MATH 490 Final Project

Diabetes Prediction for Pima Indians using Cost Complexity Pruning and Cross Validation

James Della-Guistina, Mitchell MacAdams, Kevin Martindale, Jason Riley, Jr.

May 14, 2021

Towson University

Our Data Set

Pima Indian Diabetes Dataset - EDA and Prediction

- Originally from the National Institute of Diabetes, Digestive, and Kidney Diseases. [7] [6]
- Collected from females with Pima Indian ancestral heritage least 21 years old and presented by Smith, Everheart, Dickson, et. al.[11].
- Objective of the dataset is to predict whether or not a patient would develop diabetes within 5 years based on certain diagnostic measurements.
- Original Data: 768 Data points with 8 attributes.
- After we cleaned the data: 733 Data points with 8 attributes.

Structure of Our Project

Goal: We wanted to find the optimal Decision Tree that would most accurately predict whether or not a patient would develop diabetes within 5 years.

How we Accomplished this:

1. Began by getting a baseline of how accurate a decision tree would be with no modifications/parameters.
   - We tested the accuracy using a **Confusion Matrix**.
2. Performed **Cost-Complexity Pruning** to determine the optimal decision tree for a particular training and test set.
3. Used **5-fold Cross-Validation** to redetermine the optimal decision tree for all possible training and test sets.

What are Decision Trees?

A **decision tree** is a supervised machine learning algorithm. Decision trees use human rule based decision making to organize and classify data based on attributes (parameters). Then, once given data, the decision tree will be able to then classify new data into a particular outcome. [4]

Decision Tree Algorithm Steps

Parts of a Decision Tree [5]:

- The **root node** is the initial classification attribute.
- The **branches** are the different values that can be assigned.
- More **nodes** can be added to give more accurate classification to the data.
- Finally, we end with a **leaf** which can be seen as the outcome of the data.

Steps to algorithm [14]:

1. Determine the best attribute that splits the data and ask relevant question (node).
2. Determine the answers of that question(branches).

3a. Sent to a node: Repeat steps until you've answered your question.

3b. Sent to a leaf: Data is classified.

Entropy & Information Gain

The **entropy**[13] H of a discrete random variable X with outcomes $x_1, x_2, ..., x_k$ and probabilities $P(x_1), P(x_2), ..., P(x_k)$, respectively, is

$$H(X) = -\sum_{i=1}^{k} P(x_i) \log_2 P(x_i)$$

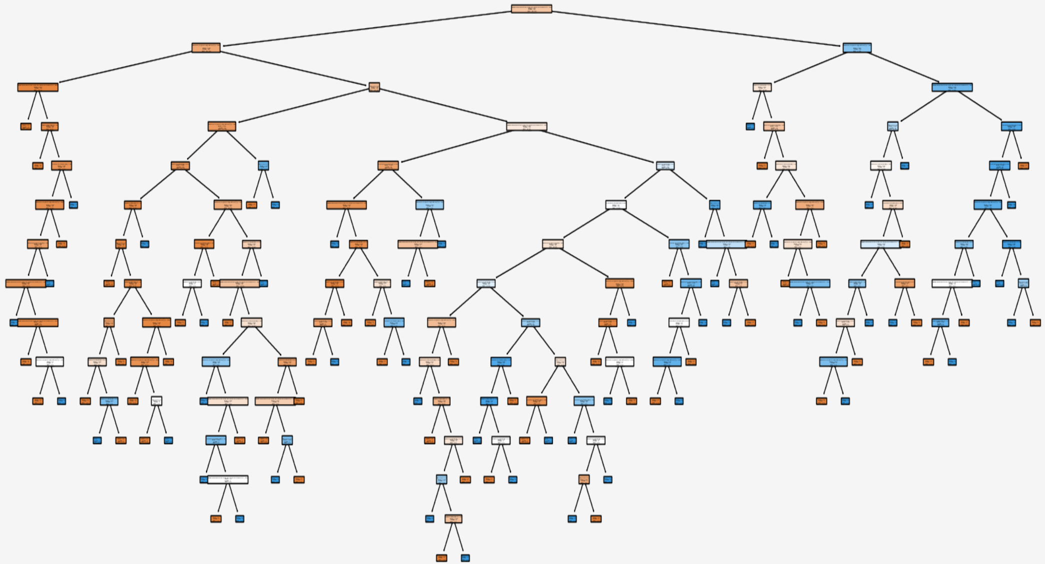
Information gain[13] is the change in entropy after splitting the data into subsets $S_1, S_2, ...S_j$ using the decision tree. We can calculate this by using

