CS 7641 Machine Learning
Assignment 1

Philip Bale
pbale3

Due February 4th, 2018 11:59pm

2 most important plots: learning curve and model complexity curve

Classification Problems

1) US Permanent Visa Applications

Overview

The first classification problem revolves around classifying whether or not a person's US permanent visa application will be accepted or denied based on the parameters of their application. Among the features used in the classifcation are:

- Job features: Industry code, job class, wage rate, wage type
- Geographic features: Country of citizen and employer location

The classes observed for this dataset are simply 'approved' and 'denied'. The dataset contains 374365 total samples.

Why is the dataset interesting?

This dataset is interesting due to its potential to aid in the visa application process from a cost and time savings potential. It could also enable confidence in those interested in applying for a US permanent visa but doubting their chances of acceptance. At the end of the day, the goal is it to try to determine the application result before time, money, and other resources are spent. As someone who has worked with a large number of first-generation visa holders and immigrants, I am extremely interested in building tools to help others to achieve the same.

From a machine learning perspective, the dataset is incredibly interesting due to its wide variety of features and the variety of values those features can take. An immense number of job types, wage rates, and citizenships alone create an extremely diverse dataset. Additionally, the number of samples available provide a comprehensive picture of historical data, lending towards greater cofidence in training and testing rates.

2) Home Sale Price Predictions

Overview

The second classification problem revolves around classifying a home's price bracket based upon the various characteristics of the home. Among the features used in the classification are :

- Subjective measurements: Exterior condition, house style, overall quality rating, and overall condition
- Objective measurements: Type of dwelling, building type, lot size, neighborhood, year built, and year sold

After an initial review of the dataset, the classes were defined as pricing brackets divided into 100k groups. I.e: 0-100k, 100k-200k, 200k-300k, etc. The dataset contains 1451 samples. An additional dataset containing another 1400 testing samples exists but was not used as it contains unclassified sale prices. It will, however, prove useful for unsupervised learning.

Why is the dataset interesting?

This dataset is interesting for two primary reasons: real-world applicability and participating in a Kaggle challenge. First, modeling home prices is both a difficult and lucrative task. If one can succesfully model home sale prices on large sets of data, he/she can make large amounts of money investing in real estate when he/she detects outliers in listed price vs. what it is expected to sell for. This applies to flipping, investing, and remodeling. Second, the dataset is part of an ongoing Kaggle competition that does not have a winning solution yet. By taking part of the competition, the dataset presents the opportunity to work towards a winning solution and advance ones algorithms over time.

Houses can have a very large amount of features–with a large amount of variety in the individual features. Similarly, housing is prone to personal taste and frequent need for upgrades/modernization. In such, I believe price estimation is an excellent problem, full of depth and complexity, that is suitable for a machine learning approach.

General Data Processing

The datasets I used were both relatively clean to begin with. One small problem, however, was that a lot of my features on both datasets were text-based. To transform the features into numeric values suitable for the machine learning algorithms, I used a label encoder built into ScikitLearn.

I also did a small amount of preprocessing of the data to make it more suitable for classification. I dropped all unnecessary columns to help speed up with data processing in general–which proved immensely helpful when dealing with the more computationall intensive algorithms. In the case of home prices, I precalculated the brackets based on the 'sale price' data label. In the case of visa applications, I pregrouped the case outcome so that results such as 'certified' and 'certified withdrawn' are both concerned as 'approved' conditions whereas 'denied', 'invalidated', and 'rejected' all resolve to 'denied'.

Decision Trees

A decision tree classifier was the first algorithm applied to the datsets. Various values of max_depth were tested as a means of pruning unnecessary leaves. Similarly, a grid search was used to test whether a 'gini' or 'entropy' criterion was more effective.

US Permanent Visa Data

Depth	Criterion	Tree Size	Train %	Train Time	Test %	Test Time
1	gini	3	0.7657	0.1212	0.7680	0.0010
3	gini	15	0.6775	0.1417	0.6749	0.0009
6	entropy	105	0.7121	0.2605	0.6924	0.0011
10	gini	889	0.7584	0.3192	0.7072	0.0011
15	gini	3013	0.8628	0.4199	0.7582	0.0014
20	gini	4751	0.9299	0.4517	0.7917	0.0010
25	gini	5349	0.9545	0.5284	0.8032	0.0010
35	entropy	5349	0.9574	0.5116	0.8081	0.0010

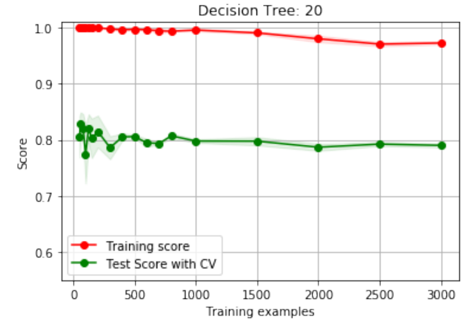
Results at multiple depths for best criterion via grid search

