IN3062

Introduction to AI

IN3062 Coursework : Tree coverage prediction

Jakub Zavacky
Department of computer Science, City, University of London

This document is intended to demonstrate understanding of artificial intelligence methods and data preparation. This coursework is based on learnings from Introduction to artificial intelligence module. Dataset was chosen for it's complexity and personal preference. Question asked in this paper is "What type of tree cover will be on certain combination of wildlife, soil and elevation and lighting aspects?"

I. **Introduction**

This paper contains my general understanding of artificial intelligence methods. The dataset chosen is Tree coverage dataset to be found on Kaggle website on the following link https://www.kaggle.com/uciml/forest-cover-type-dataset
The main focus in this work is to predict what type of tree is present based on elevation, soil type and many more aspects. This work is written in python and will use following artificial intelligence and data manipulation libraries: TensorFlow, Numpy, Pandas, sklearn

This work is based on learning gained from lectures in module IN6062, and following sources. The youtube channel "Programming with Mosh", especially his video https://www.youtube.com/watch?v=7eh4d6sabA0 , the deeplizard website and their tensorFlow tutorials, particularly https://deeplizard.com/learn/playlist/PLZbbT5o_s2xrwRnXk_yCPtnqqo4_u2YGL
We are going to explore 3 algorithms in this paper. Those are Decision tree from sklearn library, Linear regression from sklearn library and Neutral Network from tensorflow library.

II. Description of Dataset

The study area includes four wilderness areas located in the Roosevelt National Forest of northern Colorado. Each observation is a 30m x 30m patch. You are asked to predict an integer classification for the forest cover type. The seven types are:

1 - Spruce/Fir
2 - Lodgepole Pine
3 - Ponderosa Pine
4 - Cottonwood/Willow
5 – Aspen
6 - Douglas-fir
7 - Krummhol

The dataset set (cca 500 000 observations) contains both features and the Cover_Type.
Data Fields:
Elevation - Elevation in meters
Aspect - Aspect in degrees azimuth
Slope - Slope in degrees
Horizontal_Distance_To_Hydrology - Horz Dist to nearest surface water features
Vertical_Distance_To_Hydrology - Vert Dist to nearest surface water features
Horizontal_Distance_To_Roadways - Horz Dist to nearest roadway
Hillshade_9am (0 to 255 index) - Hillshade index at 9am, summer solstice
Hillshade_Noon (0 to 255 index) - Hillshade index at noon, summer solstice
Hillshade_3pm (0 to 255 index) - Hillshade index at 3pm, summer solstice
Horizontal_Distance_To_Fire_Points - Horz Dist to nearest wildfire ignition points

Wilderness_Area (4 binary columns, 0 = absence or 1 = presence) - Wilderness area designation
Soil_Type (40 binary columns, 0 = absence or 1 = presence) - Soil Type designation
Cover_Type (7 types, integers 1 to 7) - Forest Cover Type designation
Wilderness_areas and Soil_type descriptions are found in Appendix A.

III. Method

Approach to this challenge is going to be by supervised learning. The problem is classification as it predicts what kind of entity is present in certain circumstances. Methods has been selected to show which algorithms perform well on dataset and which are not.

Import data

Data has been imported by Pandas' read_csv function. In case we would have really big dataset we can use chunksize parameter to load part of csv file. Some datasets I was training on also came with separated csv files for training and test data. In our case we had all data in one csv file.

Cleaning the Data

Data has been checked if they have present any empty fields with .isnull().any() function, and as it returned False for all columns, there is no need to substitute any values or remove any rows in the dataset. If that would be case, I would have to remove the rows with NaN values in Soil or wilderness columns, if the NaN values would be in the other columns I could substitute them with mean or median value in regards to wilderness area the data are missing at.

Split Data into training and testing data

I have separated Cover_Type column from dataset to create features dataframe and target dataframe. Those dataframes I fed to sklearn library that has convenient function train_test_split, that splits data in training samples and testing samples. I am using random_state=0 parameter to reproduce the same data samples so I can reasonably compare my models with other methods in the future, and reproduce the same results when rerunning the algorithm

Create model

I am using decision tree model and linear regression model from sklearn library and Neuron networks from tensorflow library. Creating model is fairly simple with sklearn and tensorflow library – you are just creating object. In neuron networks you can simply add layers as needed.

