format. But they pulled the data from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Bank+Marketing).

Column Descriptions

Here are the columns of the data
1 - age (numeric)
2 - job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')
3 - marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)
4 - education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')
5 - default: has credit in default? (categorical: 'no','yes','unknown')
6 - balance: the balance in their accounts
7 - housing: has housing loan? (categorical: 'no','yes','unknown')
8 - loan: has personal loan? (categorical: 'no','yes','unknown')
related with the last contact of the current campaign:
9 - contact: contact communication type (categorical: 'cellular','telephone')
10 - day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')
1 - month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

12 - duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y='no'). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
other attributes:
13 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
14 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
15 - previous: number of contacts performed before this campaign and for this client (numeric)
16 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
17 - deposit: whether or not they got a term deposit as a result of the marketing campaign

More about the Data

The data was pretty evenly distributed with about 47% of the entries being successful deposits. None of the data is particularly strongly correlated with the term deposit. Balance seems to be the highest correlated. The first thing I did to the data was transform the categorical data into numerical data. Then I normalized all the numerical fields by dividing them by the max value in that field.

Model Analysis

To determine the best algorithm, I tested SVM with rbf and linear kernels, Adaboost, XGBoosted, and Randomforest decision trees, Neural Networks, K Nearest Neighbors, and Logistic Regression. I tested by gathering AUC(ROC) and Accuracy for tests of the data Cross Validated with both K-Folds and Randomization. I also did a basic test and split and tested AUC, Accuracy, precision, recall, and f1. Since the data was fairly balanced with its outcomes (47% vs 53%), AUC should match accuracy fairly well. But I still ran AUC alongside accuracy to account for the minor difference. Maximizing recall and precision should help the bank avoid spending marketing money on a false positive and help the bank avoid missing a potential customer marked as a false negative. Now I will go over each algorithm, explain how I optimized it, and discuss its scoring.

SVM

SVM was tested using Sci-Kit Learn's SVC algorithm. Both 'linear' and 'rbf' kernels were tested.

Linear

The linear kernel scored demonstrably worse on every attempt than 'rbf'. SVC's C parameter is inversely proportional to it's regularization strength. So if we get a test better with higher C, we can be confident that our data is somewhat linearly separable. We actually got lower on most metrics when raising C. So we know that our data is not linearly seperable. With our cross

validation SVC Linear typically scored 78%-80% on auc and accuracy. Testing usually took about 10-20 seconds

RBF

RBF scored pretty well. Also, it scored better as C increased until C hit about 7 or 8. This suggests that the data is geometrically separable in at least a small capacity. With cross validation, SVC(rbf) typically scored 80%-82% on auc and accuracy. Testing usually took about 15-25 seconds.

Decision Trees★

For Decision Trees, I tested three boosting algorithms, XGBoost, AdaBoost, and RandomForest. These Boosted Decision trees tested the best out of all of the types of Algorithms I tested. Oddly, RandomForest and AdaBoost tested better with a higher maximum depth and higher number of maximum estimators.This is worth mentioning because it's around 4% of difference between a maximum depth of 5 and 15 even accounting for Cross Validation. XGBoost seems to cap out performance when pruned down to about 7 or 8. Also, I got the best performance changing the gamma to .51 to make the tree more conservative. I also changed the learning rate to .05 and the number of estimators to 50, and this really worked well. XGBoost tested the best overall out of all algorithms tested. With CV, it usually tested at about 84-86% auc and accuracy. Also, the testing ran the quickest out of all tests performed at around 2.5-3 seconds. The good performance of the data on decision trees is what I expected. Our data is highly categorical and nonlinear so the good performance on decision trees was no surprise. The other Algorithm I expected decent performance on was Neural Networks

Neural Networks

Testing Neural Networks was kind of rough. I had a tough time getting it to converge. I had to increase the maximum iterations to 1000 just to get it to converge. Beyond this, I had a tough time making adjustments which increased the metrics demonstrably. I changed the learning rate and only lost accuracy when increasing the rate. I changed the solver to 'lbfgs' and stochastic gradient descent. I, once again, only lost accuracy. Also the runtime increased dramatically. I changed the alpha (regularization parameter), no significant differences. I couldn't get the model to show much better results beyond its default settings. I did at least get decent results from it. I got about 80-83% Auc and Accuracy. Also, it took a very long time to test with each test running about 2 to 3 minutes.

