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[613]=

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.

Out[549]=

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 $\text{dom}(X) = \mathbb{R}$.) Let $D_i := \text{dom}(X_i)$, 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

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

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 $\{p_1, \ldots, p_{n-1}\}$, $p_j \in [0, 1]$, $1 \le j \le n - 1$. With these lists we can compute the function

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

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 $\{p_1, \ldots, p_n\}$, $p_j \in [0, 1]$, $1 \le j \le n$. 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[549]=

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[667]:=
```
countries = CountryData["Countries"];
countries // Length
```

Out[668]= 240

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

In[671]:=
```
propNames =
  {"Name", "PopulationGrowth", "LifeExpectancy", "MedianAge",
   "LiteracyFraction", "BirthRateFraction", "DeathRateFraction",
   "MigrationRateFraction", "Population", "GDP"};
cdata = Map[Table[CountryData[#, p], {p, propNames}] &, countries];
cdata // Length
```

Out[673]= 240

We filter out the countries with missing data.

In[674]:=
```
cdata = Select[cdata, VectorQ[Rest[#], NumberQ] &];
cdata // Length
```

Out[675]= 216

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

```
In[50]:= 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[676]:= Tally[cdataLabeled[[All, -1]]]
Out[676]= {{low, 176}, {high, 40}}
```

Here is a sample of the data:

```
In[695]:= 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[697]=

Name	PopulationGrowth	LifeExpectancy	MedianAge	LiteracyFraction	BirthRateFraction	DeathRateFraction	MigrationRateFraction	GPD per capita class label
Albania	0.00345831	77.96	29.9	0.987	0.01529	0.00555	-0.00428	low
Aruba	0.0122773	75.28	37.8	0.973	0.01279	0.00771	0.0097	low
Belgium	0.00565979	79.22	41.7	0.99	0.01015	0.01044	0.00122	high
Cambodia	0.0166272	62.1	22.1	0.736	0.02573	0.00808	0.	low
Cuba	0.0000265074	77.45	37.3	0.998	0.01113	0.00724	-0.00156	low
Cyprus	0.0100962	78.33	35.5	0.976	0.01257	0.0078	0.00042	high
Dominica	-0.00331146	75.55	29.8	0.94	0.01573	0.0082	-0.00545	low
Egypt	0.018319	72.12	24.8	0.714	0.0217	0.00508	-0.0002	low
Equatorial Guinea	0.0264508	61.61	18.9	0.87	0.03652	0.00949	0.	low
Gabon	0.0184804	53.11	18.6	0.632	0.03557	0.01276	-0.00348	low
Guyana	-0.000867687	66.68	28.7	0.988	0.01756	0.00831	-0.00744	low
Iceland	0.0238012	80.67	35.1	0.99	0.01343	0.00685	0.00083	high
Macedonia	0.000737313	74.68	35.1	0.961	0.01197	0.00883	-0.00052	low
Netherlands	0.0041201	79.4	40.4	0.99	0.0104	0.00874	0.00246	high
Poland	-0.000737724	75.63	37.9	0.998	0.01004	0.01005	-0.00047	low
Rwanda	0.0281516	50.52	18.7	0.704	0.03967	0.01402	0.00217	low
Spain	0.00987781	80.05	41.1	0.979	0.00972	0.00999	0.00099	high
Togo	0.0250949	58.69	18.7	0.609	0.03644	0.00933	0.	low
Tunisia	0.00999914	75.78	29.2	0.743	0.01542	0.0052	-0.00041	low
Turkey	0.0124723	71.96	27.7	0.874	0.01866	0.0061	0.00056	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[731]:= 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[732]:=
```
{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[735]:=
```
nbcRules =
  MakeBayesianClassifiers[Rest /@ cdataLabeled[[trainingInds]], 10];
Magnify[nbcRules, 0.3]
```