$$IG(X, split) = H(X_0) - H(X, split)$$

$$\text{where } H(X, split) = \sum_{i=1}^{j} P(S_i)H(X, S_i)$$

Recall: Entropy and Information Gain are <u>negatively correlated</u>

Decision Tree Without Specified Parameters



Confusion Matrix

A **Confusion Matrix** [10] is a way to show the predicted labels versus the true labels for a given test set.

$$\frac{29}{42} = 69.05\%$$

True predictions for non-diabetic cases.

$$\frac{10}{24} = 41.7\%$$

True predictions for diabetic cases.

Overall our model produced a (59.09%) accuracy.

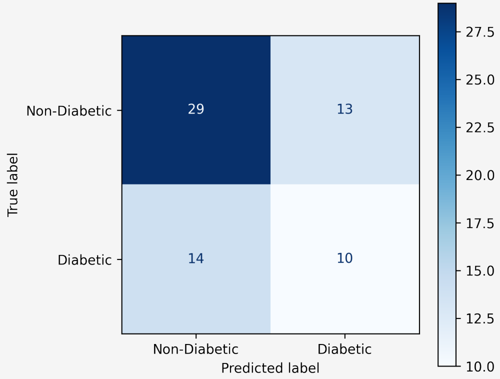


Figure 1: Confusion Matrix for the decision tree constructed with no parameters passed to the arguments.

Pruning in Decision Trees

- Decision Trees may give bad results when given new values, so we introduce "Pruning".
- Think Real World - removal of certain parts of a tree to promote healthy growth.
- The same idea is used for Decision Trees!
- This helps the tree adapt when given new data and fixes over-fitting.
- Reduces Tree size and increases training error but decreases testing error.

Cost Complexity Pruning

- 'Pre-prune' by carefully selecting certain parameters to pass as arguments when we build our tree.
- 'Post-prune' by letting our tree grow unchecked and optimize after the fact.
- We choose 'post-prune' in a process called cost complexity pruning.
- We define a **regularization** parameter α and a cost complexity function[2] R_α :

$$R_\alpha = R(T) + \alpha|T|$$

- $|T|$ is the number of leaves in our tree, $R(T)$ is the misclassification rate of the terminal nodes [8].
- So specifically we are looking for the sub-tree $T(\alpha)$ where:

$$R_\alpha(T(\alpha)) = min_{T \leq T_{max}} R_\alpha(T)$$

Cost Complexity Pruning In Action

Scikit provides some functions to help us:

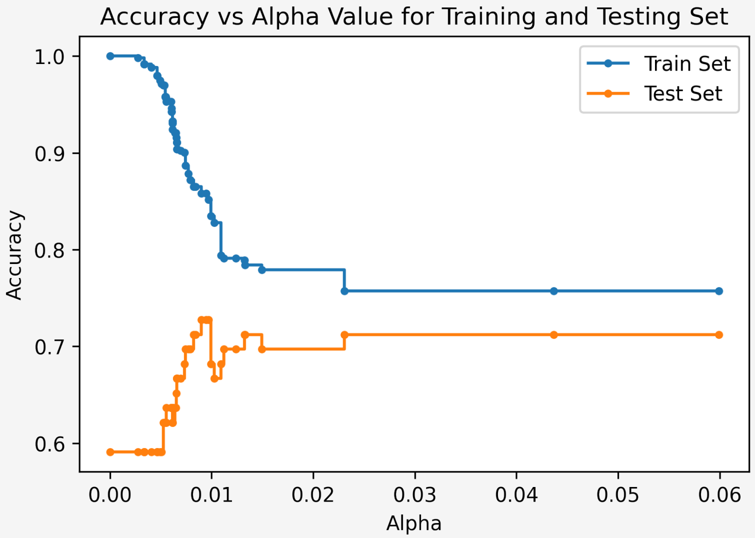
- `cost_complexity_pruning_path`
- Creates a nested sequence of sub-trees, with our full tree at the top and a sub-tree consisting of only the root node at the very bottom.
- Since there are only a finite number of sub-trees, every one corresponds to some α.
- This minimizes the cost function defined above. [2]

