Data Science with Matlab

Matlab offers functionalities to read data, compute statistics and make predictions. One source of public available sample data is the machine learning repository at UC Irvine:

https://archive.ics.uci.edu/ml/index.php

In this example we download a data set of car fuel consumption at

https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/

The file auto-mpg contains the following columns of data:

mpg, cylinders, displacement, horsepower, weight, acceleration, year, origin, car name

18.0 8 307.0 130.0 3504. 12.0 70 1 "chevrolet chevelle malibu"

15.0 8 350.0 165.0 3693. 11.5 70 1 "buick skylark 320"

18.0 8 318.0 150.0 3436. 11.0 70 1 "plymouth satellite"

16.0 8 304.0 150.0 3433. 12.0 70 1 "amc rebel sst"

17.0 8 302.0 140.0 3449. 10.5 70 1 "ford torino"

15.0 8 429.0 198.0 4341. 10.0 70 1 "ford galaxie 500"

Matlab provides functions like "csvread" and "readtable" to read those kind of data.

Unfortunately the data file is not in a nice format as it contains a mix of numbers and strings and does not use a "," delimiter between columns.

To read the file, the one can use first the 'Import Data' tool from Matlab's menu to help parsing the format and creating the format specification string below, where for example '%4s' indicates that a string of 4 letters is expected. This requires some experimentation, other useful format specifieres are %f for float, %d for integer, %s for string, %c for a single character.

First the data file is read into a cell-array called dataArray:

```
filename = 'C:\Users\a_rho\Downloads\auto-mpg.data';
formatSpec = '%4s%4s%8s%11s%11s%10s%5s%4s%s%[^\n\r]';
fileID = fopen(filename,'r');
dataArray = textscan(fileID, formatSpec, 'Delimiter', '', 'WhiteSpace', '', 'TextType', 'string
dataArray{9}=strrep(dataArray{9},'"',"");
fclose(fileID);
```

The dataArray table contains arrays of strings instead of numbers. The following code converts strings into numbers and assigns the colums to the respective array variables.

```
mpg = double(strtrim(dataArray{1}));
cylinders = double(strtrim(dataArray{2}));
displacement = double(strtrim(dataArray{3}));
```

The function strtrim() trims a string, it eleminates surounding space. For example

```
strtrim("   ABC   ")
```

```
ans =
"ABC"
```

The function strrep replaces characters with others, for example

```
strrep("CITUTECH","U","Y")
```

```
ans =
"CITYTECH"
```

The horsepower array has some "?" marks indicating missing values. To proceed futher we replace them with zeros.

```
horsepower = double(strtrim(strrep((dataArray{4}),"?","0")));
```

Continuing with the other arrays

```
weight = double(strtrim(dataArray{5}));
acceleration = double(strtrim(dataArray{6}));
year = double(strtrim(dataArray{7}));
origin = double(strtrim(dataArray{8}));
name = strtrim(dataArray{9});
```

