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:

Out[40]=

Name	PopulationGrowth	LifeExpectancy	MedianAge	LiteracyFraction	BirthRateFraction	DeathRateFraction	MigrationRateFraction	GPD per capita class label
Algeria	0.0152181	74.02	26.6	0.699	0.0169	0.00464	-0.00029	low
Cape Verde	0.0141443	71.61	21.1	0.766	0.0235	0.00622	-0.01167	low
French Polynesia	0.0128463	76.71	29.1	0.98	0.01591	0.00473	0.00273	low
Germany	-0.000951597	79.26	43.8	0.99	0.00818	0.0109	0.00219	high
Ivory Coast	0.0232824	55.45	19.2	0.487	0.03211	0.01078	0.	low
Kyrgyzstan	0.0126324	69.43	24.4	0.987	0.02344	0.00691	-0.00257	low
Panama	0.0165948	77.25	27.	0.919	0.02018	0.00466	-0.00049	low
South Africa	0.0100556	48.98	24.4	0.864	0.01993	0.01699	-0.00013	low
Sudan	0.022641	51.42	19.1	0.611	0.03374	0.01294	0.00063	low
Ukraine	-0.00641837	68.25	39.5	0.994	0.0096	0.01581	-0.00011	low

We assume that have the following observed variables.

Out[42]=

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:

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

For each value $c \in \{\text{True, False}\}$ 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:

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

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

Out[44]=

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[8]:=
```
countries = CountryData["Countries"];
countries // Length
```

Out[9]= 240

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

In[10]:=
```
propNames =
  {"Name", "PopulationGrowth", "LifeExpectancy", "MedianAge",
   "LiteracyFraction", "BirthRateFraction", "DeathRateFraction",
   "MigrationRateFraction", "Population", "GDP"};
cdata = Map[Table[CountryData[#, p], {p, propNames}] &, countries];
cdata // Length
```

Out[12]= 240

We filter out the countries with missing data.

In[13]:=
```
cdata = Select[cdata, VectorQ[Rest[#], NumberQ] &];
cdata // Length
```

Out[14]= 216

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

In[15]:=
```
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[16]:=
```
Tally[cdataLabeled[[All, -1]]]
```

Out[16]=
```
{{low, 176}, {high, 40}}
```

Here is a sample of the data:

In[17]:=
```
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[19]=

