Abstract: Gaming is becoming more and more mainstream. As it gains popularity, there begins to have monetary value associated with it, whether directly through prize pools or indirectly through betting. This project is an attempt to see if it is possible to predict the results of three different games including: Dota 2, Connect Four, and poker, based on either the starting conditions of the game, or some beginning point in the game (where a winner is not clearly evident). I attacked this problem by using decision trees, support vector machines, and random forest classifiers in an attempt to perform supervised learning on data sets for all three games.

Introduction: Dota 2 is a game that is played by millions worldwide, and has international tournaments multiple times a year with multi-million dollar prize pools. Likewise, poker is also played throughout the world, and tournaments frequently see huge amounts of money change hands. On the other hand, Connect Four is a children's game that while still wildly popular, does not have quite as much money at stake as the other games. Because of these reasons, I expect that the classifiers for Dota 2 and poker will perform very poorly, while the performance of the classifiers on Connect Four will do better.

Method: First, I had to clean the data files. For the Dota 2 file, I was able to leave it alone, since the data was already well formatted. However, due to the layout of the data, I used the first column as the Y, since the first column held the results of the game (win/loss). For the Connect Four data, I replaced the symbols that they used with 1, 0, and -1 so that the helper functions that I used would work correctly. For the poker dataset, I first merged the pre-split training and test sets. Then, I had to delete the rows with a hand that was either a straight flush or a royal flush, since those hands were too rare to properly perform machine learning.

For each of the data sets, I first shuffled them so that the order of the samples would be randomized. Then, I split the data set in three ways: 80/20, 50/50, and 20/80 (the first number refers to the size of the training set, and the second number refers to the size of the test set). Then on each of the split data sets, I ran each of the three classifiers, using 5-fold cross validation and grid search in order to determine the best hyperparameters. After I ran the three classifiers, I performed analysis on each. I determined the best hyperparameters for each of the classifiers. Then, using those parameters, I created a heatmap of accuracies per fold of the cross validation, and calculated the average training and validation accuracies for each classifier. After this, I ran the test set through the predict method of the classifier, and used a helper method to determine the test accuracy.

For the decision tree classifier, the heatmap was drawn showing the validation accuracy per fold with respect to depth, similar to what I did in homework 6.
For the support vector machine classifier, I used a linear kernel in order to improve the runtime of the computations. For the heatmaps, I displayed both the training accuracy and the validation accuracy per fold with respect to C (in separate graphs).
For the random forest tree, I had to do my heatmap slightly differently. Since I am using grid search and cross validation to determine two hyperparameters (the optimal n_estimator and the optimal max_depth), I could not display the heatmaps for all of the combinations. Instead, the

heatmap shows the accuracies for the best combination of hyperparameters of the random forest tree.

Unfortunately, I had to lower the number of hyperparameters that I tested for the support vector machine on the poker hands dataset due to time issues. I had left it running for over 14 hours without even one of the splits completing.

After I was done with the main experiments, I decided to test what would happen if I ran the decision tree algorithm on the poker hands when the data set contained fewer than the full five cards.

Experiment: For the full data, check the heatmaps section at the end to see the accuracies and optimal hyperparameters that I achieved for each section.

As predicted, the Dota 2 dataset performed poorly across the board, with accuracies in the low 50's. Likewise, the poker hands dataset also performed nearly as bad. The Connect Four dataset performed much better, but still was far below what a "good" dataset would achieve.

When I limited the number of cards available to the classifier in the poker hands data set, the accuracies became increasingly worse.

Conclusion: For a game to be fun, there has to be some element of randomness, where either player can win. After all, what is the point to playing if the outcome is predetermined? For this reason, I predict that machine learning will never be able to fully classify game based datasets.

However, there are improvements that can be made to the trials that I have done. Across the board, I have had to limit the number of hyperparameters that I tested, since I did not have access to very powerful machines or weeks to run the programs.

Additionally, there are elements that could have been improved upon in the data sets / data collection. For the Dota 2 set, the data set included the result, the location, the game mode and type, and the heroes that were selected. In addition, the games included in the dataset were played over a two hour period on a single day. Due to the complexity of Dota 2, this data could be improved drastically:

- Since Dota 2 changes frequently (patches are made regularly to the game), historical data over the course of a few patches could have more interesting results.
- By limiting the game mode to ranked, it could reduce the number of "trolls", or people playing suboptimally to have fun. Alternatively, the dataset could instead be focused on professional matches rather than casual matches, as the professional players would almost certainly be trying their best. Additionally, certain other factors (teamwork, familiarity with the role/hero, individual performance, etc.) that might influence results would be limited as much as possible.

The Connect Four data set contains all legal positions where 4 pieces have been played by each player and the next move is not forced. Due to this, there are not many improvements that I can think of, but there are certain additions that could be made for potentially more interesting results:

- Include positions where the next move is forced.
- Include positions where different numbers of pieces have been played.

The poker hands data set contains a selection of valid poker hands (including the suit of each card), as well as if the hand makes a valid poker hand. There are some improvements that could be made:

- Include all of the hands rather than just some of them. (The size of the data set is roughly 1 million samples, while the number of valid hands of five cards is $52*51*50*49*48 \approx 300$ million)
- Classify the hands into valid or invalid instead of separate categories for each valid poker hand.

It was interesting to note that the accuracies for the limited card hands were about the same. In fact, the difference in accuracy between having all five cards available and having only one card available was roughly 5%. Note that in this case, since there are more than 2 possible values that the classifier could predict, an accuracy of 50% is no longer the worst. It was interesting to see that the classifier performed much better than blind guessing, which would have resulted in an accuracy of 10%.

