Basic theory and construction of naive Bayesian classifiers

Anton Antonov
Mathematica for Prediction blog
Mathematica for Prediction project at GitHub
October 2013

Introduction

In this document we consider the following classification problem: from a given two dimensional array representing a list of observations and labels associated with them predict the label of new, unseen observation.

Consider for example this sample of a data array with country data:

Name	PopulationGrowth	LifeExpectancy	MedianAge	LiteracyFraction	BirthRateFraction	DeathRateFraction	MigrationRateFraction	GPD per capita class label
Cape Verde	0.0141443	71.61	21.1	0.766	0.0235	0.00622	-0.01167	low
Greenland	-0.0000348985	70.07	33.5	1.	0.01476	0.00814	-0.00599	high
Guam	0.01365	78.01	29.1	0.99	0.01822	0.00457	0.	low
Guinea-Bissau	0.0223265	47.9	19.3	0.424	0.03597	0.01579	0.	low
Hong Kong	0.00495751	81.86	42.3	0.935	0.00742	0.00676	0.00438	high
Ireland	0.0188522	78.24	35.	0.99	0.01423	0.00775	0.00471	high
Lithuania	-0.010479	74.9	39.3	0.996	0.00911	0.01118	-0.00072	low
Spain	0.00987781	80.05	41.1	0.979	0.00972	0.00999	0.00099	high
Tanzania	0.0291918	52.01	18.	0.694	0.03429	0.01259	-0.0013	low
Turkmenistan	0.0133066	67.87	24.4	0.988	0.01969	0.00631	-0.00197	low

We assume that have the following observed variables.

variable index	variable name
1	`PopulationGrowth`
2	`LifeExpectancy`
3	`MedianAge`
4	`LiteracyFraction`
5	`BirthRateFraction`
6	`DeathRateFraction`
7	`MigrationRateFraction`

The predicated variable is “GDB per capita”, the last column with the labels “high” and “low”.

Note that the values of the predicted variable “high” and “low” can be replaced with `True` and `False` respectively.

One way to solve this classification problem is to construct a Naive Bayesian Classifier (NBC), and then apply NBC to new, unknown records comprised by the observed variables.

NBC despite of its name is a very competitive tool for solving classification problems. The "naive" part of the name comes from the assumption that the observed variables (the variables on which the classification should be based on) are independent. Obviously, this is rarely true, but if a sufficient level of independence holds, then NBC can be applied with success.

The reasons we consider NBC are that (1) its implementation is very easy and (2) its performance is competitive with other more sophisticated classifiers.

This document provides basic theory for NBC and is also can serve as guide of using the implementations provided by [1]. A short introduction to NBC is given by [2].

General description

Let $dom(X)$ denote the domain of the variable X. (If $X \in \mathbb{R}$ then $\mathrm{dom}(X) = \mathbb{R}$.) Let $D_i := \mathrm{dom}(X_i)$, where X_i, $i \in [1, \ldots, k]$, $k \in \mathbb{N}$, correspond to the variables (the columns) of the given data array. Given $x \in D_1 \times \ldots \times D_k$ with x_i we denote the i-th coordinate of x.

In this document we assume that only two labels are used, True and False.

We define NBC as a function with domain and codomain:

$$D_1 \times D_2 \times \ldots \times D_k \to \{\text{True}, \text{False}\}. \tag{1}$$

For each value $c \in \{\text{True}, \text{False}\}$ of the predicted variable, NBC has a function of the form

$$B_c(x) := S_1^c(x_1)\, S_2^c(x_2) \ldots S_k^c(x_k), \tag{2}$$

where the functions S_i^c are piecewise constant functions with codomain $[0, 1] \subset \mathbb{R}$.

$S_i^c(y)$, $y \in D_i$ gives the probability for the predicted variable to be c when $X_i = y$. In other words

$$S_i(y) := P(c \,/\, X_i = y).$$

Since the variable we want to predict has two values, True and False, the NBC we consider has two corresponding functions B_t and B_f. The classification function of the considered NBC is

$$\mathrm{NBC}(\theta, \phi, x) := \begin{cases} \text{True} & B_t(x) \geq \theta \bigvee 1 - B_f(x) \geq \phi \\ \text{False} & B_f(x) > 0.5 \\ B_t(x) > B_f(x) & \text{otherwise} \end{cases}. \tag{3}$$

The parameters $\theta, \phi \in \mathbb{R}$ are determined by experimentation.

The classifier implementation

The most fundamental part of a real life NBC is the implementation of the piecewise constant functions S_i from (2).