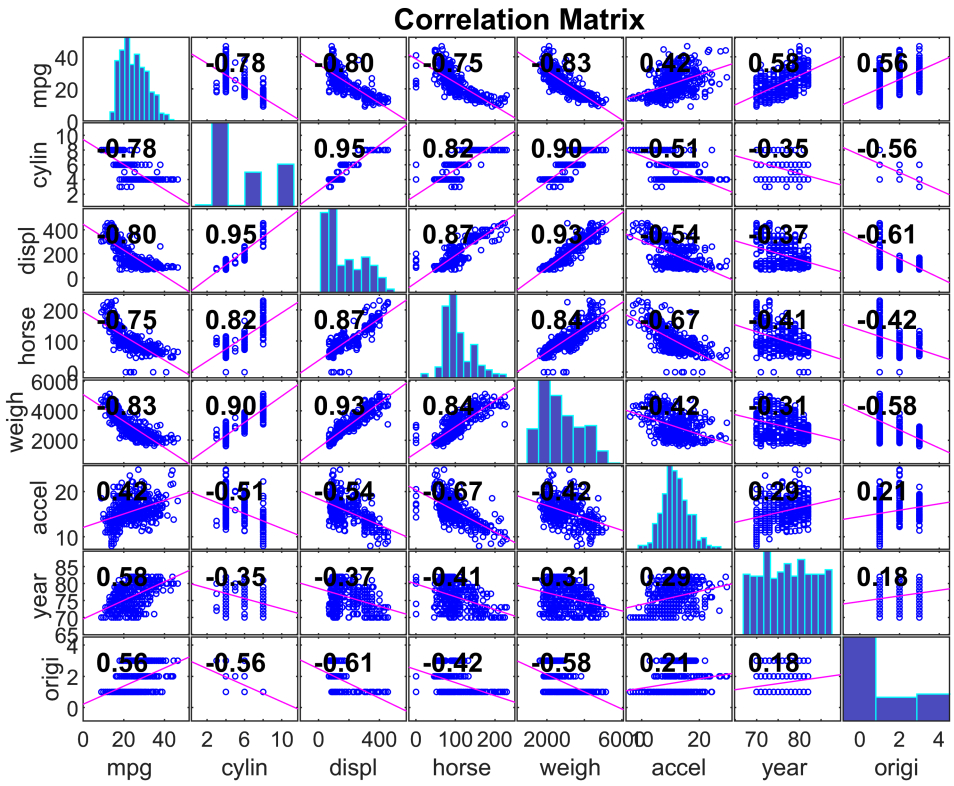
Let's check how many entries are zero in the horsepower column

```
notmissing = horsepower>0;
missing = horsepower==0;
sum(notmissing)
```

```
ans = 392
```

To get a sense of the data we plot the correlation of each data column with the others. The economics toolbox offers the corrplot function

```
corrplot([mpg,cylinders,displacement, horsepower, weight,acceleration,year,origin], ...
    'varNames',{'mpg','cylinders','displacement','horsepower','weight','acceleration','year','o
```



. The goal of this exercise is to see if we can predict the milage (mpg) given the car's other characteristics. From the plot above it seems the columns displacement, horsepower and weight correlate well with mpg. Thus the plan is to make a model that predicts mpg given those three columns.

Missing values

First we need to take care of the missing horsepower values, since we want to use the horsepower column. There are several approaches how to deal with missing data.

1. discard the missing rows all together
2. replace the missing entry with an average value
3. impute the missing entry: predict it by modelling it using the remaining columns

Let's try strategy 3. Looking at the correlation graph above we choose ,displacement, weight and acceleration to model the displacement. We build a 'RegressionTree' to model the non-missing data.

```
x= [displacement,weight,acceleration]; % known data features
y = horsepower; % target to be predicted
tree_horsepower = RegressionTree.fit(x(notmissing,:), y(notmissing))
```

```
tree_horsepower =
  RegressionTree
             ResponseName: 'Y'
    CategoricalPredictors: []
        ResponseTransform: 'none'
          NumObservations: 392
```

```
  Properties, Methods
```

Using the model we then predict the missing horsepower entries

```
horsepower(missing)=tree_horsepower.predict(x(missing,:));
horsepower(missing)
```

```
ans = 6×1
   67.0000
   84.7500
   47.5000
  107.3750
   93.3333
   90.0000
```

Now we are ready to build the model for mpg, given cylinders, displacement, horsepower and weight. For cylinders we specify it as a categorical variable, telling the model to ignor its numerical value. Thus the model will not use that 4 cylinders is more than 3 cylinders, instead it will just treat each cylinder count as a different categotical label without regard to its numerical value.