Train model

Training model is again fairly simple as we just call .fit method on our model object and provide corresponding training features data and training .target data.

Make predictions

Predictions are based on testing data. This data has only feature values and are new to our model (model never seen this data before). Model makes prediction based on what we showed him during training phase and classify the test data accordingly to its best knowledge.

Evaluate and improve

Decision tree is evaluated as a accuracy score based on how many test data he recognised correctly. For linear regression mean error and square root error is used to classify model correctness (Lower value of square root error and mean error, the better model is – 0 means that our model is perfect). We can observe and try to improve result by adjusting ratio between training and testing data in 2 formerly mentioned models. Regarding neuron networks we can adjust not just size of training and testing set, but also number of layers, what activation function use and how many nodes each layer will have. During this phase we can also compare different models and decide which gives best results for given dataset

IV. Experiment

Biases

The dataset is from natural environment, therefore we can expect some anomalies to happen (random "rogue" tree in different

forest type), but we generally expect that same species of trees would prefer the same condition as their parents and their parenting methods. Refer to book The hidden life of trees for more. Another fact we need to understand is that when forest type is changing it's not a straight line, they intersect a bit and that may confuse our model

Work with dataset

Data has been loaded at once as they fit on the memory. After that dataset was split in feature and target and divided into training and testing data. Data has been checked for empty entries during that process as well. Regarding neuron networks we had to conver our data into tensorflow's dataframe format Decision tree model algorithm has been applied to dataset that was split accordingly as described above. Then I run a function that confirmed that split was reasonable and end up with 10% being testing data and 90% being training data. But to prevent just "memorazing" I have decided to go with 20% test sample and 80% training sample, which given satisfactory results. Decision tree model predicted most of the trees correctly and do not take long to execute. The more training data he gets, the better he performs.

After finishing with Decision Tree model, Linear regression model has been applied on accordingly split data and proper division of training data and split data has been found at ratio: 14 – 18% testing data and 86 – 82% training data. As predicted, regression model performed way worse than Decision tree model, as it tries to return continuous value (coefficients of the regression line).

Last method, neuron networks was most efficient in predicting correct tree cover of all. I have used result from previous experiments and split data into 25% for testing and 75% for training. I have set up 4 models, which all have 7 nodes output layer: One layer with sigmoid function (OLS), one layer with relu function(OLR), two layer with sigmoid function(TLS), two layer with relu function(TLR). I run optimalizing for number of cores in layers this time, but more experiencing is needed on number of epoch and run it on more randomize data, different combination of activation functions or adding another ones like softmax and also different splits to determine the best model. As our model return quite good result already, I skipped this analysis due to time needed to run those analysis. I would probably try to play more with two layered relu network as it gives bst result so far. All experiments were run on google collab with use of google's CPU and RAM. The link to this experimental notebook is here:
https://colab.research.google.com/drive/1NhyWhTXXufcPzz_NN0g-3GFhmzOw1On1?usp=sharing

Decision tree and linear regression models are noticeable faster to train in comparison to Neuron network.

V. Results

Experiment with determination tree algorithm proved to be fairly effective with accuracy about 93% even for smaller training samples. Regarding how branching in decision trees work this result indicates that there might be strong correlation between what tree cover we can find on certain soil types, elevation and wilderness . Part of the error in predicting is most likely where type of tree cover is changing and that there are sometimes some "rogue" tree here and there.

Experiment with linear regression algorithm was not that successful as predicted, with root mean square error hardly get close to value of 2, which is very high and is unlikely go under markdown 2. This i(RMSE)s as this model tries to fit line over very complex and high dimensional datapoints. The data are to discreet for this approach.

Experiment with Neuron networks worked best of all with 4 different models predicting tree coverage very precisely with RMSE no higher as 0,3 and with as low as 0.226.