If X_i is a numerical variable we can specify S_i with a list of $n \in \mathbb{N}$ values $\{v_1, \ldots, v_n\}$ from D_i and a list of $n-1$ real values $\{p_1, \ldots, p_{n-1}\}$, $p_j \in [0, 1]$, $1 \leq j \leq n-1$. With these lists we can compute the function

$$S_i(y) := \begin{cases} p_1 & v_1 \leq y < v_2 \\ p_2 & v_2 \leq y < v_3 \\ \ldots & \ldots \\ p_{n-1} & v_{n-1} \leq y < v_n \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

If X_i is a categorical variable, we can specify S_i with a list of $n \in \mathbb{N}$ values $\{v_1, \ldots, v_n\}$ from D_i and a list of n real values $\{p_1, \ldots, p_n\}$, $p_j \in [0, 1]$, $1 \leq j \leq n$. With these lists we can compute the function

$$S_i(y) := \begin{cases} p_1 & y = v_1 \\ p_2 & y = v_2 \\ \ldots & \ldots \\ p_n & y = v_n \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

The implementation of the functions S_i should be generic, it should allow the computations to be done with different lists of values and probabilities for each S_i.

The implementation of formula (3) is trivial. The NBC we consider is fully implemented with the implementations corresponding to (2), (3), (4), and (5).

Determining the classifier functions

In order to find the lists that specify S_i in (4) and (5) the Bayes formula is used:

$$S_i(y) := P(c / X_i = y) = \frac{P(c \bigcap X_i = y)}{P(X_i = y)} = \frac{P(c)\, P(c \bigcap X_i = y)}{P(c)\, P(X_i = y)} = \frac{P(c)\, P(X_i = y / c)}{P(X_i = y)}. \tag{6}$$

This formula is approximated with bin counts over the data.

Example of usage

Using *Mathematica*'s function `CountryData` we can make a data array with the observation variable columns

variable index	variable name
1	PopulationGrowth
2	LifeExpectancy
3	MedianAge
4	LiteracyFraction
5	BirthRateFraction
6	DeathRateFraction
7	MigrationRateFraction

and a label column “GDP per capita”. The label “high” is assigned to countries which have GDP per capita greater than $30 000; the label “low” is assigned to the rest of the countries. (A sample of this data was shown in the introduction.)

In order to demonstrate the usage of NBC we are going to split the data array into training and testing sets and apply the NBC generation and classification functions of the package [1]. The NBC generation is done over the training set. The NBC classification is done over the test set without the label column and we can compare the predicted by the classification labels with the labels of the test set.

Data array construction -- demographic and GDP data

We have 240 countries.

```
In[1]:= countries = CountryData["Countries"];
        countries // Length
Out[2]= 240
```

We query `CountryData` for the desired variables. We also take “Population” and “GDP” in order to calculate “GDP per capita”.

```
In[3]:= propNames =
          {"Name", "PopulationGrowth", "LifeExpectancy", "MedianAge",
           "LiteracyFraction", "BirthRateFraction", "DeathRateFraction",
           "MigrationRateFraction", "Population", "GDP"};
        cdata = Map[Table[CountryData[#, p], {p, propNames}] &, countries];
        cdata // Length
Out[5]= 240
```

We filter out the countries with missing data.

```
In[6]:= cdata = Select[cdata, VectorQ[Rest[#], NumberQ] &];
        cdata // Length
Out[7]= 216
```

We replace the last two columns, “Population” and “GDP”, with a label according to their ratio.

In[8]:=
```
cdataLabeled =
  Map[Append[#⟦1 ;; -3⟧, If[#⟦-1⟧ / #⟦-2⟧ > 30 000, "high", "low"]] &,
   cdata];
```

Here is breakdown of the countries according to the assigned labels:

In[9]:=
```
Tally[cdataLabeled⟦All, -1⟧]
```

Out[9]=
```
{{low, 176}, {high, 40}}
```

Here is a sample of the data:

In[10]:=
```
gridInds = RandomSample[Range[1, Length[cdataLabeled]], 20];
gridData = cdataLabeled⟦gridInds⟧;
Magnify[#, 0.6] &@Grid[Prepend[SortBy[#, #⟦1⟧ &] &@gridData,
   Style[#, Blue, FontFamily → "Times"] & /@ Join[propNames⟦1 ;; -3⟧,
     {"GPD per capita\nclass label"}]], Alignment → Left]
```