Out[736]=
```
{high -> 0.186047 Times @@ MapThread[#1[#2] &, {{
  { 0.632353   -1/50 <= #1 < 0
    1.33095    0 <= #1 < 1/50
    0.335938   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                 &,
  { 0.413462   70 <= #1 < 72
    0.         72 <= #1 < 74
    0.597222   74 <= #1 < 76
    1.00781    76 <= #1 < 78
    3.03804    78 <= #1 < 80
    4.3        80 <= #1 < 82
    5.375      82 <= #1 < 84
    0          True                 &,
  { 0.413462   26 <= #1 < 28 || 28 <= #1 < 30
    1.075      30 <= #1 < 32
    1.53571    32 <= #1 < 34
    1.79167    34 <= #1 < 36
    1.075      36 <= #1 < 38
    1.79167    38 <= #1 < 40
    3.94167    40 <= #1 < 42
    5.375      42 <= #1 < 44 || 44 <= #1 < 46
    0          True                 &,
  { 1.79167    31/40 <= #1 < 4/5
    0.767857   4/5 <= #1 < 33/40
    0.         33/40 <= #1 < 17/20 || 17/20 <= #1 < 7/8
    0.895833   7/8 <= #1 < 9/10
    0.         9/10 <= #1 < 37/40
    0.447917   37/40 <= #1 < 19/20
    0.767857   19/20 <= #1 < 39/40
    2.10784    39/40 <= #1 < 1
    0          True                 &,
  { 2.52941    1/125 <= #1 < 1/100
    2.46354    1/100 <= #1 < 3/250
    2.01563    3/250 <= #1 < 7/500
    1.53571    7/500 <= #1 < 2/125
    0.597222   2/125 <= #1 < 9/500
    0.         9/500 <= #1 < 1/50
    0.488636   1/50 <= #1 < 11/500
    0          True                 &,
  { 1.6125     1/500 <= #1 < 1/250
    0.597222   1/250 <= #1 < 3/500
    0.80625    3/500 <= #1 < 1/125
    1.59797    1/125 <= #1 < 1/100
    1.79167    1/100 <= #1 < 3/250
    0.447917   3/250 <= #1 < 7/500
    0          True                 &,
  { 0.671875   -1/100 <= #1 < -1/200
    0.158088   -1/200 <= #1 < 0
    1.44712    0 <= #1 < 1/200
    2.44318    1/200 <= #1 < 1/100
    0.         1/100 <= #1 < 3/200
    5.375      3/200 <= #1 < 1/50
    2.6875     1/50 <= #1 < 1/40
    0          True                 &}, #1}] &,
 low -> 0.813953 Times @@ MapThread[#1[#2] &, {{
  { 1.22857    -1/50 <= #1 < -1/100
    1.06476    -1/100 <= #1 < 0
    0.804156   0 <= #1 < 1/100
    1.05657    1/100 <= #1 < 1/50
    1.12327    1/50 <= #1 < 3/100
    1.22857    3/100 <= #1 < 1/25 || 1/25 <= #1 < 1/20
    0          True                 &,
  { 1.22857    30 <= #1 < 35 || 35 <= #1 < 40 || 40 <= #1 < 45 || 45 <= #1 < 50 || 50 <= #1 < 55 || 55 <= #1 < 60 || 60 <= #1 < 65 || 65 <= #1 < 70
    1.17143    70 <= #1 < 75
    0.793452   75 <= #1 < 80
    0.230357   80 <= #1 < 85
    0          True                 &,
  { 1.22857    15 <= #1 < 20 || 20 <= #1 < 25
    1.15179    25 <= #1 < 30
    0.982857   30 <= #1 < 35
    0.832258   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
    1.10571    7/10 <= #1 < 4/5
    1.13757    4/5 <= #1 < 9/10
    0.911075   9/10 <= #1 < 1
    0          True                 &,
  { 0.65042    1/200 <= #1 < 1/100
    0.716667   1/100 <= #1 < 3/200
    1.12327    3/200 <= #1 < 1/50
    1.16032    1/50 <= #1 < 1/40
    1.22857    1/40 <= #1 < 3/100 || 3/100 <= #1 < 7/200 || 7/200 <= #1 < 1/25 || 1/25 <= #1 < 9/200 || 9/200 <= #1 < 1/20 || 1/20 <= #1 < 11/200
    0          True                 &,
  { 0.921429   0 <= #1 < 1/200
    1.00194    1/200 <= #1 < 1/100
    0.955556   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    -3/200 <= #1 < -1/100
    1.075      -1/100 <= #1 < -1/200
    1.19244    -1/200 <= #1 < 0
    0.897802   0 <= #1 < 1/200
    0.67013    1/200 <= #1 < 1/100
    1.22857    1/100 <= #1 < 3/200
    0.         3/200 <= #1 < 1/50
    0.614286   1/50 <= #1 < 1/40
    0          True                 &}, #1}] &}
```