Name	PopulationGrowth	LifeExpectancy	MedianAge	LiteracyFraction	BirthRateFraction	DeathRateFraction	MigrationRateFraction	GPD per capita class label
Anguilla	0.0238851	80.65	32.6	0.95	0.01302	0.00436	0.01406	low
Azerbaijan	0.011472	66.66	28.2	0.988	0.01762	0.0083	-0.00169	low
Burundi	0.0301446	52.09	16.7	0.593	0.04142	0.01267	0.00404	low
Cape Verde	0.0141443	71.61	21.1	0.766	0.0235	0.00622	-0.01167	low
Costa Rica	0.0135337	77.58	27.5	0.949	0.01743	0.00434	0.00047	low
Dominica	-0.00331146	75.55	29.8	0.94	0.01573	0.0082	-0.00545	low
Dominican Republic	0.0141664	73.7	24.9	0.87	0.02239	0.00528	-0.00222	low
Equatorial Guinea	0.0264508	61.61	18.9	0.87	0.03652	0.00949	0.	low
Ethiopia	0.0262862	55.41	16.9	0.427	0.04366	0.01155	-0.0002	low
Gibraltar	0.00111	80.19	40.5	0.8	0.01067	0.00956	0.	high
Guam	0.01365	78.01	29.1	0.99	0.01822	0.00457	0.	low
Iraq	0.0207009	69.94	20.4	0.741	0.0301	0.00503	0.	low
Kazakhstan	0.00735532	67.87	29.6	0.995	0.0166	0.00939	-0.0033	low
Martinique	0.0057074	79.18	34.1	0.977	0.01374	0.00648	-0.00003	low
Montserrat	0.00510638	72.76	28.5	0.97	0.01236	0.00844	0.	low
Mozambique	0.0234653	41.18	17.4	0.478	0.03798	0.02007	0.	low
Rwanda	0.0281516	50.52	18.7	0.704	0.03967	0.01402	0.00217	low
Saint Kitts and Nevis	0.0128528	73.2	28.6	0.978	0.01767	0.00805	-0.00115	low
Vanuatu	0.0256876	63.98	24.2	0.74	0.02153	0.00755	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[20]:=
```
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[21]:= {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[24]:= nbcRules =
           MakeBayesianClassifiers[Rest /@ cdataLabeled⟦trainingInds⟧, 8];
         Magnify[nbcRules, 0.3]
```

```
Out[25]= {high -> (0.186047 Times @@ MapThread[#1[#2] &, {{
   { 0.948529   -1/50 ≤ #1 < 0
     1.33095    0 ≤ #1 < 1/50
     0.223958   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 } &,
   { 0.255952   70 ≤ #1 < 72
     0.         72 ≤ #1 < 74
     0.948529   74 ≤ #1 < 76
     0.413462   76 ≤ #1 < 78
     3.11184    78 ≤ #1 < 80
     4.70313    80 ≤ #1 < 82
     5.375      82 ≤ #1 < 84
     0          True } &,
   { 0.488636   55/2 ≤ #1 < 30
     1.075      30 ≤ #1 < 65/2
     1.34375    65/2 ≤ #1 < 35
     2.44318    35 ≤ #1 < 75/2
     1.69737    75/2 ≤ #1 < 40
     4.03125    40 ≤ #1 < 85/2
     5.375      85/2 ≤ #1 < 45 || 45 ≤ #1 < 95/2
     0          True } &,
   { 0.895833   4/5 ≤ #1 < 33/40
     0.         33/40 ≤ #1 < 17/20 || 17/20 ≤ #1 < 7/8
     0.767857   7/8 ≤ #1 < 9/10
     0.         9/10 ≤ #1 < 37/40
     0.767857   37/40 ≤ #1 < 19/20
     0.358333   19/20 ≤ #1 < 39/40
     2.4016     39/40 ≤ #1 < 1
     0          True } &,
   { 5.375      3/500 ≤ #1 < 1/125
     2.52941    1/125 ≤ #1 < 1/100
     2.57065    1/100 ≤ #1 < 3/250
     2.6875     3/250 ≤ #1 < 7/500
     1.34375    7/500 ≤ #1 < 2/125
     0.316176   2/125 ≤ #1 < 9/500
     0.5375     9/500 ≤ #1 < 1/50
     0          True } &,
   { 1.53571    1/500 ≤ #1 < 1/250
     0.655488   1/250 ≤ #1 < 3/500
     0.826923   3/500 ≤ #1 < 1/125
     1.64236    1/125 ≤ #1 < 1/100
     2.09028    1/100 ≤ #1 < 3/250
     0.413462   3/250 ≤ #1 < 7/500
     0          True } &,
   { 0.488636   -1/100 ≤ #1 < -1/200
     0.170635   -1/200 ≤ #1 < 0
     1.39352    0 ≤ #1 < 1/200
     3.13542    1/200 ≤ #1 < 1/100
     0.         1/100 ≤ #1 < 3/200
     5.375      3/200 ≤ #1 < 1/50
     0          True } &}, #1}] &),
 low -> (0.813953 Times @@ MapThread[#1[#2] &, {{
   { 1.22857    -1/50 ≤ #1 < -1/100
     0.998214   -1/100 ≤ #1 < 0
     0.826494   0 ≤ #1 < 1/100
     1.032      1/100 ≤ #1 < 1/50
     1.16216    1/50 ≤ #1 < 3/100
     1.22857    3/100 ≤ #1 < 1/25 || 1/25 ≤ #1 < 1/20
     0          True } &,
   { 1.22857    30 ≤ #1 < 40 || 40 ≤ #1 < 50 || 50 ≤ #1 < 60 || 60 ≤ #1 < 70
     1.01016    70 ≤ #1 < 80
     0.136508   80 ≤ #1 < 90
     0          True } &,
   { 1.22857    15 ≤ #1 < 20 || 20 ≤ #1 < 25
     1.15411    25 ≤ #1 < 30
     0.965306   30 ≤ #1 < 35
     0.778095   35 ≤ #1 < 40
     0.258647   40 ≤ #1 < 45
     0          True } &,
   { 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
     1.13757    4/5 ≤ #1 < 9/10
     0.885714   9/10 ≤ #1 < 1
     0          True } &,
   { 0.614286   0 ≤ #1 < 1/100
     0.856767   1/100 ≤ #1 < 1/50
     1.22857    1/50 ≤ #1 < 3/100 || 3/100 ≤ #1 < 1/25 || 1/25 ≤ #1 < 1/20 || 1/20 ≤ #1 < 3/50
     0          True } &,
   { 0.949351   0 ≤ #1 < 1/200
     0.997454   1/200 ≤ #1 < 1/100
     0.930736   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 } &,
   { 1.22857    -1/40 ≤ #1 < -1/50
     0          -1/50 ≤ #1 < -3/200
     1.22857    -3/200 ≤ #1 < -1/100
     1.11688    -1/100 ≤ #1 < -1/200
     1.18957    -1/200 ≤ #1 < 0
     0.910053   0 ≤ #1 < 1/200
     0.511905   1/200 ≤ #1 < 1/100
     1.22857    1/100 ≤ #1 < 3/200
     0          True } &}, #1}] &)}
```

