Permutation Decision Tree Project Report - 4

For Partial Fulfillment of CS F266

Pranav Srinivas
2020A7PS1694G
f20201694@goa.bits-pilani.ac.in

Ravi Sanker S
2020A7PS0142G
f20200142@goa.bits-pilani.ac.in

I. Introduction

In this report, we wish to understand how the Permutation Decision Tree performs compared to a Decision Tree which uses Shannon Entropy as the criteria for split. We also wish to wish to understand how well the Permutation Decision Tree performs on five different permutation of the custom data-set we've been using so far.

To be more specific, the data-set we are using is an extensive generated Auto-Regressive Data-set with the following two equations and use it as input for the Permutation Decision Tree we had implemented in Project Report-1 [1].

$$x(t+1) = c_1 * x(t) + c_2$$

$$y(t+1) = b_1 * y(t) + b_2 * x(t) + b_3$$

Unlike what we did in Project Report-2, we have used the same Auto-Regressive co-efficients for all the features.

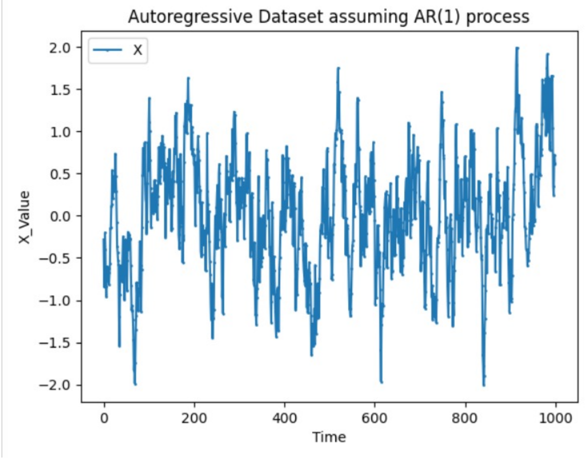
We are using Scikit-learn's TimeSeriesSplit for Cross-Validation to study the performance of the Permutation Decision Tree with Auto-Regressive Data-sets and see if it is able to correctly classify causes and effects.

In addition to this, we're also interested in understanding how well the Permutation Decision Tree performs on a real world data-set. The data-set we have chosen is Wisconsin Breast Cancer Diagnostic data-set originally obtained from the the University of Wisconsin Hospitals, Madison, from Dr. William H. Wolberg's publications.

II. Methodology

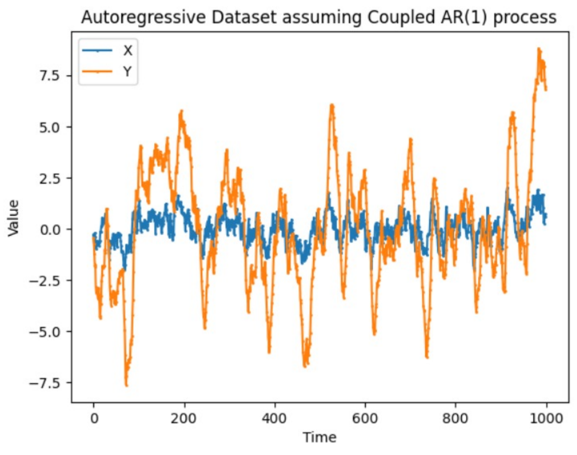
A. Generating the Auto-Regressive Data-set

Initially, we have assumed the data-set contains 10 features. However, note that this can be generalised if required. Each feature follows the equations described above independently. The constants mentioned in the equation have been hard-coded for now, except for c_2 and b_3 which have been taken as noise and are generated randomly. They are generated from a normal distribution with $mean = 0$ and $variance = \frac{1}{3}$. A quick visual inspection check for an Auto-Regressive process is to check if the data varies about a mean. Following is a plot of one of the features with time:



B. Coupled Auto-Regressive Process and the Coupling Coefficient

A coupled Auto-Regressive (AR) process is a statistical model that is useful for modeling and analyzing the relationships between multiple time series variables that can influence each other over time. The primary idea is that each time series variable is regressed on its own past values as well as the past values of other related time series variables. We have represented this using the variable y which not only depends on its past values, but also on the past values of variable x. Following is a plot of the Coupled Auto-Regressive Processes $x(t)$ and $y(t)$:



The coupling coefficient determines the strength and direction of the relationship between the variables in the system at different lags. It is represented by b_2 in our equations and has been hard-coded to the value $b_2 = 0.6$.

C. Feeding the Data-set to the Permutation Decision Tree

Before using the data to train the Decision tree, we have to remove the first few data instances to make sure we have no transient data. Transience refers to the behavior where the influence of past values on the current value of the time series gradually diminishes as you go further back in time. This means that current time series values do not depend heavily on the transient data values. For this report, we have omitted the first 100 data instances.