Cost Complexity Pruning Code Snippet

```
tree_path=first_dt.cost_complexity_pruning_path(X_train,Y_train)
ccp_alphas=tree_path.ccp_alphas
ccp_alphas=ccp_alphas[:-1]
first_dts=[]

for ccp_alpha in ccp_alphas:
first_dt = DecisionTreeClassifier(criterion='entropy',random_state=20,
ccp_alpha=ccp_alpha)
first_dt.fit(X_train, Y_train)
first_dts.append(first_dt)
```

Analysis of Pruning Parameter α



Results from 'Optimal' α

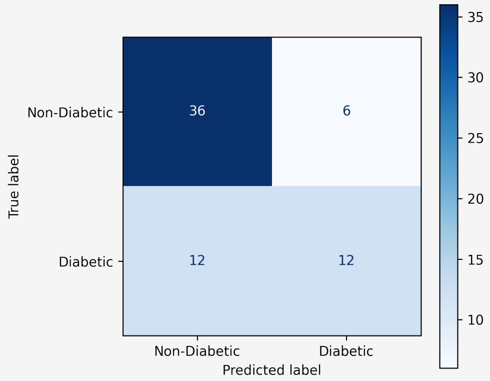


Figure 2: Confusion Matrix for the decision tree constructed with the first chosen alpha parameter passed to the arguments with an accuracy of 72.73%.

$$\frac{36}{42} = 85.71\%$$

True predictions for non-diabetic cases.

$$\frac{12}{24} = 50.0\%$$

True predictions for diabetic cases.

'Optimal' Decision Tree

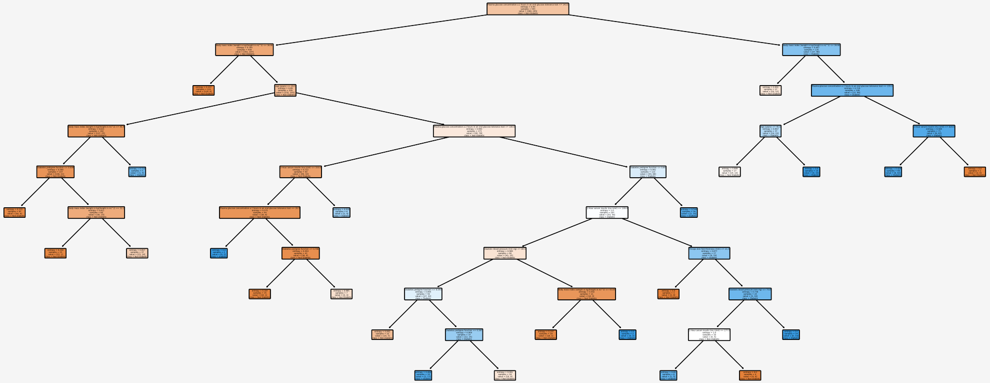
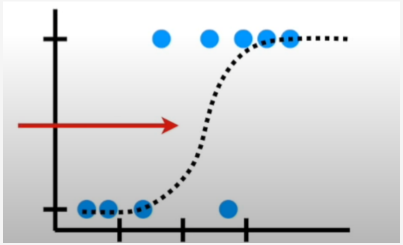


Figure 3: Our first choice of α that results in a tree of depth 10 and 24 leaves.

Cross Validation In Machine Learning

- **Cross validation** is a statistical method used to estimate the performance/accuracy of the model.
- Our original training set is split into two more subsets, where one of the portions becomes a "new" testing sample that we set aside. This sample is later used for testing/validating.
- We do this to know how the method will work on data that is not trained on.
- Choosing the wrong portion of data would make certain good models useless. Cross Validation fixes this by using multiple, sequential samples that cover all data [9].

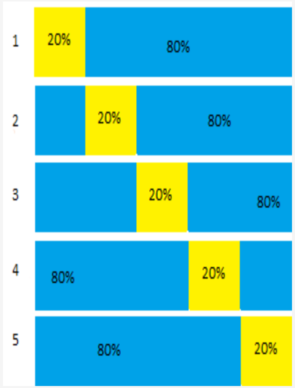
Cross Validation Continuation



When we have gotten our training data we must follow a few steps:

1. Estimate the parameters for the machine learning methods (training the algorithm).
2. Evaluate how well the machine learning method works (testing the algorithm)[12].
3. Repeat testing based on which cross validation technique is being used.