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

In[737]:=
```
{hf, lf} = {"high" /. nbcRules, "low" /. nbcRules};
```

Classification

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

In[738]:=
```
res = NBCClassify[{hf, "high"}, {lf, "low"}, 0.5,
    0.8, Rest[Most[#]], All] & /@ cdataLabeled[[testInds]]
```

Out[738]=
```
{low, high, low, low, low, low, low, low, high, low, high,
 low, low, low, low, low, low, high, high, low, high, low,
 low, low, low, high, low, low, low, low, low, low, high,
 high, low, high, low, low, high, low, high, low, low, low}
```

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

In[739]:=
```
NBCClassify[hf, lf, 0.5, 0.8, Rest[Most[#]], All] & /@
 cdataLabeled[[testInds]]
```

Out[739]=
```
{False, True, False, False, False, False, False, False,
 True, False, True, False, False, False, False, False, False,
 True, True, False, True, False, False, False, False, True,
 False, False, False, False, False, False, True, True, False,
 True, False, False, True, False, True, False, False, False}
```

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

In[740]:=
```
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[Prepend[gridData, gridColumnNames],
   Alignment → Left]
```

Out[741]=

Name	GPD per capita class label	Predicted
American Samoa	low	low
Australia	high	high
Bahamas	low	low
Belarus	low	low
Brazil	low	low
Brunei	high	low
Cape Verde	low	low
Chad	low	low
Croatia	low	high
Cuba	low	low
Denmark	high	high
Djibouti	low	low
Ecuador	low	low
French Guiana	low	low
French Polynesia	low	low
Guadeloupe	low	low
Guam	low	low
Hong Kong	high	high
Japan	high	high
Libya	low	low
Macau	high	high
Malawi	low	low
Malaysia	low	low
Marshall Islands	low	low
Mauritania	low	low
Micronesia	low	high
Montserrat	low	low
Nicaragua	low	low
Nigeria	low	low
Paraguay	low	low
Saint Vincent and the Grenadines	low	low
Samoa	low	low
San Marino	high	high
Serbia	low	high
Sierra Leone	low	low
Singapore	high	high
Somalia	low	low
Sudan	low	low
Taiwan	low	high
Tonga	low	low
Trinidad and Tobago	low	high
Turkey	low	low
Vanuatu	low	low
Zimbabwe	low	low

We can compute statistics of the comparison

In[742]:=
```
Count[MapThread[Equal, {cdataLabeled[[testInds, -1]], res}], True] /
  Length[res] // N
```

Out[742]= 0.863636

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

In[756]:=
```
NBCClassificationStatistics[{hf, "high"}, {lf, "low"}, 0.5, 0.8,
  cdataLabeled[[testInds]], Range[2, Dimensions[cdataLabeled][[2]] - 1]]
```

Out[756]=
```
{{all records, 0.863636},
 {high records, 0.875}, {low records, 0.861111}}
```