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

```
In[26]:= {hf, lf} = {"high" /. nbcRules, "low" /. nbcRules};
```

Classification

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

```
In[27]:= res = NBCClassify[{hf, "high"}, {lf, "low"}, 0.5,
             0.8, Rest[Most[#]], All] & /@ cdataLabeled⟦testInds⟧

Out[27]= {high, low, low, high, low, low, low, low, low, high, low,
          low, low, low, low, high, low, low, low, high, high, low,
          high, high, high, low, high, low, high, high, low, low, low,
          low, high, high, low, high, low, low, high, low, low, low}
```

If we do not specify the labels, then the classification result is returned as `{True|False..}`.

```
In[28]:= NBCClassify[hf, lf, 0.5, 0.8, Rest[Most[#]], All] & /@
          cdataLabeled⟦testInds⟧

Out[28]= {True, False, False, True, False, False, False, False,
          False, True, False, False, False, False, False, True, False,
          False, False, True, True, False, True, True, True, False,
          True, False, True, True, False, False, False, False, True,
          True, False, True, False, False, True, False, False, False}
```

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

In[29]:=
```
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[30]=

Name	GPD per capita class label	Predicted	Name	GPD per capita class label	Predicted
Afghanistan	low	high	Kuwait	high	high
Azerbaijan	low	low	Lithuania	low	high
Belarus	low	low	Macau	high	high
Bulgaria	low	high	Maldives	low	low
Central African Republic	low	low	Martinique	low	high
Chile	low	low	Moldova	low	low
China	low	low	Netherlands	high	high
Colombia	low	low	New Zealand	low	high
Comoros	low	low	Northern Mariana Islands	low	low
Czech Republic	low	high	Paraguay	low	low
Ecuador	low	low	Republic of the Congo	low	low
French Guiana	low	low	Réunion	low	low
French Polynesia	low	low	Saint Pierre and Miquelon	low	high
Gambia	low	low	San Marino	high	high
Gaza Strip	low	low	Senegal	low	low
Greece	high	high	Slovakia	low	high
Grenada	low	low	South Africa	low	low
Guadeloupe	low	low	Togo	low	low
Guam	low	low	United Arab Emirates	high	high
Hong Kong	high	high	Uruguay	low	low
Ireland	high	high	Yemen	low	low
Jamaica	low	low	Zambia	low	low

We can compute statistics of the comparison

In[31]:=
```
Count[MapThread[Equal, {cdataLabeled[[testInds, -1]], res}], True] /