K-Nearest-Neighbors

For K-Nearest Neighbors, I tested the data set with differing K values. I used Sci-Kit Learn's KNeighborsClassifier model. I saw the best accuracy and auc when looking at the 3 nearest neighbors. This model was one of the worst performers. At best it typically obtained about 70-73% AUC and accuracy. This was actually somewhat surprising, given that the rbf kernel SVM model performed somewhat well. But it is potentially due to the high amount of categorical data. KNN ran the fastest at about .5 seconds.

Logistic Regression

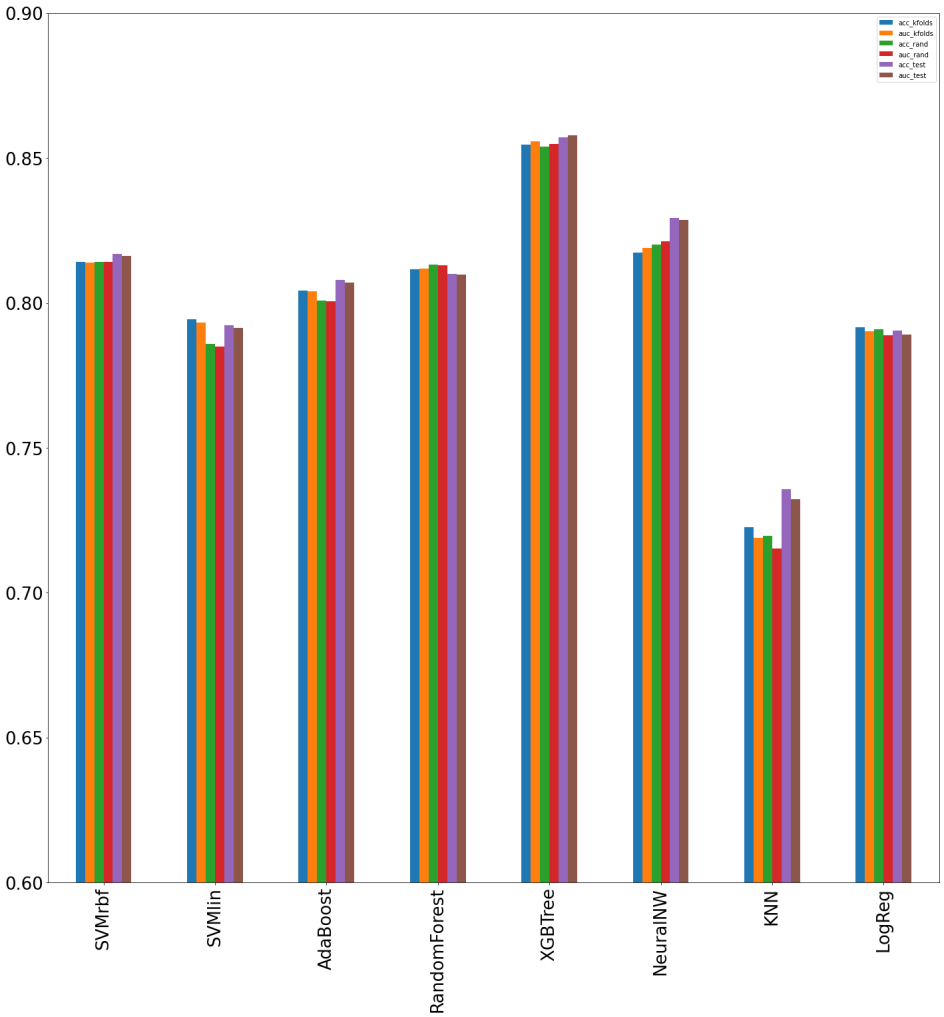
Finally, I tried the LogisticRegression Model. It scored about 78-80%. This is higher than I expected. Nothing I did really changed the accuracy or auc that much. The data doesn't have much linearity to it and is mostly categorical so I'm genuinely surprised that it scored this high. Logistic Regression also ran very fast.