For the purpose to get a small tree that we can plot for illustration purposes we set below the MaxNumSplits is set to 5 only.

```
x= [cylinders, displacement,horsepower, weight];
y = mpg;
tree_mpg =  RegressionTree.fit(x, y, ...
    'MaxNumSplits', 5, 'CategoricalPredictors',[1], ...
    'PredictorNames',{'cylinders','displacement','horsepower','weight'}, ...
    'ResponseName','mpg')
```

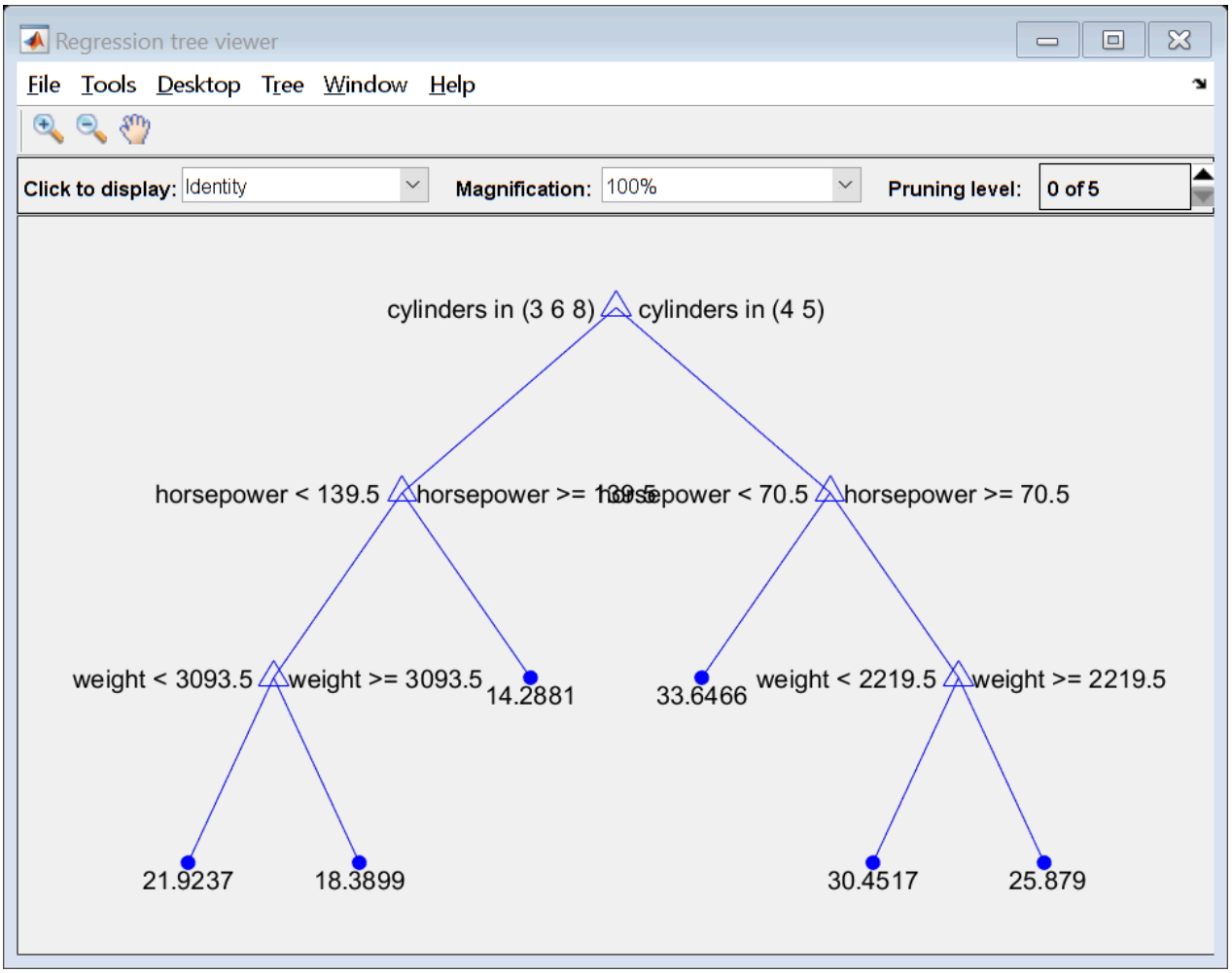
```
tree_mpg =
  RegressionTree
           PredictorNames: {'cylinders'  'displacement'  'horsepower'  'weight'}
             ResponseName: 'mpg'
    CategoricalPredictors: 1
        ResponseTransform: 'none'
          NumObservations: 398


  Properties, Methods
```