  Length[res] // N
```

Out[31]= 0.818182

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

In[53]:=
```
resRules = NBCClassificationSuccess[
  NBCClassify[{hf, "high"}, {lf, "low"}, 0.5, 0.8, #] &,
  cdataLabeled[[testInds, 2 ;; -1]]]
```

Out[53]=
```
{{high, True} → 1., {high, False} → 0.,
 {low, True} → 0.777778, {low, False} → 0.222222,
 {All, True} → 0.818182, {All, False} → 0.181818}
```

The resulting rules are interpreted with the following table construction:

In[55]:=
```
Block[{labels = {"low", "high", All}, gridData},
 gridData = Outer[{#1, #2} /. resRules &, labels, {True, False}];
 gridData = MapThread[Prepend, {gridData, labels}];
 Grid[Prepend[gridData, Style[#, Blue, FontFamily → "Times"] & /@
     {"Label", "Fraction of\ncorrect guesses",
      "Fraction of\nincorrect guesses"}], Alignment → Left,
  Dividers → {{False, True, False}, {False, True, False}}]
]
```

Out[55]=

Label	Fraction of correct guesses	Fraction of incorrect guesses
low	0.777778	0.222222
high	1.	0.
All	0.818182	0.181818

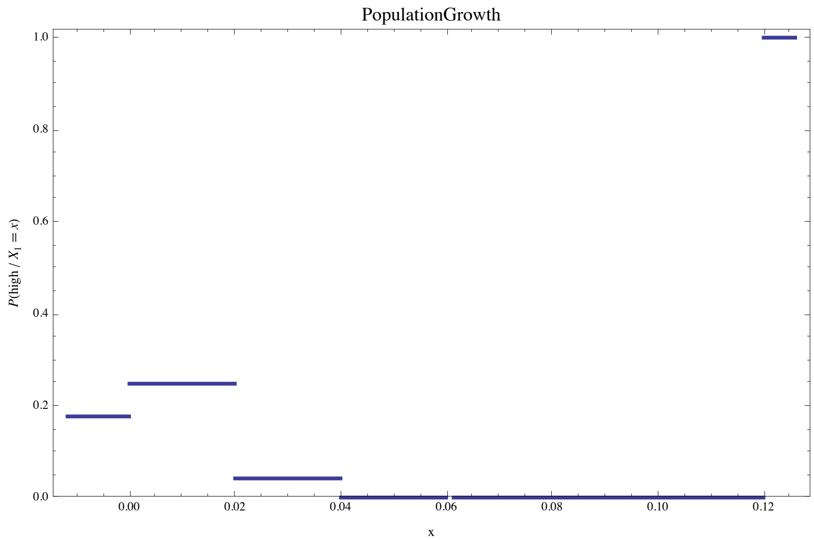
Plots

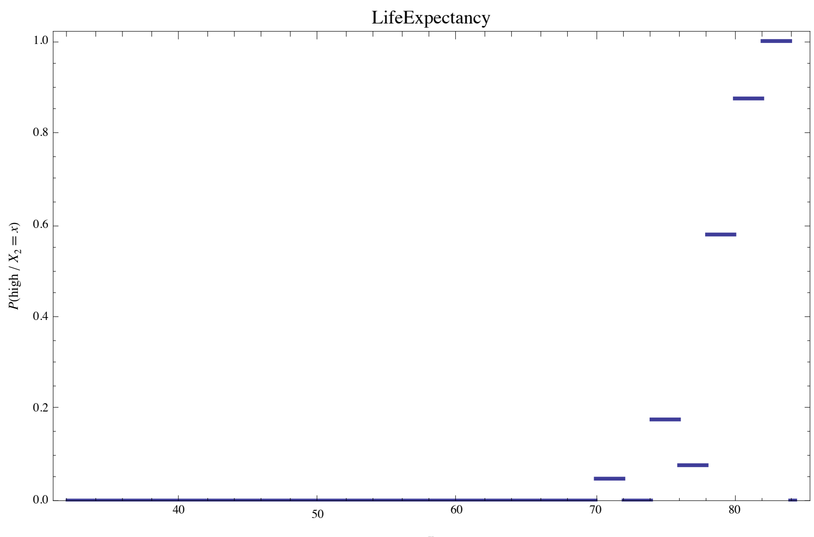
This section has code for plotting the S_i's that correspond to the variables.