Out[12]=

Name	PopulationGrowth	LifeExpectancy	MedianAge	LiteracyFraction	BirthRateFraction	DeathRateFraction	MigrationRateFraction	GPD per capita class label
Bosnia and Herzegovina	-0.00140535	78.5	39.8	0.967	0.00885	0.00863	0.00317	low
Equatorial Guinea	0.0264508	61.61	18.9	0.87	0.03652	0.00949	0.	low
France	0.00537242	80.98	39.4	0.99	0.01257	0.00856	0.00148	high
French Polynesia	0.0128463	76.71	29.1	0.98	0.01591	0.00473	0.00273	low
Hong Kong	0.00495751	81.86	42.3	0.935	0.00742	0.00676	0.00438	high
Madagascar	0.0272289	62.89	18.	0.689	0.03814	0.00814	0.	low
Maldives	0.014329	73.97	25.7	0.963	0.01455	0.00365	-0.01258	low
Mali	0.0239275	50.35	15.8	0.464	0.04915	0.01582	-0.00567	low
Martinique	0.0057074	79.18	34.1	0.977	0.01374	0.00648	-0.00003	low
Mauritius	0.00680882	74.	31.9	0.844	0.01441	0.00659	-0.00006	low
Mayotte	0.03317	62.91	17.2	0.844	0.03926	0.0072	0.00111	low
Nigeria	0.0236283	46.94	19.	0.68	0.03665	0.01656	-0.0001	low
Saint Pierre and Miquelon	0.00085	79.07	35.2	0.99	0.01276	0.00695	-0.00496	low
Saint Vincent and the Grenadines	0.00102748	73.65	28.9	0.96	0.01527	0.00691	-0.0118	low
South Korea	0.00396607	78.72	37.3	0.979	0.00893	0.00594	-0.00033	low
Thailand	0.00607686	73.1	33.3	0.926	0.0134	0.00725	0.	low
Trinidad and Tobago	0.00389394	70.86	32.1	0.986	0.01436	0.00811	-0.00728	low
Uganda	0.0332703	52.72	15.	0.668	0.04784	0.01209	-0.00883	low
Yemen	0.0291064	63.27	16.8	0.502	0.04214	0.00761	0.	low
Zimbabwe	0.00109726	45.77	17.6	0.907	0.03149	0.01619	0.	low

Note that the first column, the one with the country names, is not needed for the NBC generation.

NBC generation

First we load the package [1]:

In[13]:=
```
Get[
 "~/MathFiles/MathematicaForPrediction/NaiveBayesianClassifier.m"]
```

With the commands below we find the indices of the rows of the training set with the label “low”, then take randomly 80% of them. We do the same for the label “high”. By joining these two lists of indices we obtain the list of indices of the training set. The list of indices for the test set is derived by complement.

In[14]:=
```
{tallyLow, tallyHigh} =
   {"low", "high"} /. (Rule @@@ Tally[cdataLabeled〚All, -1〛]);
trainingInds =
   Join[
    RandomSample[Flatten[Position[cdataLabeled〚All, -1〛, "low"]],
     Floor[0.8 * tallyLow]],
    RandomSample[Flatten[Position[cdataLabeled〚All, -1〛, "high"]],
     Floor[0.8 * tallyHigh]]
   ];
testInds = Complement[Range[1, Length[cdataLabeled]], trainingInds];
```

With the following command we generate NBC classifier functions for the labels in the training set. These functions are the ones described with formula (2). The NBC generation result is returned as a list of rules.

In[43]:=
```
nbcRules =
  MakeBayesianClassifiers[Rest /@ cdataLabeled[[trainingInds]], 8];
Magnify[nbcRules, 0.3]
```