References:

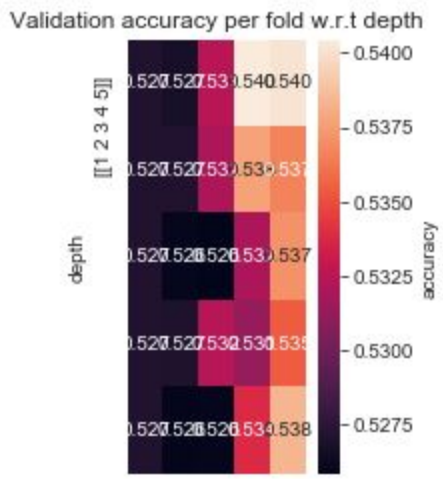
1. **http://archive.ics.uci.edu/ml/datasets/Dota2+Games+Results**
2. **http://archive.ics.uci.edu/ml/datasets/Connect-4**
3. **http://archive.ics.uci.edu/ml/datasets/Poker+Hand**
4. **https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html**
5. **https://scikit-learn.org/stable/modules/tree.html**
6. **https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html**
7. **https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html**

Bonus Points: I believe that I deserve some bonus points for the following reasons:

- Large data sets - the average size of the data sets that I used was around 1 million samples.
- Additional effort cleaning and formatting the data sets - the data sets that I used were not perfect initially, and I had to spend some time formatting them so that they would work correctly with the algorithms that I used.
- Additional experiments with fewer cards in the poker hands data set - I performed additional trials with the decision tree classifier on the poker hands data set by limiting the number of cards available in each sample to the classifier.

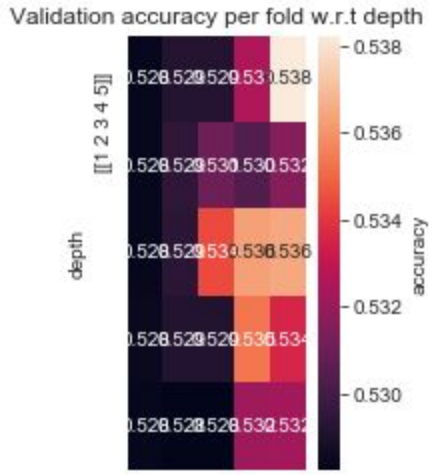
Heatmaps and Statistics:

Dota 2, Decision Tree, 80/20



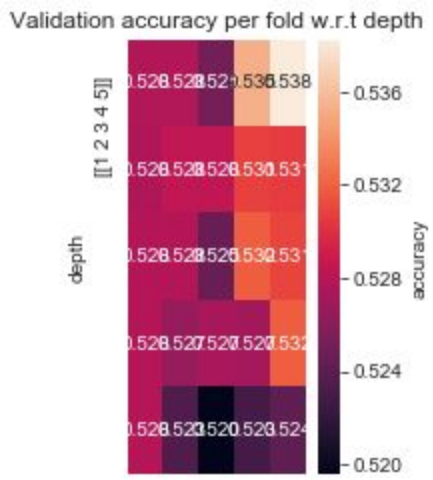
Optimal maximum depth = 5
Average Train Error = [0.52713254 0.52736628 0.53121852 0.53631537 0.54193432]
Average Test Error = [0.52713254 0.52658612 0.52977961 0.53507377 0.53756299]
Test accuracy = 0.5378114527174704

Dota 2, Decision Tree, 50/50



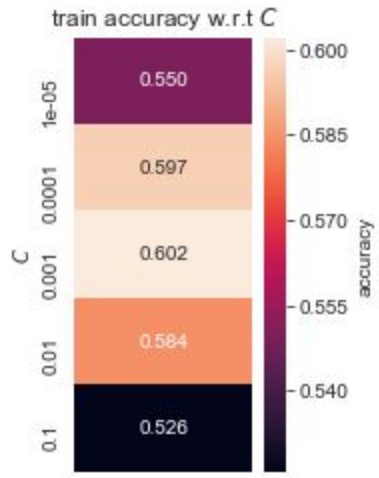
Optimal maximum depth = 5
Average Train Error = [0.52842322 0.52908377 0.53199316 0.53824896 0.54195491]
Average Test Error = [0.52842322 0.52908377 0.53040488 0.53335794 0.53444591]
Test accuracy = 0.5323476841778054

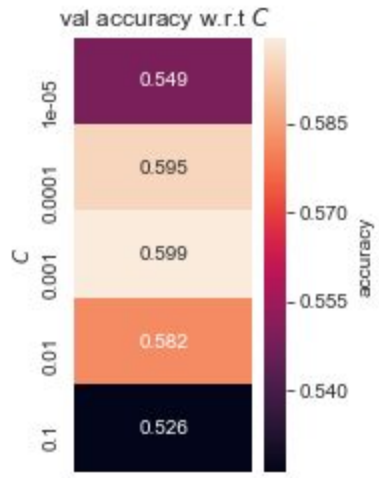
Dota 2, Decision Tree, 20/80



Optimal maximum depth = 5
Average Train Error = [0.52778317 0.52830532 0.53078244 0.53798331 0.54405482]
Average Test Error = [0.52778317 0.52681173 0.52501457 0.52962891 0.53118321]
Test accuracy = 0.5364029336053232

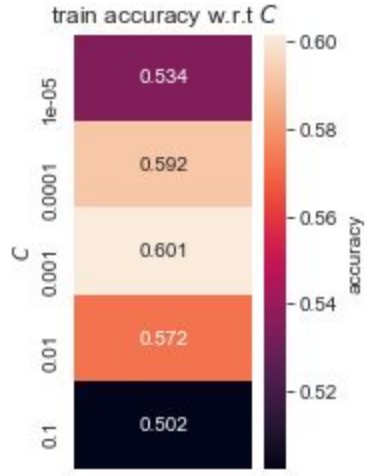
Dota 2, SVM, 80/20

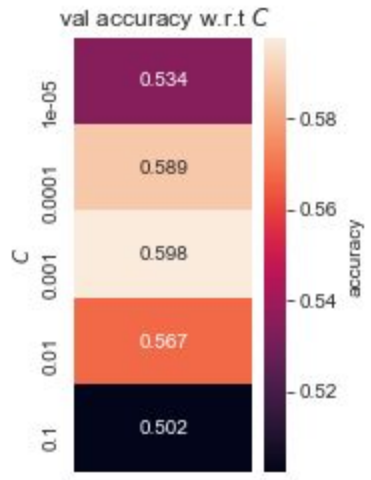




Test accuracy = 0.5944922045752586
Optimal C = 0.001
Average Train Error = [0.54996661 0.59652723 0.60210673 0.58443016 0.52561473]
Average Test Error = [0.54900127 0.59510655 0.59935644 0.58166474 0.52617327]