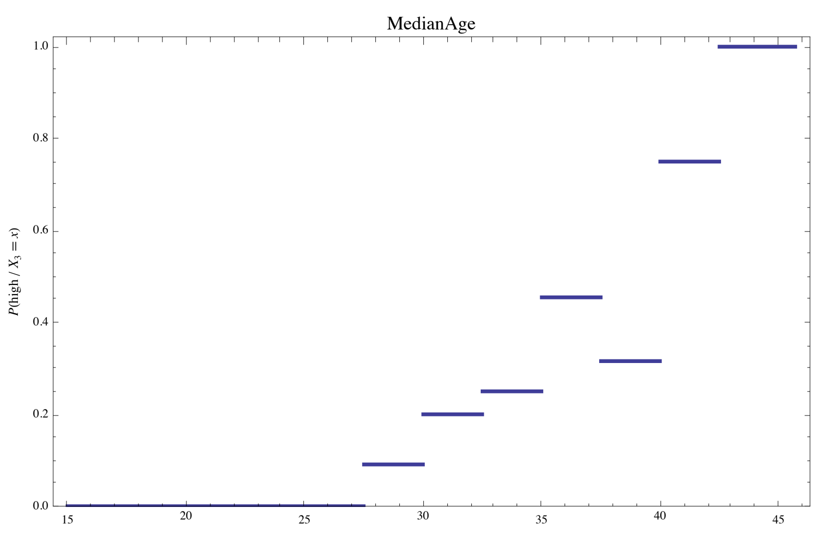
In[33]:=
```
factor = hf⟦1, 1⟧;
funcs = Cases[hf, _Piecewise, ∞];
funcs = Table[
   With[{f = factor, fun = funcs⟦i⟧}, f * fun &], {i, Length[funcs]}];
```

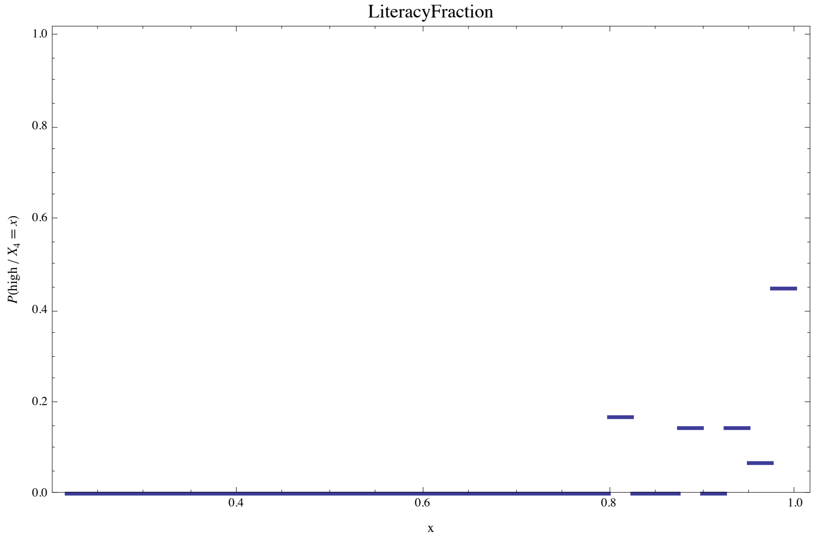
In[36]:=
```
nbcPlots = Table[
   Plot[funcs⟦ind⟧[x],
    {x, Min[cdata⟦All, ind + 1⟧], Max[cdata⟦All, ind + 1⟧]},
    PlotRange → {All, {0, 1.02}}, PlotStyle → Thickness[0.005],
    Frame -> True, FrameLabel → Map[Style[#, Larger] &,
      {"x", TraditionalForm[P[Row[{"high", "  /  ", X_ind == x}]]]}],
    Axes → False, PlotLabel → Style[(propNames⟦ind + 1⟧), Larger],
    ImageSize → 600], {ind, Range[1, 7]}];
```

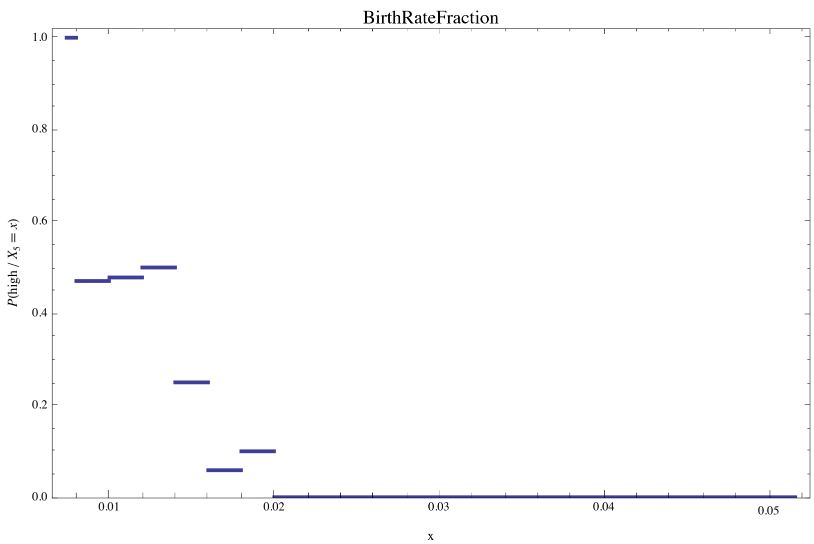
In[37]:=
```
Print[Magnify[#, 0.7]] & /@ nbcPlots⟦1 ;; 7⟧;
```

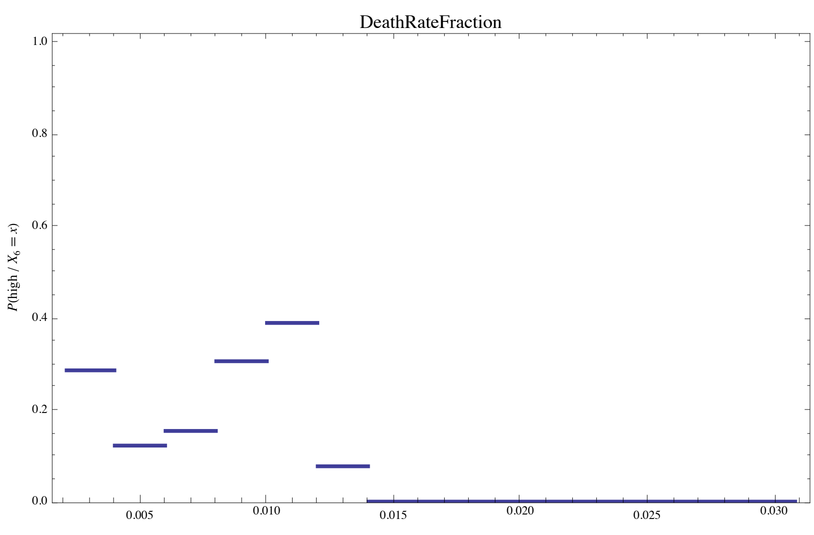

PopulationGrowth
P(high / X_1 = x)
x

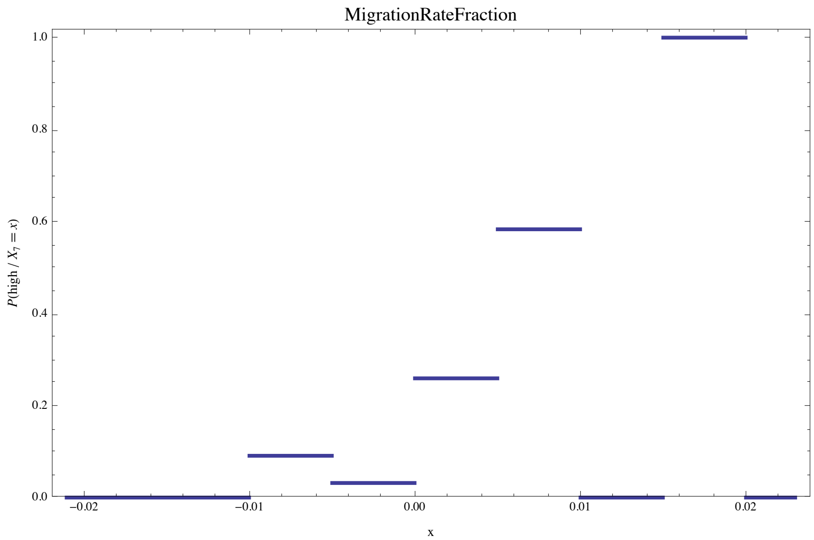

LifeExpectancy
P(high / X_2 = x)
x


MedianAge
P(high / X_3 = x)
x


LiteracyFraction
$P(\text{high} / X_4 = x)$
x


BirthRateFraction
$P(\text{high} / X_5 = x)$
x


DeathRateFraction
$P(\text{high} / X_6 = x)$
x


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