In the data-set that we have generated using the above-mentioned equations, $x(t)$ represents the *cause* and $y(t)$ represents the *effect*. We generated appropriate labels to train the Permutation Decision tree classifier where *Class-0* represents *cause* and *Class-1* represents *effect*.

D. Generating Five Random Permutations

To generate random permutations, we used the random number generator which generates random numbers between 0 and 1. If the value generated is less than 0.5, we add an instance of cause to the data. If the value generated is greater than 0.5, we add an instance of effect to the data.

E. Using TimeSeriesSplit

Due to the temporal nature of the data-set, we had to use a specific method called TimeSeriesSplit to split the data-set into Training and Testing. This is because unlike the regular TrainTestSplit method, TimeSeriesSplit takes care to ensure that the temporal structure of the data is not disturbed in order to ensure that there is no bias involved when training the classifier.
We made 10 splits and evaluated the performance of the Permutation Decision Tree on different possible temporal arrangements of the data-set.

III. Experiments

A. Performance of Permutation Decision Tree Relative to Entropy-Based Decision Tree

The permutation we used for this evaluation was the 'Permutation With Causes Followed by Effect' in Project Report-2.

The data was arranged such that the first 200 data instances are from the *cause* class and the next 200 data instances are from the *effect* class. This again gives us a total data-set size of 400 rows and 10 columns.

We then made 10 splits of this data-set using the TimeSeriesSplit and evaluated the performance of our model using SkLearn's *accuracy score* which gives us the percentage of correctly predicted data instances.

B. Performance of Five Random Permutations on Permutation Decision Tree

The permutation we used for this evaluation was generated using the procedure mentioned under 'Methodology', Section D.

The data-set contains 400 rows and 10 columns.

We then made 10 splits of this data-set using the TimeSeriesSplit and evaluated the performance of our model using SkLearn's *accuracy score* which gives us the percentage of correctly predicted data instances.

C. Comparison of Performance of PDT and DT on Wisconsin Breast Cancer Diagnosis Data-set

From the UC Irwin Machine Learning Repository, we were able to get access to this data-set. The data-set contains 569 instances each containing 30 features. Each instance is either classified as benign or malignant. The composition of the data-set is 63 percent benign and 37 percent malignant.

We split the data into a training and testing data-set in the ratio of 80:20. With a slight modification to the NSRPS algorithm, we were able to re-use the same implementation. There was no modification done to the ordering of the data instances. This can be done if the motivation is to generate a pool different Decision Trees for a domain expert to choose from.

IV. Observations

In the case of using the data-set generated with different Auto-Regressive coefficients for each feature and using ten-fold cross-validation, we get the following accuracy values:

Split Number	Class-0 Freq	Class-1 Freq	PDT Accuracy	DT Accuracy
1	36	0	100.0	100.0
2	36	0	100.0	100.0
3	36	0	100.0	100.0
4	36	0	100.0	100.0
5	16	20	44.44	44.44
6	0	36	22.22	22.22
7	0	36	33.33	61.11
8	0	36	33.33	47.22
9	0	36	41.67	52.78
10	0	36	72.22	68.44

In the case of using the data-set generated with the same Auto-Regressive coefficients for each feature and using ten-fold cross-validation, we get the following accuracy values:

Split Number	Class-0 Freq	Class-1 Freq	PDT Accuracy	DT Accuracy
1	36	0	100.0	100.0
2	36	0	100.0	100.0
3	36	0	100.0	100.0
4	36	0	100.0	100.0
5	16	20	44.44	44.44
6	0	36	30.56	13.88
7	0	36	8.33	16.66
8	0	36	8.33	11.11
9	0	36	66.67	55.55
10	0	36	44.45	47.22

In the case of using the data-set generated with the same Auto-Regressive coefficients for each feature and using five different permutations as described in the Methodology, and using ten-fold cross-validation, we get the following average accuracy values for each permutation:

Permutation	Average PDT Accuracy	Average DT Accuracy
A	57.78	61.66
B	53.33	63.33
C	58.06	65.56
D	52.49	73.61
E	53.36	61.94

A. Wisconsin Breast Cancer Diagnosis Data-set

The entropy based Decision Tree achieved an accuracy of 92.10 while the ETC based Permutation Decision Tree was able to achieve an accuracy of 96.49.

V. Conclusion

As observed, the performance of the Permutation Decision Tree is slightly worse-off, if not par with the Entropy-based Decision Tree, in the case of using our own custom generated data-set.

We believe the Permutation Decision Tree is applicable in cases where there is scope for multiple interpretations and the most appropriate Decision Tree can be chosen by the domain expert. In this scenario, our Permutation Decision Tree ends up performing better than the traditional Decision Tree.

References

[1] Harikrishnan, N.B., Nithin Nagaraj (2023). Permutation Decision Tree. arXiv:2306.02617v2.