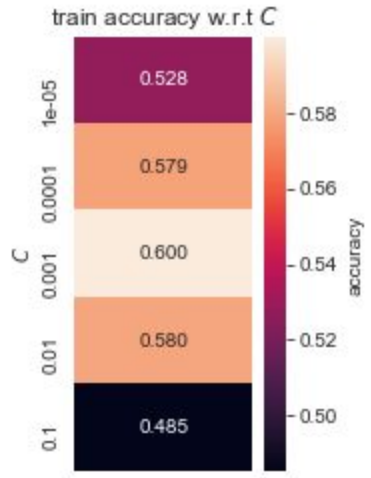
Dota 2, SVM, 50/50

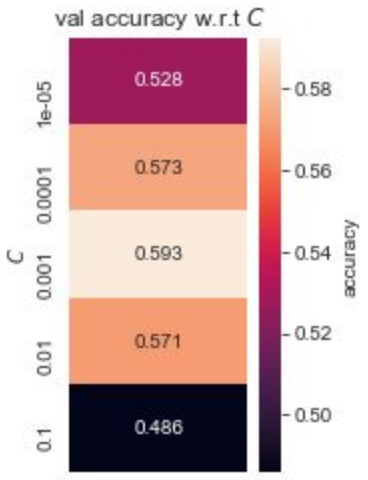




Test accuracy = 0.5972373329188685
Optimal C = 0.001
Average Train Error = [0.53445563 0.59201604 0.60148236 0.5720245 0.50215207]
Average Test Error = [0.53425163 0.58888328 0.59762589 0.56745415 0.50209823]

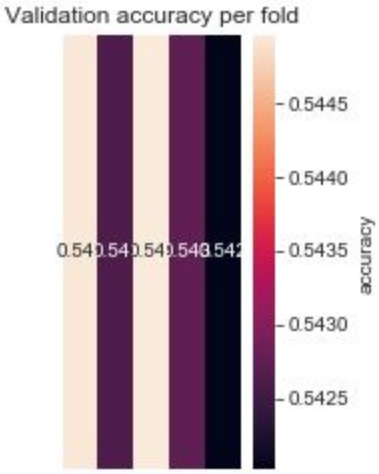
Dota 2, SVM, 20/80





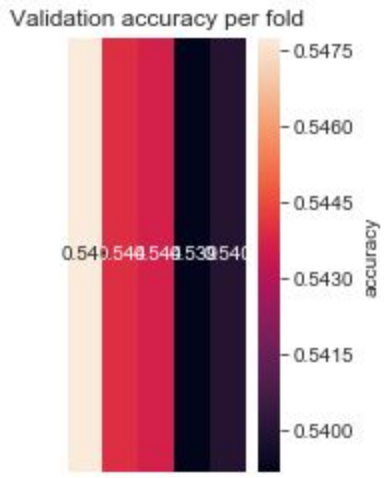
Test accuracy = 0.5961799990286075
Optimal C = 0.001
Average Train Error = [0.52778317 0.57873516 0.59994895 0.57959739 0.48530651]
Average Test Error = [0.52778317 0.57344084 0.59252963 0.57115796 0.48571984]

Dota 2, Random Forest, 80/20



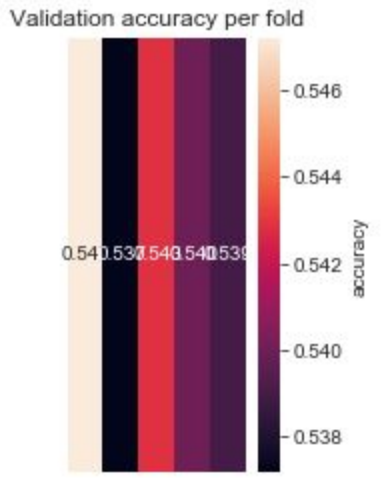
Optimal maximum depth = 5
Optimal n estimators = 12
Average Train Error = 0.5501669601210015
Average Test Error = 0.5434521279825147
Test accuracy = 0.5431541114187187

Dota 2, Random Forest, 50/50



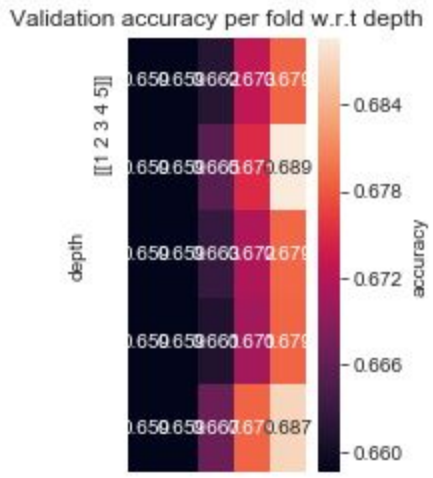
Optimal maximum depth = 5
Optimal n estimators = 10
Average Train Error = 0.5527714099545509
Average Test Error = 0.5428971091078645
Test accuracy = 0.5425085483369599

Dota 2, Random Forest, 20/80



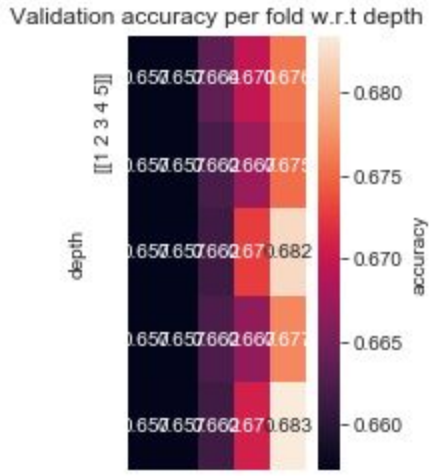
Optimal maximum depth = 5
Optimal n estimators = 10
Average Train Error = 0.5613342853718702
Average Test Error = 0.5412376141441616
Test accuracy = 0.5462747097964933