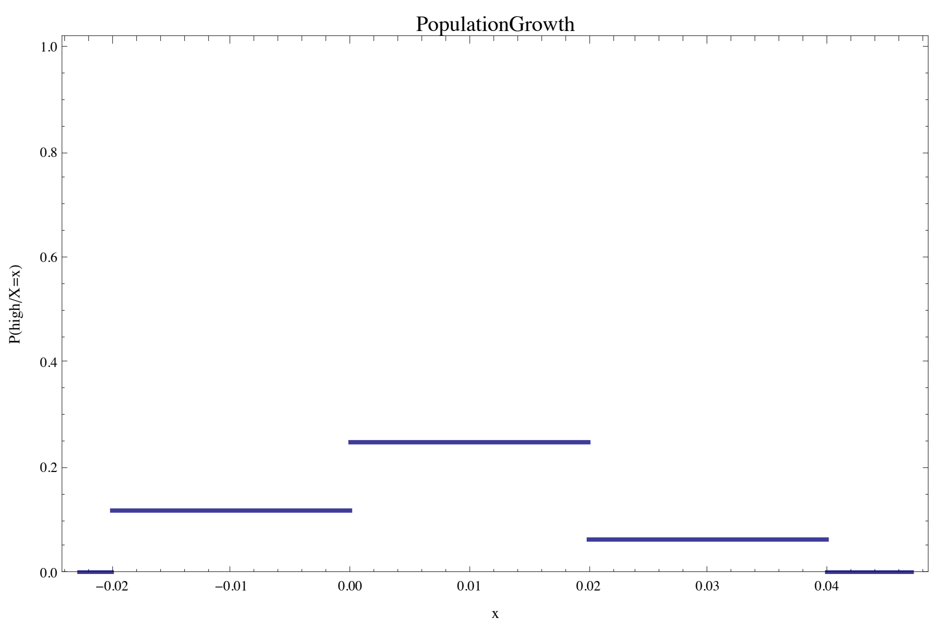
Plots

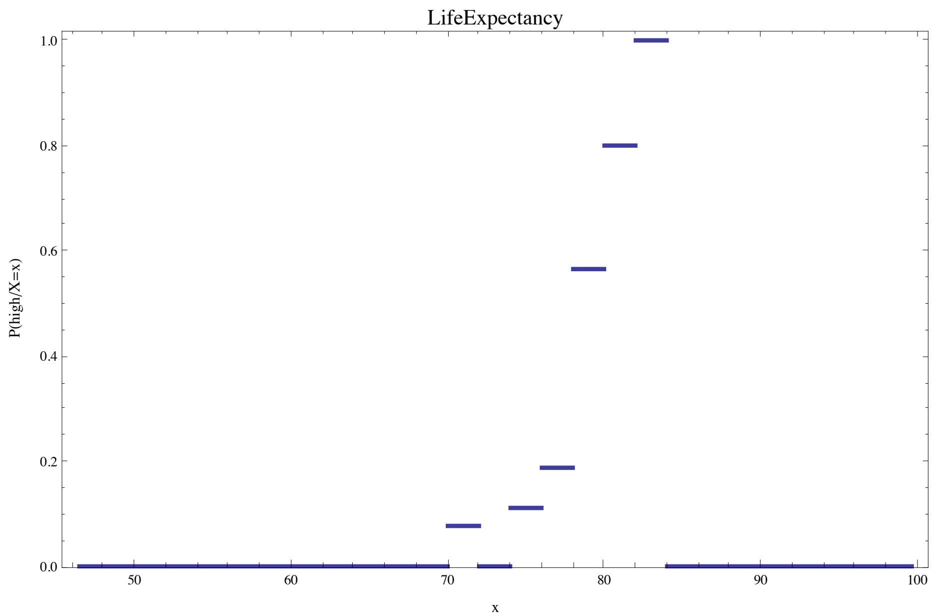
Below are plots of S_i's for four of the variables.