Out[44]=
```
{high → (0.186047 Times @@ MapThread[#1[#2] &, {{
   Piecewise[{{0.632353, -1/50 ≤ #1 < 0}, {1.30461, 0 ≤ #1 < 1/50}, {0.43, 1/50 ≤ #1 < 1/25},
     {0., 1/25 ≤ #1 < 3/50}, {0, 3/50 ≤ #1 < 2/25 || 2/25 ≤ #1 < 1/10 || 1/10 ≤ #1 < 3/25},
     {5.375, 3/25 ≤ #1 < 7/50}, {0, True}}] &,
   Piecewise[{{0.316176, 70 ≤ #1 < 72}, {0., 72 ≤ #1 < 74}, {0.565789, 74 ≤ #1 < 76},
     {0.671875, 76 ≤ #1 < 78}, {2.95625, 78 ≤ #1 < 80}, {4.42647, 80 ≤ #1 < 82},
     {5.375, 82 ≤ #1 < 84 || 84 ≤ #1 < 86}, {0, True}}] &,
   Piecewise[{{0.447917, 25 ≤ #1 < 55/2}, {0.282895, 55/2 ≤ #1 < 30}, {1.075, 30 ≤ #1 < 65/2},
     {0.767857, 65/2 ≤ #1 < 35}, {2.89423, 35 ≤ #1 < 75/2}, {1.19444, 75/2 ≤ #1 < 40},
     {3.67763, 40 ≤ #1 < 85/2}, {5.375, 85/2 ≤ #1 < 45 || 45 ≤ #1 < 95/2}, {0, True}}] &,
   Piecewise[{{0.767857, 4/5 ≤ #1 < 33/40}, {0., 33/40 ≤ #1 < 17/20 || 17/20 ≤ #1 < 7/8},
     {0.597222, 7/8 ≤ #1 < 9/10 || 9/10 ≤ #1 < 37/40}, {1.15179, 37/40 ≤ #1 < 19/20},
     {0.848684, 19/20 ≤ #1 < 39/40}, {2.19388, 39/40 ≤ #1 < 1}, {0, True}}] &,
   Piecewise[{{5.375, 3/500 ≤ #1 < 1/125}, {2.41875, 1/125 ≤ #1 < 1/100}, {2.26316, 1/100 ≤ #1 < 3/250},
     {2.09028, 3/250 ≤ #1 < 7/500}, {1.79167, 7/500 ≤ #1 < 2/125}, {0.358333, 2/125 ≤ #1 < 9/500},
     {0., 9/500 ≤ #1 < 1/50}, {0.413462, 1/50 ≤ #1 < 11/500}, {0, True}}] &,
   Piecewise[{{1.6125, 1/500 ≤ #1 < 1/250}, {0.581081, 1/250 ≤ #1 < 3/500}, {0.871622, 3/500 ≤ #1 < 1/125},
     {1.69737, 1/125 ≤ #1 < 1/100}, {1.79167, 1/100 ≤ #1 < 3/250}, {0.413462, 3/250 ≤ #1 < 7/500},
     {0, True}}] &,
   Piecewise[{{0.597222, -1/100 ≤ #1 < -1/200}, {0.179167, -1/200 ≤ #1 < 0}, {1.2608, 0 ≤ #1 < 1/200},
     {2.6875, 1/200 ≤ #1 < 1/100}, {3.58333, 1/100 ≤ #1 < 3/200}, {5.375, 3/200 ≤ #1 < 1/50},
     {0, True}}] &}, #1}] &),
 low → (0.813953 Times @@ MapThread[#1[#2] &, {{
   Piecewise[{{1.22857, -1/50 ≤ #1 < -1/100}, {1.06476, -1/100 ≤ #1 < 0}, {0.819048, 0 ≤ #1 < 1/100},
     {1.05306, 1/100 ≤ #1 < 1/50}, {1.10256, 1/50 ≤ #1 < 3/100},
     {1.22857, 3/100 ≤ #1 < 1/25 || 1/25 ≤ #1 < 1/20}, {0, True}}] &,
   Piecewise[{{1.22857, 30 ≤ #1 < 40 || 40 ≤ #1 < 50 || 50 ≤ #1 < 60 || 60 ≤ #1 < 70},
     {1.01491, 70 ≤ #1 < 80}, {0.193985, 80 ≤ #1 < 90}, {0, True}}] &,
   Piecewise[{{1.22857, 15 ≤ #1 < 20 || 20 ≤ #1 < 25}, {1.14931, 25 ≤ #1 < 30}, {1.01176, 30 ≤ #1 < 35},
     {0.792627, 35 ≤ #1 < 40}, {0.35102, 40 ≤ #1 < 45}, {0, True}}] &,
   Piecewise[{{1.22857, 1/5 ≤ #1 < 3/10 || 3/10 ≤ #1 < 2/5 || 2/5 ≤ #1 < 1/2 || 1/2 ≤ #1 < 3/5 || 3/5 ≤ #1 < 7/10 || 7/10 ≤ #1 < 4/5},
     {1.14667, 4/5 ≤ #1 < 9/10}, {0.86405, 9/10 ≤ #1 < 1}, {0, True}}] &,
   Piecewise[{{0.614286, 0 ≤ #1 < 1/100}, {0.905263, 1/100 ≤ #1 < 1/50}, {1.19624, 1/50 ≤ #1 < 3/100},
     {1.22857, 3/100 ≤ #1 < 1/25 || 1/25 ≤ #1 < 1/20}, {0, True}}] &,
   Piecewise[{{0.955556, 0 ≤ #1 < 1/200}, {0.982857, 1/200 ≤ #1 < 1/100}, {0.967965, 1/100 ≤ #1 < 3/200},
     {1.22857, 3/200 ≤ #1 < 1/50 || 1/50 ≤ #1 < 1/40}, {0, 1/40 ≤ #1 < 3/100},
     {1.22857, 3/100 ≤ #1 < 7/200}, {0, True}}] &,
   Piecewise[{{1.22857, -3/100 ≤ #1 < -1/50 || -1/50 ≤ #1 < -1/100}, {1.17516, -1/100 ≤ #1 < 0},
     {0.89831, 0 ≤ #1 < 1/100}, {0.245714, 1/100 ≤ #1 < 1/50}, {1.22857, 1/50 ≤ #1 < 3/100},
     {0, True}}] &}, #1}] &)}
```