Connect Four, Decision Tree, 80/20



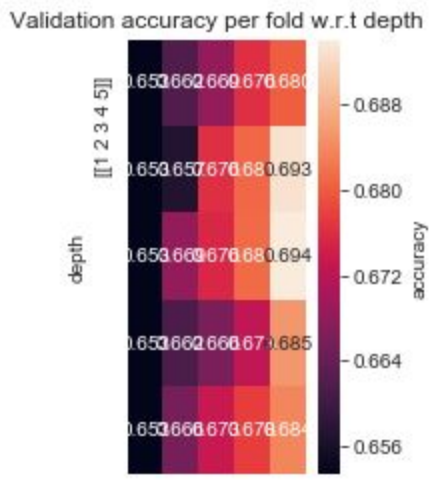
Optimal maximum depth = 5
Average Train Error = [0.65859932 0.65859932 0.66390509 0.67397078 0.68434639]
Average Test Error = [0.65859932 0.65859932 0.66357665 0.67380886 0.68243131]
Test accuracy = 0.6839105979869745

Connect Four, Decision Tree, 50/50



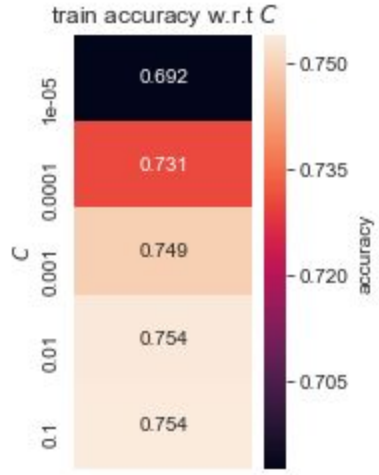
Optimal maximum depth = 5
Average Train Error = [0.65729173 0.65729173 0.66235419 0.6718056 0.68201927]
Average Test Error = [0.65729173 0.65729173 0.6623542 0.66954823 0.67881461]
Test accuracy = 0.6848633766541342

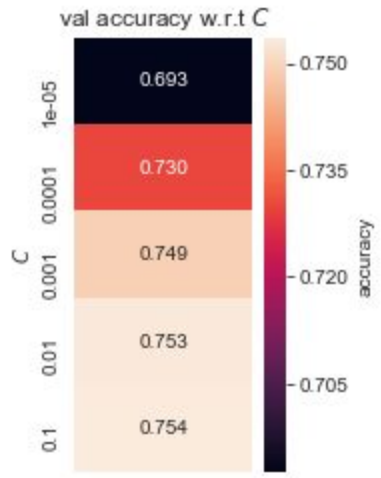
Connect Four, Decision Tree, 20/80



Optimal maximum depth = 5
Average Train Error = [0.65339353 0.66529117 0.67173037 0.68386866 0.69669139]
Average Test Error = [0.65339353 0.66294131 0.67197099 0.67781807 0.68729184]
Test accuracy = 0.6971283721274469

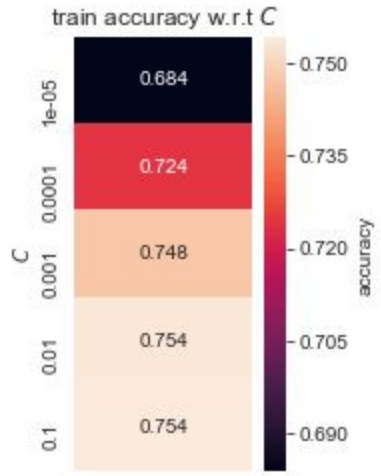
Connect Four, SVM, 80/20

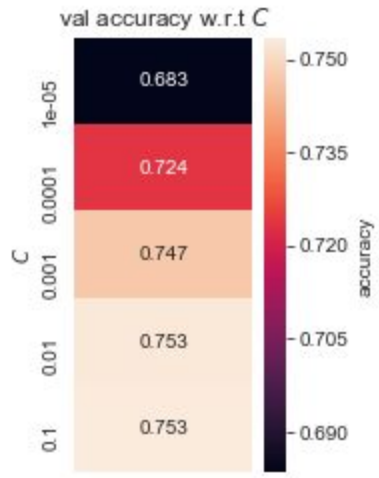




Test accuracy = 0.7549585553582001
Optimal C = 0.1
Average Train Error = [0.69248312 0.73054862 0.74935702 0.75369599 0.75403368]
Average Test Error = [0.6925155 0.73009529 0.74915348 0.75318716 0.75357572]

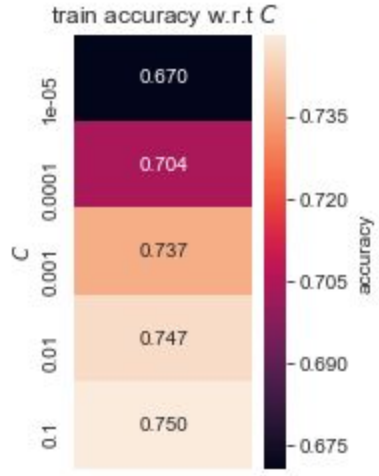
Connect Four, SVM, 50/50

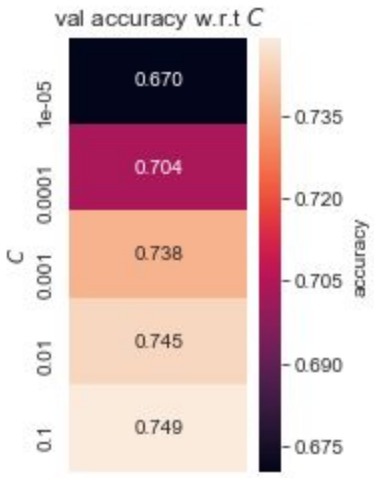




Test accuracy = 0.7547884780484917
Optimal C = 0.1
Average Train Error = [0.68381048 0.72429541 0.74754278 0.75359701 0.75421131]
Average Test Error = [0.68334419 0.72378471 0.74702469 0.7529161 0.7534786]