One way to assess the quality of the model is to compute the cross-validation loss, the error made in prediction in case one splits the data set in training and testing portions. The training part is used to fit the model while the test portion is used to evaluate the quality of the prediction. Doing so varifies if the model generalizes to data that was not used in training.

```
cvloss(tree_mpg)
```

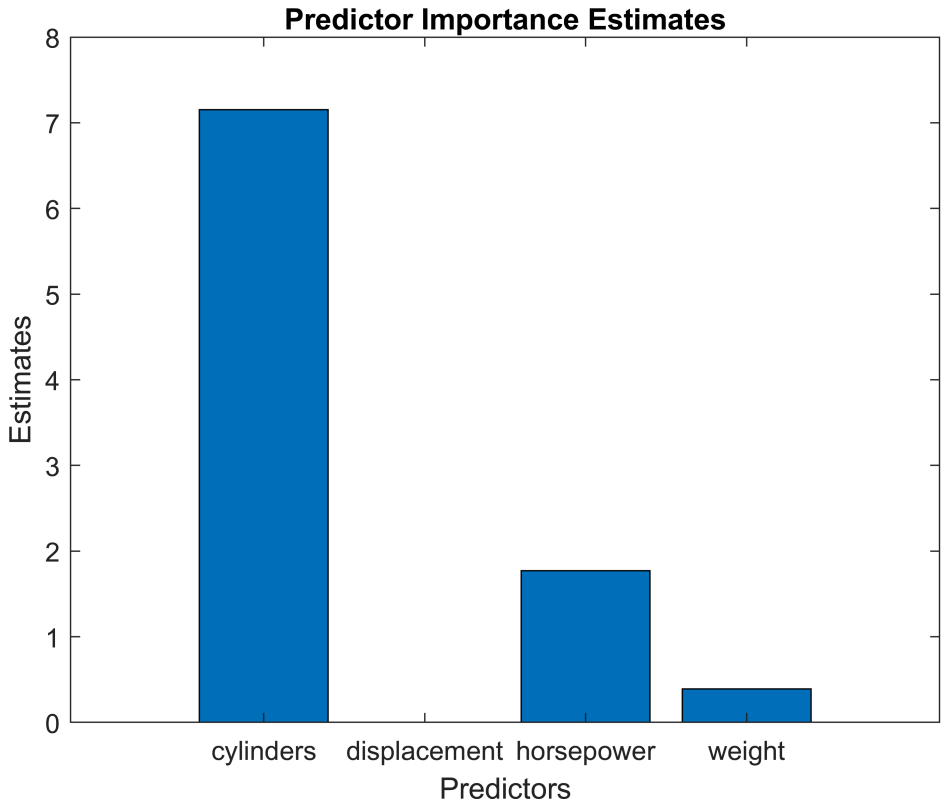
```
ans = 19.2635
```

```
view(tree_mpg,'Mode','graph')
```



The following code uses the model to show which feature has the most predictability for mpg.

```
imp = predictorImportance(tree_mpg);
figure;
bar(imp);
title('Predictor Importance Estimates');
ylabel('Estimates');
xlabel('Predictors');
ax = gca;
ax.XTickLabel = tree_mpg.PredictorNames;
```



Matlab offers a set of model functions that can be used in a similar way to predict either a continues variable (regression) or a discrete label (classification).

Task:

Use the automated interface to build a regression model by typing "regressionLearner" into Matlab's command line and import the data matrix D from the workspace (New Session / as define below.

(note there is also `classificationLearner for predicting binary labels`)

```
D = [mpg,cylinders,displacement, horsepower, weight,acceleration,year,origin];
```

The regression learner allows to train and compare automatically all available models. Run all models to predict mpg. What models gives the lowest RMSE error?