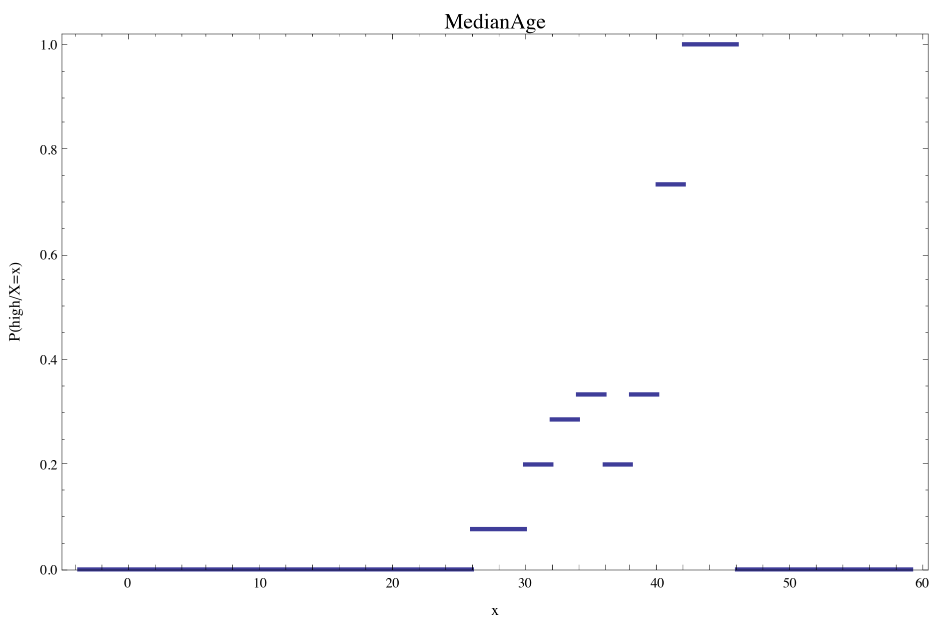
In[752]:=
```
funcs = Cases[hf, _Piecewise, ∞];
funcs = Table[
   With[{f = factor, fun = funcs[[i]]}, f * fun &], {i, Length[funcs]}];
```

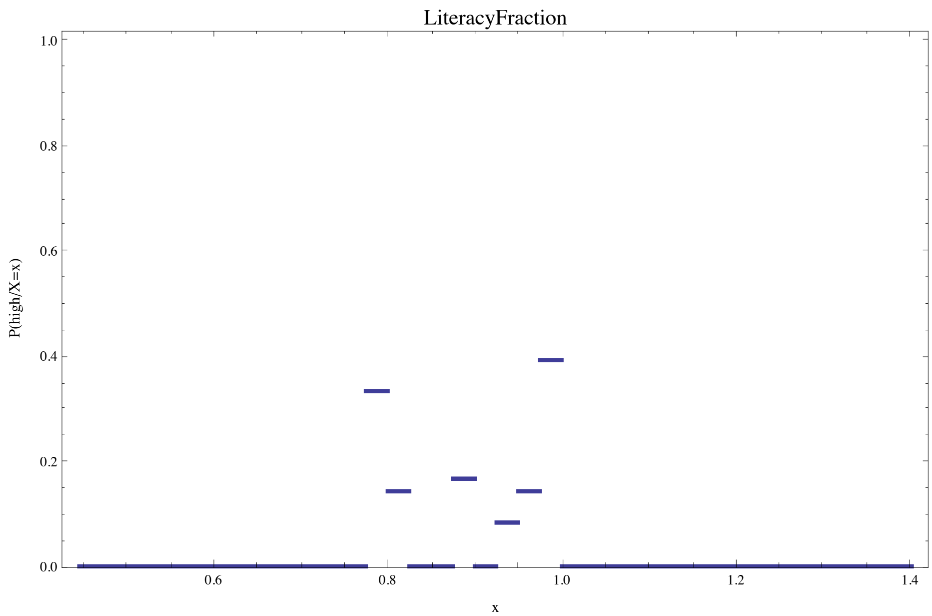
In[754]:=
```
nbcPlots = Table[
   Plot[funcs[[ind]][x], {x, cs[[ind, 1]] - 2 cs[[ind, 2]],
     cs[[ind, 1]] + 2 cs[[ind, 2]]}, PlotRange → {All, {0, 1.02}},
    PlotStyle → Thickness[0.005], Frame -> True,
    FrameLabel → Map[Style[#, Larger] &, {"x", "P(high/X=x)"}],
    Axes → False, PlotLabel → Style[(propNames[[ind + 1]]), Larger],
    ImageSize → 600], {ind, Range[1, 7]}];
```

In[755]:=
```
Print[Magnify[#, 0.8]] & /@ nbcPlots[[1 ;; 4]];
```




LifeExpectancy
P(high/X=x)
1.0
0.8
0.6
0.4
0.2
0.0
50
60
70
80
90
100
x


MedianAge
P(high/X=x)
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
60
x


LiteracyFraction
P(high/X=x)
x
1.0
0.8
0.6
0.4
0.2
0.0
0.6
0.8
1.0
1.2
1.4

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 .