We assign to the symbols `hf` and `lf` the probabilities functions for “high” and “low” respectively:

In[45]:=
```
{hf, lf} = {"high" /. nbcRules, "low" /. nbcRules};
```

Classification

We do the classification with the function `NBCClassify`, which implements formula (3).

In[46]:=
```
res = NBCClassify[{hf, "high"}, {lf, "low"}, 0.5,
    0.8, Rest[Most[#]], All] & /@ cdataLabeled[[testInds]]
```

Out[46]=
```
{high, low, low, low, high, low, low, low, low, low, low,
 high, low, low, low, low, low, low, low, high, low, low,
 low, low, low, low, low, high, high, low, low, low, low,
 low, low, high, low, high, low, low, low, high, low, low}
```

If we do not specify the labels, then the classification result is returned as

```
{True|False..}.
```

In[47]:=
```
NBCClassify[hf, lf, 0.5, 0.8, Rest[Most[#]], All] & /@
  cdataLabeled⟦testInds⟧
```

Out[47]=
```
{True, False, False, False, True, False, False, False,
 False, False, False, True, False, False, False, False, False,
 False, False, True, False, False, False, False, False, False,
 False, True, True, False, False, False, False, False, False,
 True, False, True, False, False, False, True, False, False}
```

Here is table with the actual labels and the predicted labels for the test set countries:

In[63]:=
```
gridData =
  Flatten /@ Transpose[{cdataLabeled⟦testInds, {1, -1}⟧, res}];
gridColumnNames = Style[#, Blue, FontFamily → "Times"] & /@ Join[
    propNames⟦{1}⟧, {"GPD per capita\nclass label", "Predicted"}];
Magnify[#, 0.6] &@Grid[List@
    Map[Grid[Prepend[#, gridColumnNames], Alignment → Left] &,
     {gridData⟦1 ;; Floor[Length[gridData] / 2]⟧,
      gridData⟦Floor[Length[gridData] / 2] + 1 ;; -1⟧}], Spacings → 2]
```

Out[64]=

Name	GPD per capita class label	Predicted	Name	GPD per capita class label	Predicted
Andorra	high	high	Mayotte	low	low
Angola	low	low	Mexico	low	low
Antigua and Barbuda	low	low	Moldova	low	low
Belize	low	low	Mongolia	low	low
Bermuda	high	high	Nepal	low	low
Brunei	high	low	Niger	low	high
Colombia	low	low	Norway	high	high
Cuba	low	low	Philippines	low	low
Estonia	low	low	Saint Kitts and Nevis	low	low
Ethiopia	low	low	Saint Lucia	low	low
Gaza Strip	low	low	Saint Vincent and the Grenadines	low	low
Germany	high	high	Samoa	low	low
Guadeloupe	low	low	Seychelles	low	low
Guam	low	low	South Korea	low	high
Haiti	low	low	Sudan	low	low
Iran	low	low	Taiwan	low	high
Kyrgyzstan	low	low	Trinidad and Tobago	low	low
Laos	low	low	Tunisia	low	low
Lesotho	low	low	United Arab Emirates	high	low
Liechtenstein	high	high	United Kingdom	high	high
Madagascar	low	low	Yemen	low	low
Marshall Islands	low	low	Zimbabwe	low	low

We can compute statistics of the comparison

In[50]:=
```
Count[MapThread[Equal, {cdataLabeled⟦testInds, -1⟧, res}], True] /
  Length[res] // N
```

Out[50]=
```
0.886364
```

We can also use the function `NBCClassificationStatistics` provided by [1] to compute the classifier success ratios for the different classes of records:

In[51]:=
```
NBCClassificationStatistics[{hf, "high"},
  {lf, "low"}, 0.5, 0.8, cdataLabeled[[testInds]],
  Range[2, Dimensions[cdataLabeled][[2]] - 1]] //
 Grid[Prepend[#, Style[#, Blue, FontFamily → "Times"] & /@
    {"type", "success ratio"}], Alignment → Left] &
```

Out[51]=

type	success ratio
all records	0.886364
high records	0.75
low records	0.916667

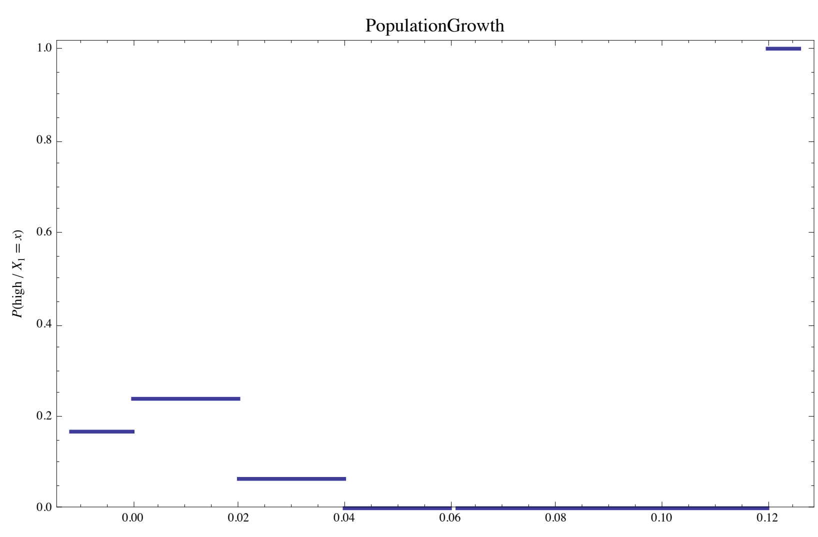
Plots

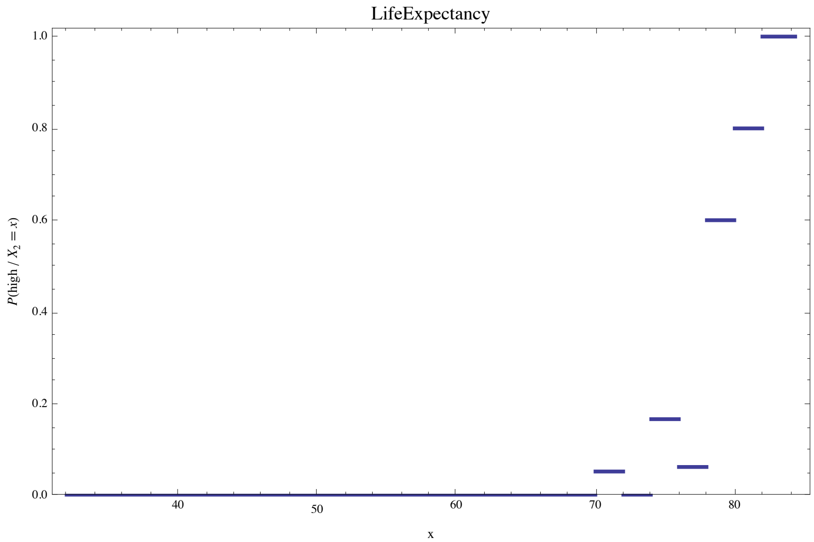
This section has code for plotting the S_i's that correspond to the variables.

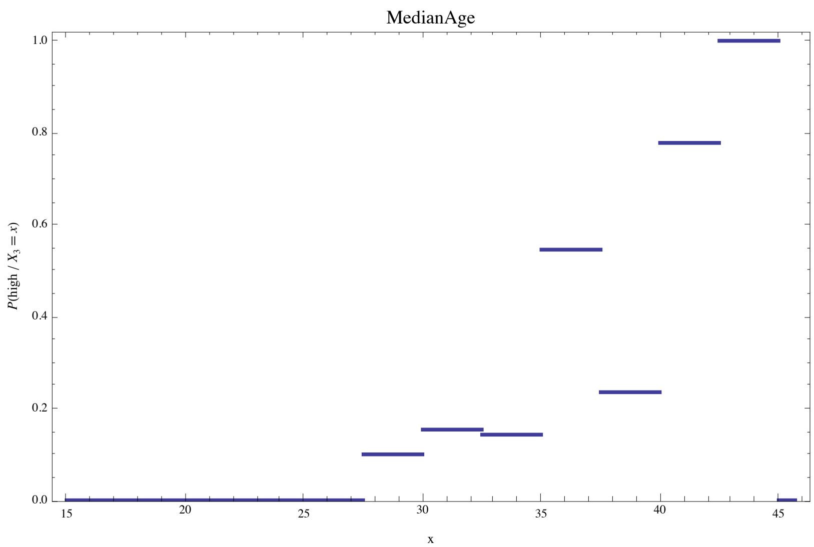
In[52]:=
```
factor = hf[[1, 1]];
funcs = Cases[hf, _Piecewise, ∞];
funcs = Table[
   With[{f = factor, fun = funcs[[i]]}, f * fun &], {i, Length[funcs]}];
```