Final Analysis of Models

Decision Trees performed the best on average. The XGBoosted Decision Tree performed the best. In cross validation testing it scored 84-86% on auc and accuracy. This is likely due to how non-linear and categorical the data is. Because the data was very even, AUC, Accuracy, and all of the other metrics tended to be very similar, which is why I really didn't discuss metrics like f1 and recall.

Graphical Summary

The legend is small so I'll put it here. Blue = accuracy of KFolds CV, Orange = AUC of KFolds CV, Green= accuracy of Randomized CV, Red = AUC of Randomized CV, Purple = Accuracy of Test/Train Split, Brown = AUC of Test/Train split



Breast Mass Classification

The Second problem I picked was to take the measurements of a cell nuclei taken from a breast mass and use those measurements to classify the breast mass as either benign or malignant. I picked this data set for a few reasons. The first reason is that it contrasts the last data set I picked. This data set is highly numerical as opposed to categorical. I expected that this data set would perform much better under regression, SVM, or KNN. The second reason I picked this data set is because I remember using it in the book “An Introduction to Statistical Learning: With Applications in R” by Gareth M. James. In that book, the data set was used with KNN. I eventually stopped reading that book because I knew that I was too lazy to learn machine learning without fear of a bad grade. But after reading more about SVM with an rbf kernel, I wanted to try that on this data set. This data set comes from the UCI Machine Learning Repository (https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29). Though I pulled the data from Kaggle because it was already ready to go (https://www.kaggle.com/uciml/breast-cancer-wisconsin-data). I mainly just normalised the data by dividing all of the entries by the max for each column.

Column Descriptions

The attribute information is as follows:
Attribute Information:

1) ID number
2) Diagnosis (M = malignant, B = benign)
3-32)

Ten real-valued features are computed for each cell nucleus:

a) radius (mean of distances from center to points on the perimeter)
b) texture (standard deviation of gray-scale values)
c) perimeter
d) area
e) smoothness (local variation in radius lengths)
f) compactness (perimeter^2 / area - 1.0)
g) concavity (severity of concave portions of the contour)
h) concave points (number of concave portions of the contour)
i) symmetry
j) fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image,

resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

Model Analysis

I picked most of the same models and methods as with the problem above. Because the data has only 37% of its diagnoses as positive, AUC(ROC) becomes much more important in this analysis. Also, since we are talking about cancer diagnoses, a high recall is very important. We want as few false negatives as possible. So recall, auc, and Accuracy will probably be discussed the most. I have still used cross validation to ensure that overfitting is accounted for. This is especially important with this analysis as opposed to the last, because some algorithms seem like they perform really well until you account for overfitting.

SVM★

SVMs tested the absolute best out of all of the models I built. The rbf performs better, but this could be a result of how I normalized the data.

Linear

For the linear model, I really only changed C. Changing the maximum iterations didn't result in improved performance. Neither did using probability estimates. This was true for rbf as well. The Linear Model hit its optimum at about C=10-20. The linear model hit 96-97% in KFold and Randomized CV. It hit 95% accuracy and 93% recall in the test-train test. The test suite ran in about .06 seconds.

RBF

SVM with an rbf kernel tested the best all around of any model I tested. As Above, the only hyperparameter which really made a large difference was C. It was optimal at about 5-10. It's AUC and Accuracy were about 97% in Cross Validation testing! This means that the model is very likely not overfitted. Also, the model hit about 98% AUC and accuracy in the test-train split with about 97% recall. This is better than every other model by a fair margin, though other models performed very well. This tells us that the data, while not being perfectly linearly separable, it is at least very geometrically separable, with hyperplanes being able to divide the data by classification very well. Also, it ran in less than a tenth of a second.

