Machine Learning Notes (Random Forest)

Deep Learning (Neural Networks) are much better to deal with most problems which all these algorithms can do.

Libraries – sklearn (scikit), IPython

Pandas and Numpy work well with each other.

Different Machine Learning Algorithms -

<https://scikit-learn.org/stable/supervised_learning.html>

<https://towardsdatascience.com/do-you-know-how-to-choose-the-right-machine-learning-algorithm-among-7-different-types-295d0b0c7f60>

Prediction of Auction price for Bulldozers using Random Forests is the exercise.

Curse of Dimensionality and No Free Lunch theorem do not generally apply in real world.

* <https://medium.com/@hiromi_suenaga/machine-learning-1-lesson-1-84a1dc2b5236>

RandomForestRegressor — regressor is a method for predicting continuous variables (i.e. regression)

RandomForestClassifier — classifier is a method for predicting categorical variables (i.e. classification)

For Notebooks – There is a bit of work which needs to be done before it is usable.

!pip install fastai==0.7.0

!pip3 install torch torchvision

!pip install "torchtext==0.3.1"

!pip3 install spacy

!python -m spacy download en

from fastai.imports import \*

from fastai.structured import \*

from torch import \*

from pandas\_summary import DataFrameSummary

from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier

from IPython.display import display

from sklearn import metrics

Data will also have to be uploaded and will vanish once the Colab is disconnected.

For Lesson 1 – train.csv was uploaded after creating necessary folders on the colab machine.

Alternate solution for data is to use google drive and using that path in the program or downloading from URL (Kaggle or Google Drive). Use whichever works and is the fastest/easiest.

Random Forest Exercise Summary

Refer the Jupyter notebook for details - <https://colab.research.google.com/drive/1ARrxUcm8cJdQ0J7sONa5FfQuan9nZSDH#scrollTo=6jeziZfhnA2x>

* Take an input csv. (CSV can be populated by any means with whatever relevant data is needed for your requirement)
* The goal is to predict a measure (in this case Auction Price)
* Look at the data to understand what it has type of columns (String, number, date...), NULL values, blank values etc.
* Read the csv data into a Pandas Dataframe. While reading there are options to specify if there are any date columns. Specify that and Pandas will read those as Dates rather than strings.
* Choose a Metric to evaluate the prediction vs actual.

It's important to note what metric is being used for a project. Generally, selecting the metric(s) is an important part of the project setup. However, in this case Kaggle tells us what metric to use: RMSLE (root mean squared log error) to compare between the actual and predicted auction prices. Therefore, we take the log of the prices and then RMSE on top of this will give RMSLE.

* The data contains continuous and categorical values.
* Any Date columns needs to be converted to Categoricals. If it is continuous (like how it is in the file i.e. just a date column), then we cannot capture trend and will not be useful for prediction. Below function does that and should be defined in your program and used. It is also present in fastai’s old library. But it converts to String Categoricals.



* Since the above functions converts the date into String Categoricals, we still need to convert it into Panda categoricals. Panda categoricals are like key value pair (0 🡪 value1 , 1 🡪 value2 and so on) for each of the values. This is needed for efficiency of the Random Forest. Below function is used for that purpose.



* Both these functions change the dataframe in-place. The function will need to be changed if it has to write to a different dataframe.
* Dataframe now has Category Objects in it for all non-numerical columns (each non-numerical column we saw in csv is a category object now). There are functions which can be run on Category objects in Python. Cat is one such function and it further has child functions.
* We now check for blank or missing values for all category objects. Below can give the percentage of such values in each of them. We will deal with them in a while.

display (df\_raw.isnull().sum().sort\_index()/len(df\_raw))

* We can save the dataframe at such a point in a feather format. Saves to disc in the same format as it was in RAM.
* We can load it back from disc to a dataframe again later.
* Then we process the dataframe (proc\_df) to fix missing values and more importantly makes a copy of dataframe, drops the dependent value (in this case Auction Price object) in this copy dataframe and converts categorical data into their respective codes It also adds 1 to codes to convert Null categories which will have code as -1 to 0 and this also changes the other codes to 1,2,3 and so on. Below is the function which can help with that. This function further uses other child functions mentioned below.





* The original dataframe is used to get the dependent value into a “y” variable.

**The above output is in a format that can be passed into Random Forest Regressor.**

In statistics, the coefficient of determination, denoted R2 or r2 and pronounced "R squared", is the proportion of the variance in the dependent variable that is predictable from the independent variable(s). <https://en.wikipedia.org/wiki/Coefficient_of_determination>

m = RandomForestRegressor(n\_jobs=-1)

m.fit(df, y)

m.score(df,y)

However, this could be an overfit since we have used all the data from train.csv

We need to verify this using a validation set. There are various hyperparameters that need to be set and various methods to be employed to get balances Random forest which has a balance between variance (highly overfit for Training data) and bias (highly biased even with Training data)

Good read - <https://towardsdatascience.com/an-implementation-and-explanation-of-the-random-forest-in-python-77bf308a9b76>

Different options for Random Forest Regressor.

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html>