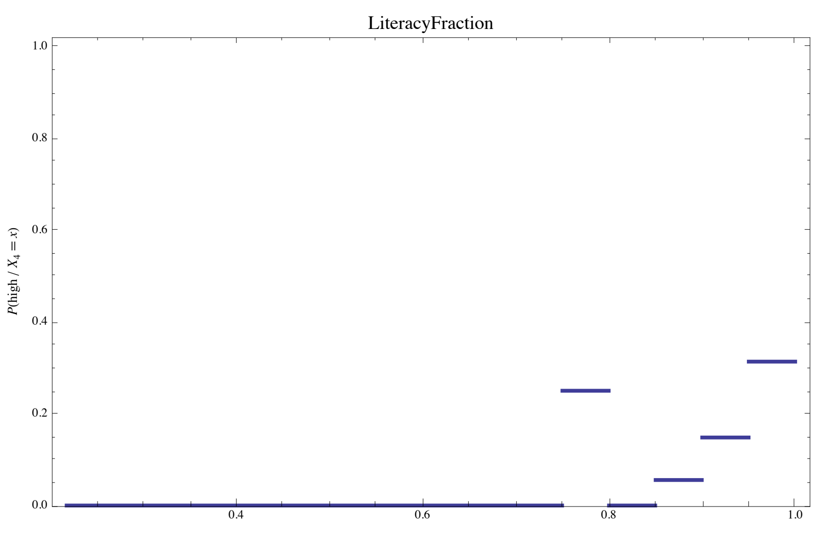
In[116]:=
```
nbcPlots = Table[
   Plot[funcs[[ind]][x],
    {x, Min[cdata[[All, ind + 1]]], Max[cdata[[All, ind + 1]]]},
    PlotRange → {All, {0, 1.02}}, PlotStyle → Thickness[0.005],
    Frame -> True, FrameLabel → Map[Style[#, Larger] &,
      {"x", TraditionalForm[P[Row[{"high", " / ", X_ind == x}]]]}],
    Axes → False, PlotLabel → Style[(propNames[[ind + 1]]), Larger],
    ImageSize → 600], {ind, Range[1, 7]}];
```

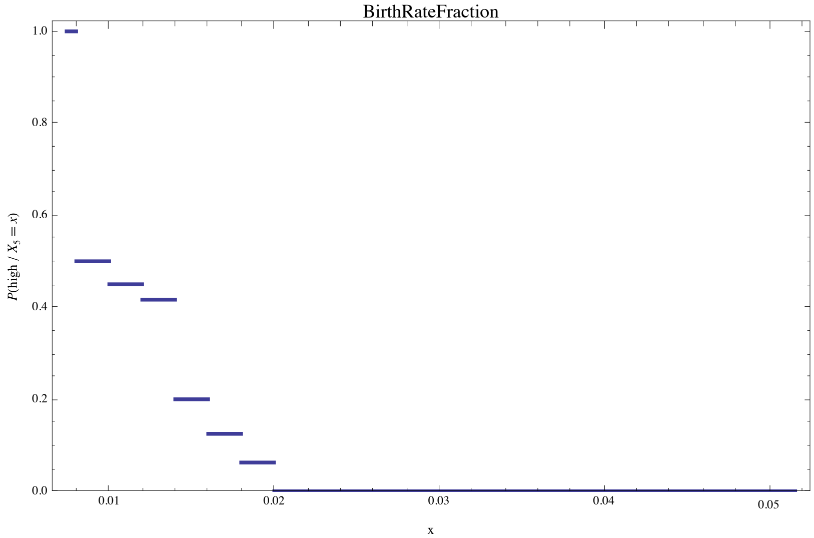
In[118]:=
```
Print[Magnify[#, 0.7]] & /@ nbcPlots[[1 ;; 7]];
```