5-Fold Validation



- Select some number of folds for our training set to split into our two subsets.
- Since we have 5 portions, the data is split into 80/20 so we can cover all data evenly.
- The blue would be used to train the data while the yellow would be used to test the data.
- Cross Validation uses all trials, one at a time, and summarizes the result at the end.
- We then compare our methods by seeing how well each one categorized the test data [9].

5-fold Cross-Validation

```
mean_scores=[]

for ccp_alpha in ccp_alphas:
first_dt = DecisionTreeClassifier(criterion='entropy', random_state=20,
ccp_alpha=ccp_alpha)
scores=cross_val_score(first_dt, X.train, Y.train, cv=5)
print(scores)
mean_scores.append(np.mean(scores))
```

Mean Scores of 5-Fold Cross Validation

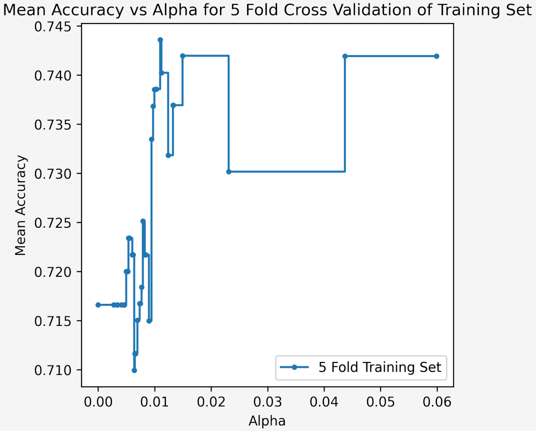


Figure 4: A plot to determine the α that will maximize our mean accuracy of the 5-fold cross validation. We can see $\alpha = .011$.

Optimal Decision Tree

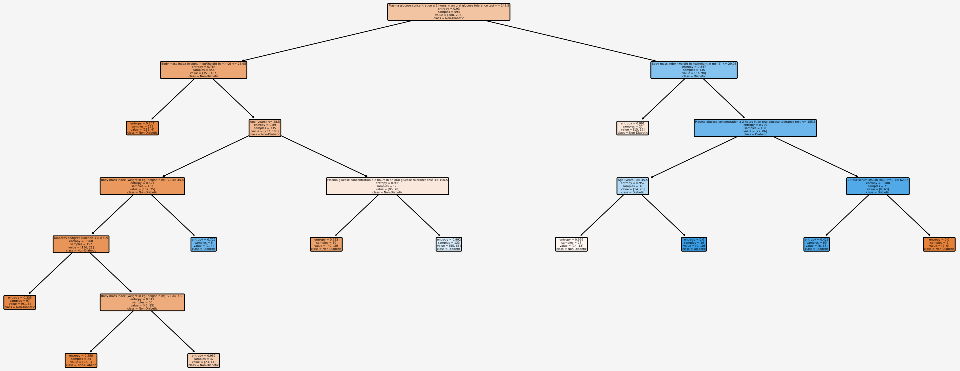


Figure 5: Our final optimized decision tree with a depth of 7 and 12 leaves.

Confusion Matrix Comparison

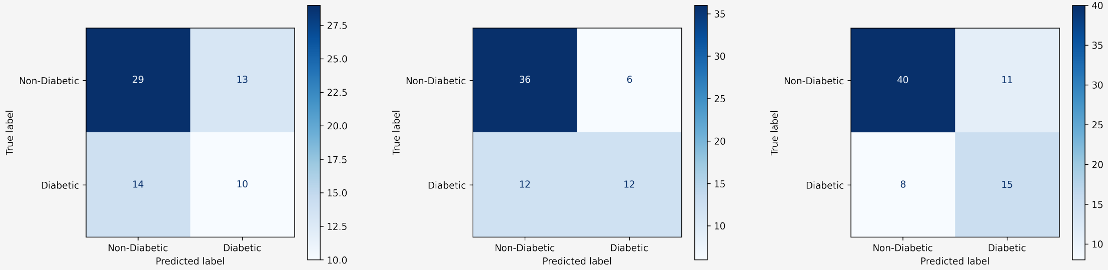


Figure 6: From left to right; the initial tree (59.09%), first choice of α (72.73%), and the α chosen based on the mean scores of 5 fold cross validation (74.32%).