Decision Trees

For decision trees, I tested all three boosted models. AdaBoost, RandomForest, and XGBoost. For AdaBoost, I attempted to change the learning rate and didn't get significant change. For both AdaBoost and RandomForest. They both benefited from pruning and hit their best scores in AUC, accuracy, and recall at about a depth of 5. AdaBoost hit accuracy and auc of 92-93% on CV. On the test-train test it hit accuracy and auc of about 95% and a recall of about 94%. XGBoost did very well. In fact, I ran this same set of models on about 6 data sets and this algorithm probably performed the best on average overall. For this Booster, I changed gamma to around .51 again. This time, a learning rate of around .01 worked really well. Also, it converged fairly quickly at 25 estimators giving the best performance. This model performed exceptionally well with the test-train testing. It tested as well as SVC, giving an accuracy, auc,

and recall of about 98%. However, with CV, it still gets blown out of the water. It only tested about 93%-94% with cross validation. This suggests that it gets so high in test-train because it overfits the data. The tree models probably fail by attempting to fit “if-then” patterns where they don’t really belong. If the data is highly geometrically separable, this very well could cause trees to overfit easily. Each tree took about a second to run on the testing suite. Except XGB, it ran in about .2 seconds.

Neural Network

With the ‘adam’ solver, the model had a tough time converging. I switched to the ‘lbfgs’ solver and got much quicker convergence along with much higher scores. I also changed the learning rate to ‘adaptive’ and got a little closer. Also, increasing alpha, made my CV scores increase, but the model has a tougher time converging. I changed the code to converge by removing alpha so that when the grader runs it, it isn’t an absolute mess in the console. The neural network running ‘lbfgs’ works pretty well. It gets about 95-96% in auc and accuracy in CV testing and 97-98% in test-train for auc, accuracy, and recall. This means that it is slightly overfitted, but not as much as xgboost. It is worse than SVM rbf. The neural network took about 1.6 seconds to run all of the tests.

KNieghbors

KNN runs pretty good. It doesn’t run as well as the neural network, xgboost or SVM(rbf), but it runs well. It runs about 96-97% consistently through CV and test-train-split testing. Also, it runs very fast. It hit it’s highest accuracy at N = 5. This was expected, part of the reason I picked this data set is because I knew that KNN would perform well and I wanted to see if SVM with an rbf kernel, which seems like a suped-up KNN in functionality, would perform. This test suite ran pretty quickly at about a tenth of a second.