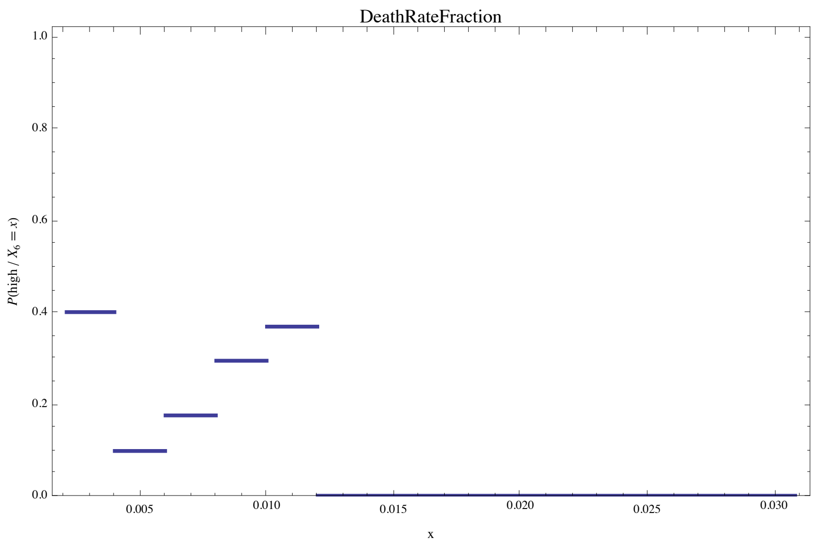


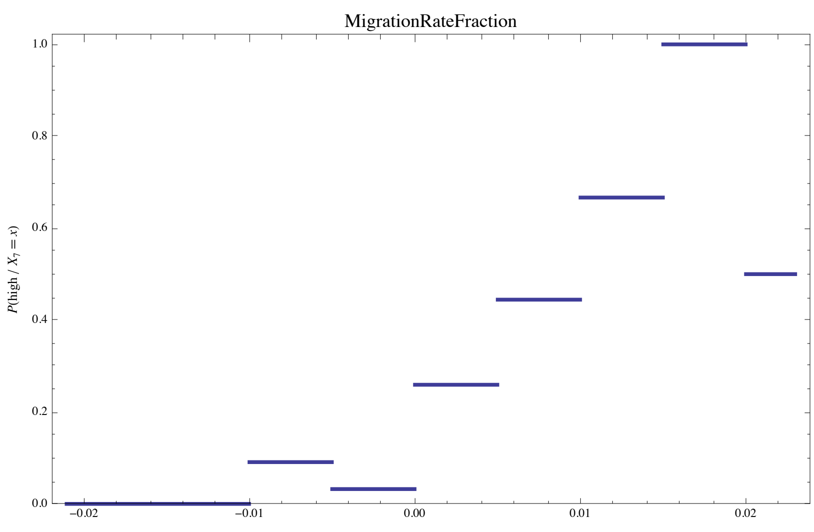

LifeExpectancy
1.0
0.8
0.6
0.4
0.2
0.0
$P(\text{high} / X_2 = x)$
40
50
60
70
80
x


MedianAge
1.0
0.8
0.6
0.4
0.2
0.0
$P(\text{high} / X_3 = x)$
15
20
25
30
35
40
45
x


LiteracyFraction
1.0
0.8
0.6
0.4
0.2
0.0
$P(\text{high} / X_4 = x)$
0.4
0.6
0.8
1.0
x


BirthRateFraction
$P(\text{high} / X_5 = x)$
1.0
0.8
0.6
0.4
0.2
0.0
0.01
0.02
0.03
0.04
0.05
x


DeathRateFraction
$P(\text{high} / X_6 = x)$
1.0
0.8
0.6
0.4
0.2
0.0
0.005
0.010
0.015
0.020
0.025
0.030
x


MigrationRateFraction
$P(\text{high} / X_7 = x)$
1.0
0.8
0.6
0.4
0.2
0.0
−0.02
−0.01
0.00
0.01
0.02
x

References

[1] Anton Antonov, Implementation of naive Bayesian classifier generation in *Mathematica*, source code at GitHub, https://github.com/antononcube/MathematicaForPrediction, package NaiveBayesianClassifier.m, (2013).

[2] Wikipedia, Naive Bayes Classifier, http://en.wikipedia.org/wiki/Naive_Bayes_classifier .