	Accepted	Denied
Accepted	96	135
Denied	217	1376

Test Data Confusion matrix

Learning Curve for max_depth = 3

Learning Curve for max_depth = 10



Learning Curve for max_depth = 20

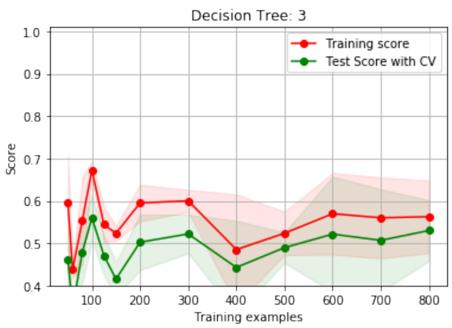
Housing Prices Data

Depth	Criterion	TreeSize	Train %	Train Time	Test %	Test Time
1	gini	3	0.0912	0.0355	0.0890	0.0006
3	entropy	15	0.5816	0.0401	0.5890	0.0003
6	entropy	91	0.6820	0.0442	0.6438	0.0003
10	gini	331	0.8575	0.0566	0.6918	0.0006
15	gini	593	0.9854	0.0695	0.7603	0.0003
20	gini	643	1.0000	0.0559	0.7603	0.0007
25	gini	639	1.0000	0.0550	0.7603	0.0006
35	gini	639	1.0000	0.0570	0.7603	0.0006

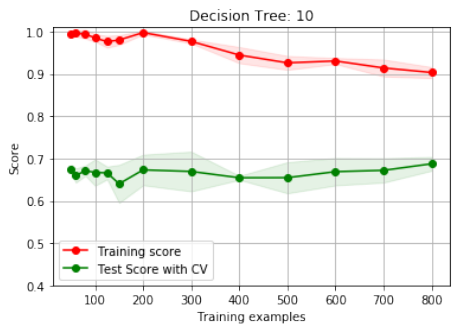
Results at multiple depths for best criterion via grid search

	0-1	1-2	2-3	3-4	4-5
0-1	2	6	0	0	0
1-2	4	87	7	0	0
2-3	0	8	15	1	2
3-4	0	0	6	3	0
4-5	0	0	2	0	3

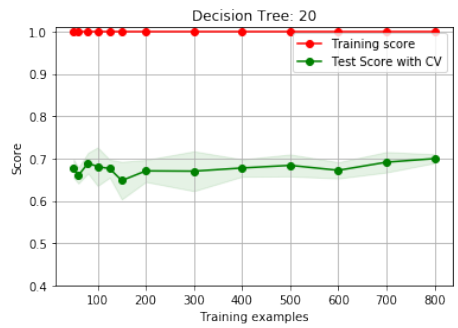
Test Data Confusion matrix (classes in 100ks)



Learning Curve for max_depth = 3



Learning Curve for max_depth = 10



Learning Curve for max_depth = 20

Analysis for Decision Tree

Overall the classifier worked quite well for both datasets, but was extremely prone to overfitting. Examining the results of the decision tree classifier on the two datasets provides numerous observations and basis for analysis, which are provided below.

Effects of dataset size cross validation:

Above, learning curves are provided for both datasets. It is immediately apparent that the size of the dataset greatly affects the performance of the algorithm–though diminishes over time. This makes sense for a few different reasons. The more data we have to train on, the more likely it is that we see the full spectrum of possible variability. Similarly, the broader the set of examples, the less biased our algorithm will be. This is because if we only train on a few data samples, then our algorithm can only make decisions based on the features learned from the small, simple sample size–thus generating a bias (and therefore underfitting).

The learning curves for both datasets clearly level out as dataset size increases, demonstrating less variance with a more informed model. Similarly, cross-validation was used to normalize the training samples and smooth the learnign curve. Without cross-validationg, the model was prone to unrepresentative, biased dataset samples during training and, in-turn, testing.

Tuning Parameters:

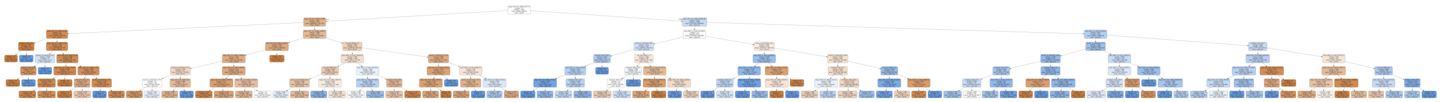
The two main paremeters tuned were maximum tree depth and the split criterion. The split criterion measures the quality of a tree split. Both 'gini' and 'entropy' were tested and evaluated using a gridsearch. For the large majority of trials, especially with larger and more complex trees, 'gini' was the more effective splitting criterion–though the results for the two were nearly identical. If anything, Gini was more performant due to it's mathematical simplicity over the entropy formula.

Tree depth provided to be a singificant influencer over the performance of our model–especially for the housing dataset. By allowing for a greater tree depth, we allow for a more complex tree with more possible decisions. A problem emerges when the tree becomes too deep, however. Instead of becoming more robust, the tree begins to overfit with very specific branches for very specific data items. By analyzing the test % vs. depth tradeoff, we can prune unnecessary tree depths for the optimal tree size. This allowed for both a fast, accurate tree with a reasonable footprint.

Performance:

Peformance varied for the two datasets across a few different domains. In terms of runtime, housing prices took longer than the visa data to train (about 10x)–though both were extremely fast on a powerful laptop. This was mainly due to a larger dataset size. As the depth of the tree increased, the training time also took longer for both datasets due to the increased tree size and complexity(as can be seen from the table above).

In terms of accuracy, the visa data started out somewhat high but was able to gain about 4% through optimizing the depth of the tree. This is due to the fact that features like 'salary' and 'job type' proved to heavily influence the approval process. The housing data, however, saw more greater improvements as depth increased. As a more diverse and variable dataset, the model was able to gain a lot of accuracy as the tree grew because it could accommodate more specific feature branches. While the test data started out at <10%, it was able to grow to approximately 76%.



SampleDecision Tree: Permanent Visa with max_depth of 7

Boosted Tree