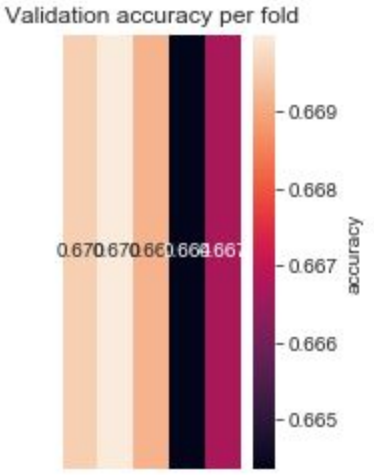
Connect Four, SVM, 20/80





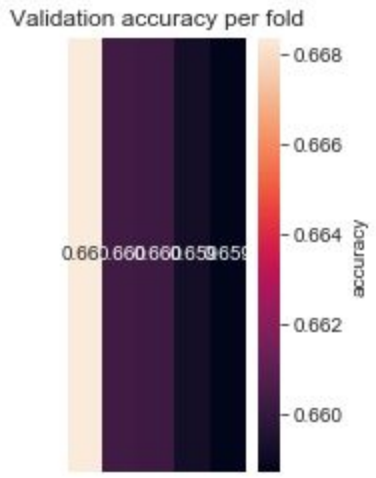
Test accuracy = 0.7546349406061503
Optimal C = 0.1
Average Train Error = [0.67049064 0.70370431 0.73719565 0.74659541 0.74988901]
Average Test Error = [0.67026867 0.70350085 0.73754718 0.74487455 0.74909333]

Connect Four, Random Forest, 80/20



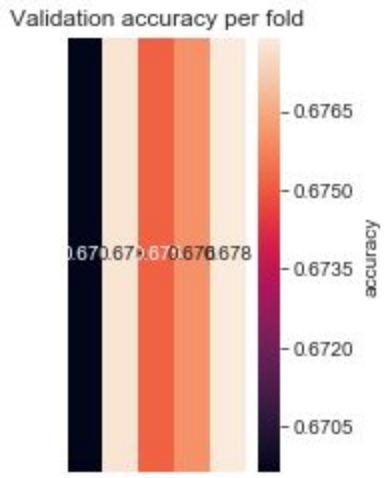
Optimal maximum depth = 5
Optimal n estimators = 5
Average Train Error = 0.6678740053778305
Average Test Error = 0.6679618836155056
Test accuracy = 0.6859088217880402

Connect Four, Random Forest, 50/50



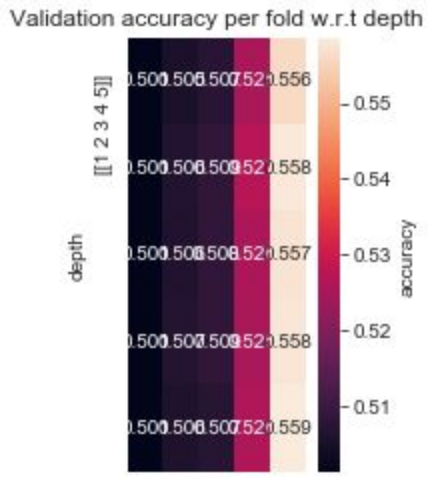
Optimal maximum depth = 5
Optimal n estimators = 5
Average Train Error = 0.6616215544930802
Average Test Error = 0.6613772277813962
Test accuracy = 0.6848337724621807

Connect Four, Random Forest, 20/80



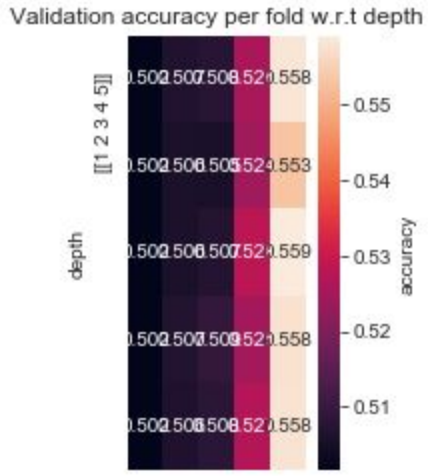
Optimal maximum depth = 5
Optimal n estimators = 8
Average Train Error = 0.6788726821054881
Average Test Error = 0.6753016060987344
Test accuracy = 0.6736298708507568

Poker, Decision Tree, 80/20



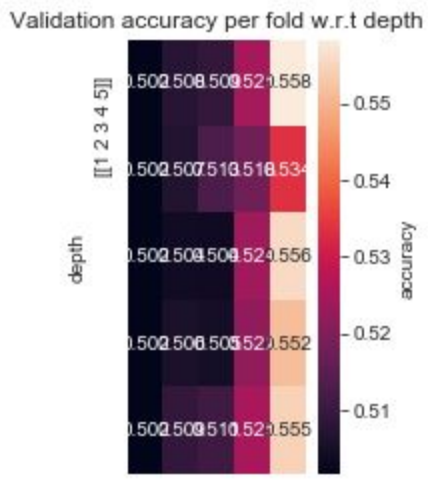
Optimal maximum depth = 5
Average Train Error = [0.50126706 0.50642341 0.50834474 0.52632108 0.55768968]
Average Test Error = [0.50126706 0.50616701 0.50806699 0.52602536 0.55768968]
Test accuracy = 0.5564823757816997

Poker, Decision Tree, 50/50



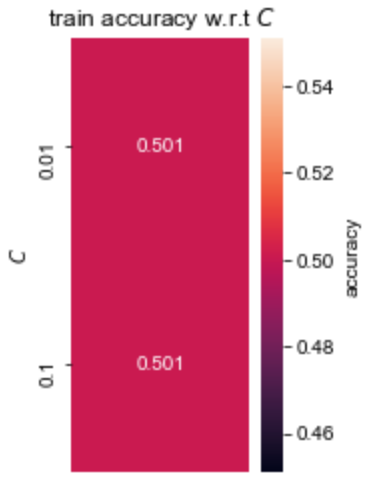
Optimal maximum depth = 5
Average Train Error = [0.50154828 0.50654384 0.50833504 0.52627096 0.55720237]
Average Test Error = [0.50154828 0.50624482 0.50750334 0.52612755 0.55720237]
Test accuracy = 0.5576940712773534

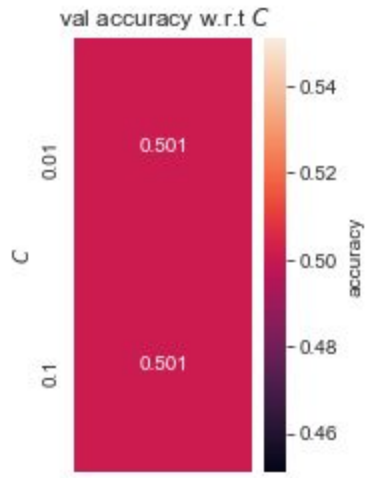
Poker, Decision Tree, 20/80



Optimal maximum depth = 5
Average Train Error = [0.50160486 0.50632677 0.50927065 0.52496925 0.55172506]
Average Test Error = [0.50160486 0.50666335 0.50830724 0.52285831 0.55074585]
Test accuracy = 0.557842118613477