Logistic Regression

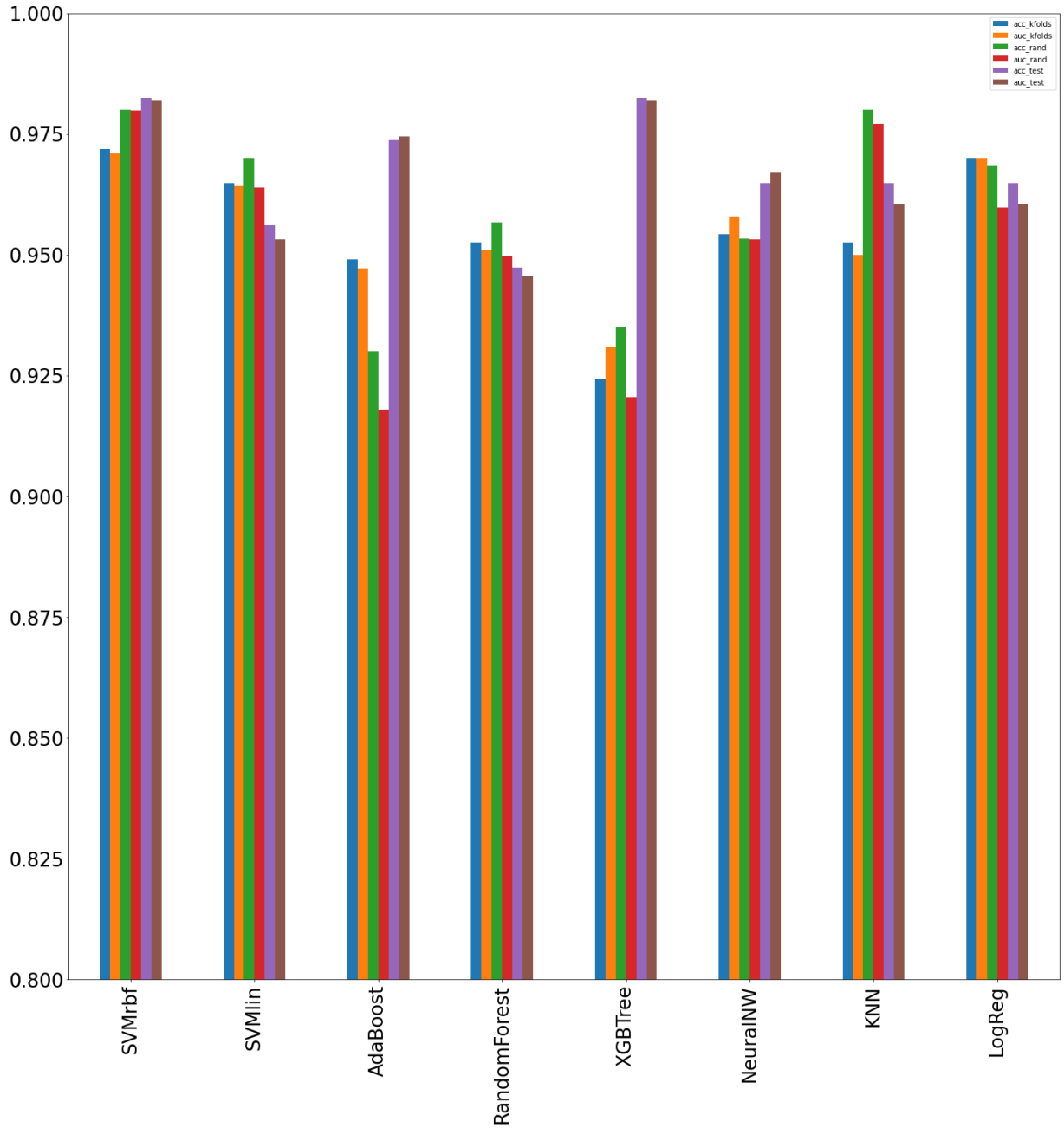
Logistic Regression performed decently well scoring 93-94% in CV and in test-train-split. But then, I decided to try a large C of 40, and its accuracy shot up to about 95-97%. This suggests that the data is at least decently linear. This was kind of surprising. Also, for how accurate it is, it also runs quickly. The entire test suite took about .1 seconds to run.

Final Analysis of Models

As expected, models which deal with a lot of continuous input types did very well. SVM(rbf) did the best, suggesting that the data is well separated by class when graphed. XGBoost did surprising well. Logistic Regression doing well, suggests that there is also a linear relationship between the variables and the classification.

Graphical Summary

The legend is small so I’ll put it here. Blue = accuracy of KFolds CV, Orange = AUC of KFolds CV, Green= accuracy of Randomized CV, Red = AUC of Randomized CV, Purple = Accuracy of Test/Train Split, Brown = AUC of Test/Train split



Works Cited

I haven't written a Works Cited page in a while, so please let me know if I've got anything wrong!

To build the models for SVM, KNN, AdaBoost Decision Trees, Random Forest Decision Trees, Logistic Regression, I used Sci-Kit Learn. Their site asks for a link to the following paper

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.
https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html

To Build the XGBoost model, I used XGBOOST, here is the site.
https://xgboost.readthedocs.io/en/latest/get_started.html
They don't mention anything on how they would like to be cited

Both Datasets where provided by the much esteemed UCI Machine Learning Repository

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

I think that's it