Summary

1. Import the data.
2. Identify non-existing data (zeros).
3. Eliminate what we can without oversimplifying (4% of data).
4. Create decision tree built around parameters of interest.
5. Start with no arguments, but Confusion Matrix shows there is over-fitting.
6. After the first test we look to prune our tree and optimize parameters.
7. Create an array to store all decision trees for all combinations of pruned leaves.
8. Pruning and evaluating alpha gives us optimal tree for training and test data.
9. Use 5-fold cross validation to check every alpha for the best.
10. Identify the optimal decision tree produced from pruning and 5-fold cross validation.

Other Study - Artificial Neural Network

https://www.kaggle.com/gauravahujaravenclaw/neural-network-to-predict-diabetes

1. Import the data.
2. Identify non-existing data (zeros).
3. Replace zeros with mean of data for specific category.
4. Perform binning of data to combat over-fitting with such a small set.
5. Split data set for testing and training using train_test_split (80/20 Split).
6. Scale the data appropriately.
7. Create an ANN (artificial neural network) to test on subsets of training data before test data.

Artificial Neural Network Study's Result

```
0.8181818181818182, [3,3]
0.7922077922077922, [3,4]
0.7987012987012987, [3,5]
0.8051948051948052, [3,6]
0.8311688311688312, [4,3]
0.8051948051948052, [4,4]
0.8311688311688312, [4,5]
0.7987012987012987, [4,6]
0.7922077922077922, [5,3]
0.7987012987012987, [5,4]
0.7857142857142857, [5,5]
0.7922077922077922, [5,6]
0.7987012987012987, [6,3]
0.8116883116883117, [6,4]
0.8051948051948052, [6,5]
0.8051948051948052, [6,6]
```

- This model uses 2 different hidden layers and tested the number of nodes in each hidden layer to find the best model.
- According to the testing data results,
  - The best number of nodes in the first hidden layer is 4.
  - The best number of nodes in the second hidden layer is either 3 or 5.
- In these models, there is a 83.12% Accuracy.

Future Testing

If we were to test this dataset again, we would:

- Use a different Machine Learning method, such as Neural Networks.
- Increase the training set.
- Determine a better way to replace the non-existing data (zeros).

References

1.10. Decision Trees. URL: https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning.

11.8.2 - Minimal Cost-Complexity Pruning: STAT 508. URL: https://online.stat.psu.edu/stat508/lesson/11/11.8/11.8.2.

3.1. Cross-validation: evaluating estimator performance. URL: https://scikit-learn.org/stable/modules/cross_validation.html.

Kumar Barnwal. *Random Forests explained intuitively*. 2017. URL: https://www.datasciencecentral.com/profiles/blogs/random-forests-explained-intuitively.

Prashant Gupta. *Decision Trees in Machine Learning*. 2017. URL: https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052.

UCI Machine Learning. *Pima Indians Diabetes Database*. 2016. URL: https://www.kaggle.com/uciml/pima-indians-diabetes-database.

National Institute of Diabetes and Digestive and Kidney Diseases (NIDDK). 2020. URL: https://www.nih.gov/about-nih/what-we-do/nih-almanac/national-institute-diabetes-digestive-kidney-diseases-niddk.

Post pruning decision trees with cost complexity pruning. URL: https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html.

Sanjay.M. *Why and how to Cross Validate a Model?* 2020. URL: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f.

sklearn.metrics.confusion_matrix. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.

Jack W. Smith et al. *Using the ADAP Learning Algorithm to Forecast the Onset of Diabetes Mellitus*. 1988. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2245318/.

Stephanie. *Cross Validation (Statistics)*. 2020. URL: https://www.statisticshowto.com/cross-validation-statistics/.

Victor Zhou. *A Simple Explanation of Information Gain and Entropy*. 2019. URL: https://victorzhou.com/blog/information-gain/.

Victor Zhou. *Random Forests for Complete Beginners*. 2019. URL: https://victorzhou.com/blog/intro-to-random-forests/.

Questions?