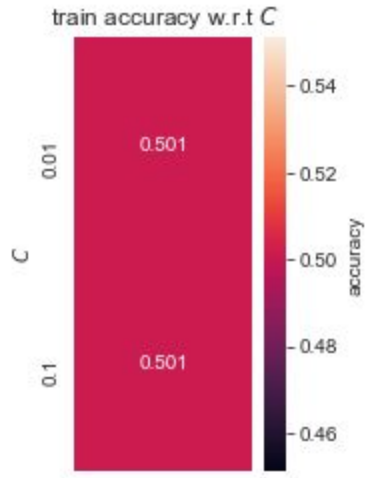
Poker, SVM, 80/20

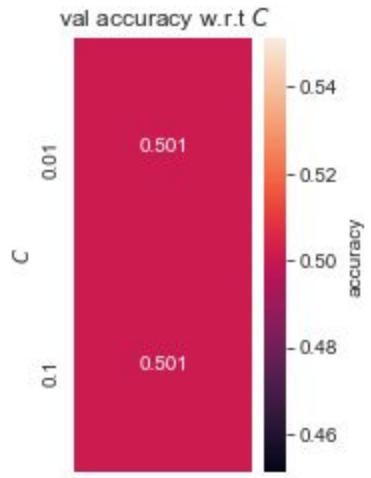




Test accuracy = 0.5017049029985805
Optimal C = 0.01
Average Train Error = [0.5010488 0.5010488]
Average Test Error = [0.5010488 0.5010488]

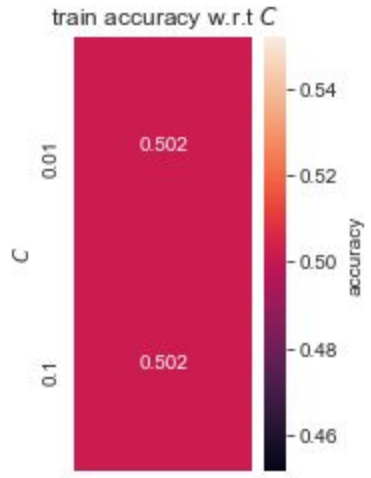
Poker, SVM, 50/50

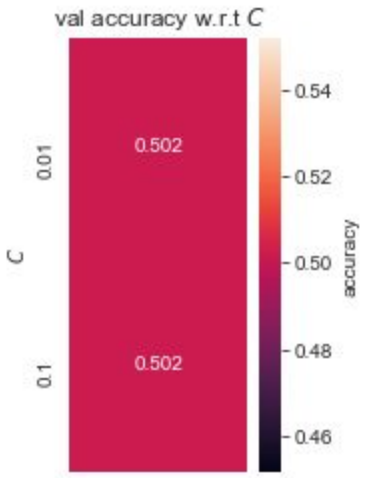




Test accuracy = 0.5011814795519158
Optimal C = 0.01
Average Train Error = [0.50117855 0.50117855]
Average Test Error = [0.50117855 0.50117855]

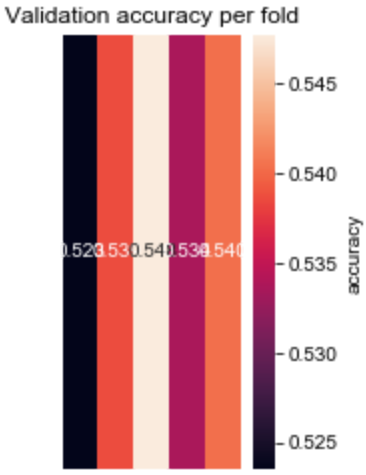
Poker, SVM, 20/80





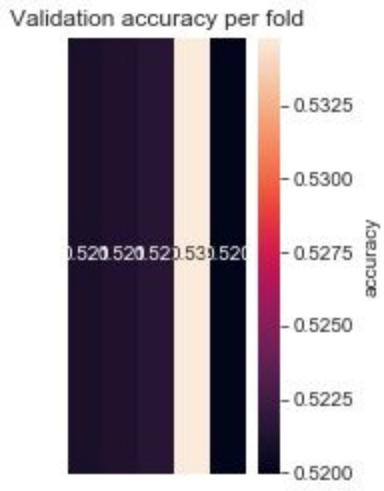
Test accuracy = 0.50103172241545
Optimal C = 0.01
Average Train Error = [0.5017732 0.5017732]
Average Test Error = [0.5017732 0.5017732]

Poker, Random Forest, 80/20



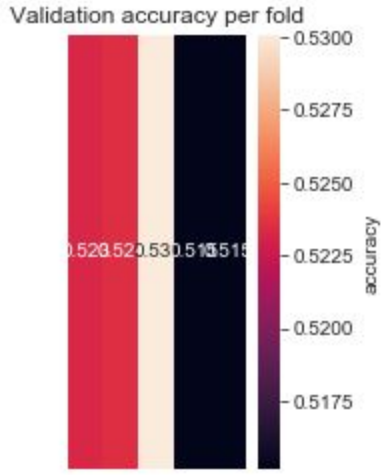
Optimal maximum depth = 5
Optimal n estimators = 8
Average Train Error = 0.5369130197277798
Average Test Error = 0.536794440894257
Test accuracy = 0.5468665395103343

Poker, Random Forest, 50/50



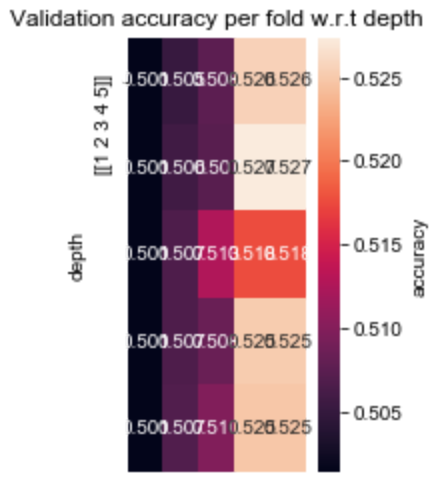
Optimal maximum depth = 5
Optimal n estimators = 5
Average Train Error = 0.5237349974161792
Average Test Error = 0.5237037846444432
Test accuracy = 0.5544329385962344