Models performed as follow:

OLS: number of nodes: 128
: RMSE: 0,258

OLR: number of nodes: 544
: RMSE:0,249

TLS: number of nodes: 64
: RMSE: 0,259
TLR: number of nodes: 480
: RMSE:0,226

VI. Conclusion

The determination tree and neuron network results were promising and I would use them for further work on dataset if I have some industrial use of those data. If project would be tight on time but we would be satisfied with lower accuracy I would go with Decision tree model, otherwise the Neuron networks seems to have higher potential on this dataset as we can play a lot with settings of the number of nodes what activations functions are used in many different type of layers. There is also option that we can use someone's else general model and make its parameters untrainable and add some of our layers and train just those on our dataset to get better results of those models for our data in the future. Let's say we would have model for mixed forest but we would like to identify our pine forest. We would use model for mixed forest, add some layers and retrain it with data from pine forest and we would get our pine forest model faster and more precise. The linear regression is not a correct way how to train our model, but it was nice to have insight how to set it up and see what kind of results it's getting if applied on wrong type of problem. I would use it for different purpose.

VII. Further research

Further research will be on how to use satellite and aerial observatory data to identify forests, their types and its health (and all changes related to these, let's say if forest got a parasite or is overheating), what are the conditions if we want some kind of forest to thrive. Those observation and ground data will be fed into the model and used to predict if forest is getting sick and need some help by human interference (Not mentioning to reduce our pollution footprint). It might also track municipalities greenery and their health and needs. It might help us to analyse reasons for deforestation of our rainforests.

APENDIX A

The wilderness areas are:

1 - Rawah Wilderness Area
2 - Neota Wilderness Area
3 - Comanche Peak Wilderness Area
4 - Cache la Poudre Wilderness Area

The soil types are:

1 Cathedral family - Rock outcrop complex, extremely stony.
2 Vanet - Ratake families complex, very stony.
3 Haploborolis - Rock outcrop complex, rubbly.
4 Ratake family - Rock outcrop complex, rubbly.
5 Vanet family - Rock outcrop complex complex, rubbly.
6 Vanet - Wetmore families - Rock outcrop complex, stony.
7 Gothic family.
8 Supervisor - Limber families complex.
9 Troutville family, very stony.
10 Bullwark - Catamount families - Rock outcrop complex, rubbly.
11 Bullwark - Catamount families - Rock land complex, rubbly.
12 Legault family - Rock land complex, stony.
13 Catamount family - Rock land - Bullwark family complex, rubbly.
14 Pachic Argiborolis - Aquolis complex.
15 unspecified in the USFS Soil and ELU Survey.
16 Cryaquolis - Cryoborolis complex.
17 Gateview family - Cryaquolis complex.
18 Rogert family, very stony.
19 Typic Cryaquolis - Borohemists complex.
20 Typic Cryaquepts - Typic Cryaquolls complex.
21 Typic Cryaquolls - Leighcan family, till substratum complex.
22 Leighcan family, till substratum, extremely bouldery.
23 Leighcan family, till substratum - Typic Cryaquolls complex.
24 Leighcan family, extremely stony.
25 Leighcan family, warm, extremely stony.
26 Granile - Catamount families complex, very stony.
27 Leighcan family, warm - Rock outcrop complex, extremely stony.
28 Leighcan family - Rock outcrop complex, extremely stony.
29 Como - Legault families complex, extremely stony.
30 Como family - Rock land - Legault family complex, extremely stony.
31 Leighcan - Catamount families complex, extremely stony.
32 Catamount family - Rock outcrop - Leighcan family complex, extremely stony.
33 Leighcan - Catamount families - Rock outcrop complex, extremely stony.
34 Cryorthents - Rock land complex, extremely stony.
35 Cryumbrepts - Rock outcrop - Cryaquepts complex.
36 Bross family - Rock land - Cryumbrepts complex, extremely stony.
37 Rock outcrop - Cryumbrepts - Cryorthents complex, extremely stony.
38 Leighcan - Moran families - Cryaquolls complex, extremely stony.
39 Moran family - Cryorthents - Leighcan family complex, extremely stony.
40 Moran family - Cryorthents - Rock land complex, extremely stony.