Poker, Random Forest, 20/80



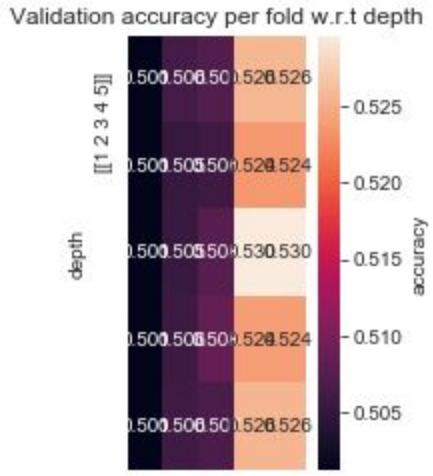
Optimal maximum depth = 5
Optimal n estimators = 5
Average Train Error = 0.5216723383531523
Average Test Error = 0.5214320209564043
Test accuracy = 0.5359407698649249

Poker (4 cards), Decision Tree, 80/20



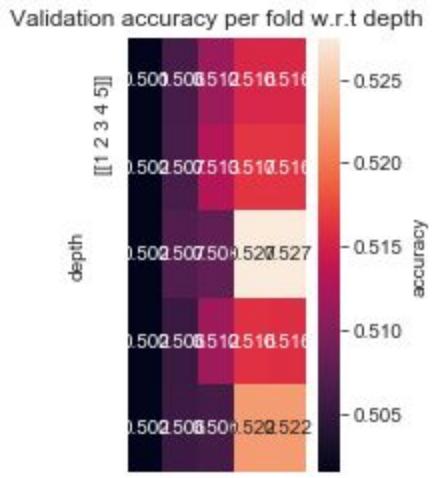
Optimal maximum depth = 4
Average Train Error = [0.50133904 0.50634003 0.50924312 0.52420798 0.52420798]
Average Test Error = [0.50133904 0.50621716 0.50915135 0.5241345 0.5241345]
Test accuracy = 0.5250125611594316

Poker (4 cards), Decision Tree, 50/50



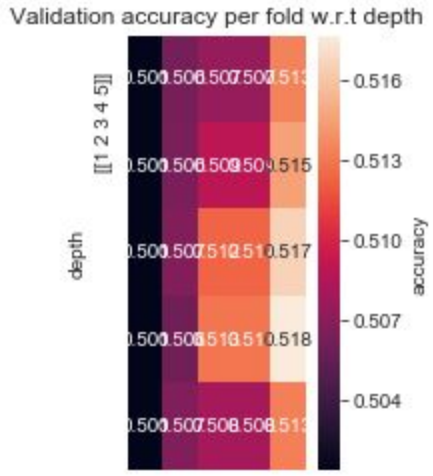
Optimal maximum depth = 4
Average Train Error = [0.50113368 0.50624351 0.50781768 0.52578772 0.52578772]
Average Test Error = [0.50113368 0.50574253 0.50720011 0.52578772 0.52578772]
Test accuracy = 0.525390590700751

Poker (4 cards), Decision Tree, 20/80



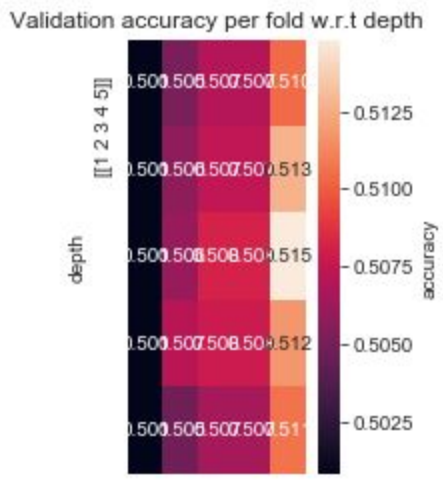
Optimal maximum depth = 4
Average Train Error = [0.50151953 0.50631229 0.51045017 0.51998197 0.52011246]
Average Test Error = [0.50151953 0.5062318 0.51014893 0.51962712 0.51959297]
Test accuracy = 0.5257296448240706

Poker (3 cards), Decision Tree, 80/20



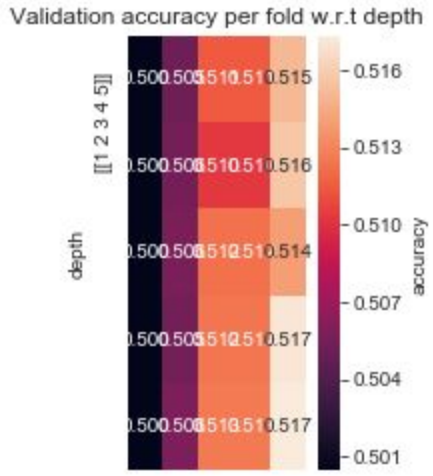
Optimal maximum depth = 5
Average Train Error = [0.50131953 0.50635406 0.51006051 0.51006051 0.51570175]
Average Test Error = [0.50131953 0.50643546 0.50997941 0.50997941 0.51514169]
Test accuracy = 0.5178173339122036

Poker (3 cards), Decision Tree, 50/50



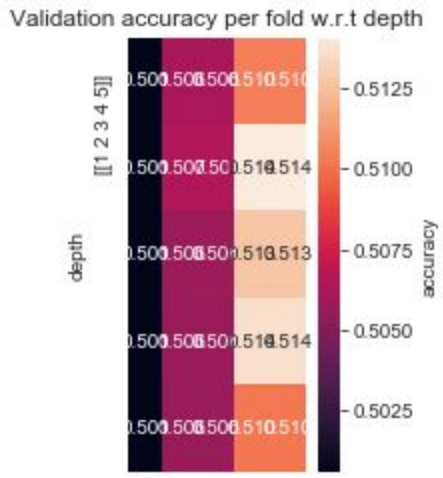
Optimal maximum depth = 5
Average Train Error = [0.50080782 0.5058596 0.50742451 0.50742451 0.51286069]
Average Test Error = [0.50080782 0.5058596 0.50742451 0.50742451 0.51210165]
Test accuracy = 0.5127952967162478

Poker (3 cards), Decision Tree, 20/80



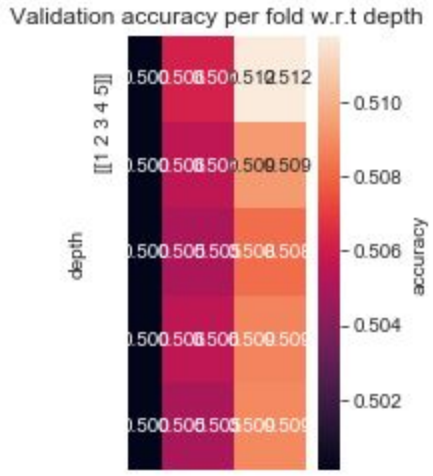
Optimal maximum depth = 5
Average Train Error = [0.50048049 0.50553911 0.511827   0.511827   0.51595755]
Average Test Error = [0.50048049 0.50553911 0.511827   0.511827   0.51585145]
Test accuracy = 0.5162599940486934

Poker (2 cards), Decision Tree, 80/20



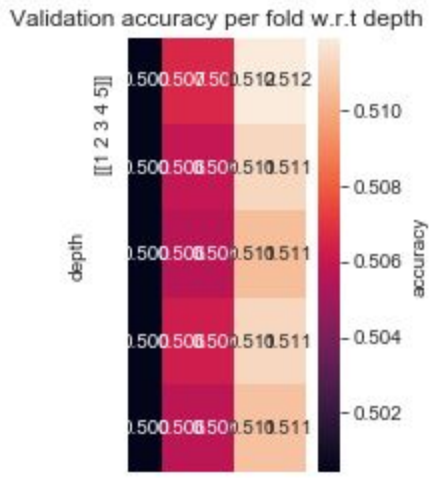
Optimal maximum depth = 4
Average Train Error = [0.50057074 0.50582783 0.50582783 0.51210597 0.51210597]
Average Test Error = [0.50057074 0.50601838 0.50601838 0.51223311 0.51223311]
Test accuracy = 0.514856314970463

Poker (2 cards), Decision Tree, 50/50



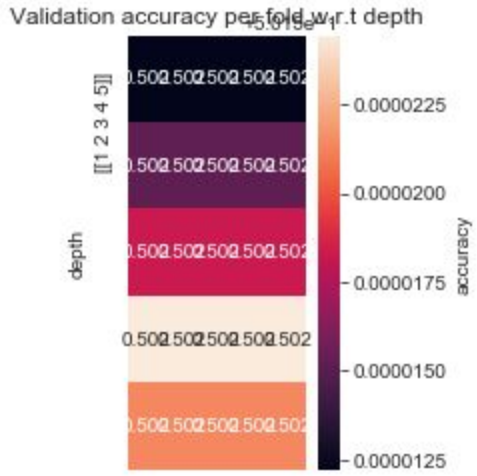
Optimal maximum depth = 4
Average Train Error = [0.50010732 0.50546643 0.50546643 0.50943674 0.50943674]
Average Test Error = [0.50010732 0.50550057 0.50550057 0.50939917 0.50939917]
Test accuracy = 0.5106020960286287

Poker (2 cards), Decision Tree, 20/80



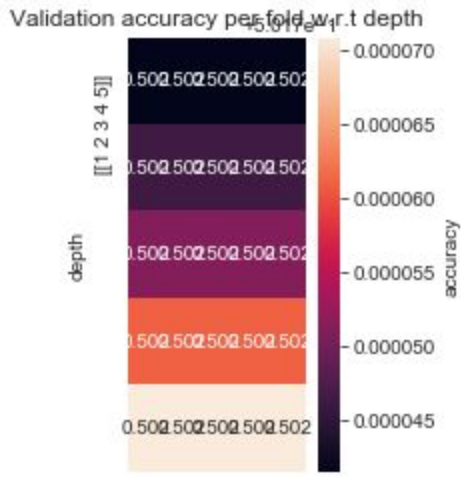
Optimal maximum depth = 4
Average Train Error = [0.50045123 0.5060757 0.5060757 0.51131114 0.51131114]
Average Test Error = [0.50045123 0.5060757 0.5060757 0.51113431 0.51113431]
Test accuracy = 0.511422118372464

Poker (1 card), Decision Tree, 80/20



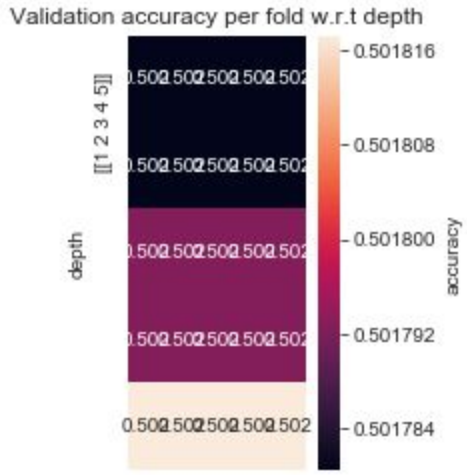
Optimal maximum depth = 1
Average Train Error = [0.50151831 0.50151831 0.50151831 0.50151831 0.50151831]
Average Test Error = [0.50151831 0.50151831 0.50151831 0.50151831 0.50151831]
Test accuracy = 0.49982682673404977

Poker (1 card), Decision Tree, 50/50



Optimal maximum depth = 1
Average Train Error = [0.50175417 0.50175417 0.50175417 0.50175417 0.50175417]
Average Test Error = [0.50175417 0.50175417 0.50175417 0.50175417 0.50175417]
Test accuracy = 0.500605861933724

Poker (1 card), Decision Tree, 20/80



Optimal maximum depth = 1
Average Train Error = [0.50179271 0.50179271 0.50179271 0.50179271 0.50179271]
Average Test Error = [0.50179271 0.50179271 0.50179271 0.50179271 0.50179271]
Test accuracy = 0.5010268442952823