WEEK 10 HOMEWORK – SAMPLE SOLUTIONS

IMPORTANT NOTE

These homework solutions show multiple approaches and some optional extensions for most of the questions in the assignment. You don't need to submit all this in your assignments; they're included here just to help you learn more – because remember, the main goal of the homework assignments, and of the entire course, is to help you learn as much as you can, and develop your analytics skills as much as possible!

Question 14.1

The breast cancer data set `breast-cancer-wisconsin.data.txt` *from* *http://archive.ics.uci.edu/ml/machine-learning-databases/breast-cancer-wisconsin/ (description at http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Original%29) has missing values.*

1. *Use the mean/mode imputation method to impute values for the missing data.*
2. *Use regression to impute values for the missing data.*
3. *Use regression with perturbation to impute values for the missing data.*
4. *(Optional) Compare the results and quality of classification models (e.g., SVM, KNN) build using*
   (1) the data sets from questions 1,2,3;
   (2) the data that remains after data points with missing values are removed; and
   (3) the data set when a binary variable is introduced to indicate missing values.

Here's one possible solution. Please note that a good solution doesn't have to try all of the possibilities in the code; they're shown to help you learn, but they're not necessary.

The file `solution 14.1.R` shows one possible solution. In it, missing data is identified (only variable V7 has any, and it is only a small amount). Five different data sets are created to deal with the missing data:

(1) Replacing missing values with the mode. This could have gone either way (mode or mean). The data is categorical, but it takes integer values from 1 to 10, and as we'll see later the values seem to have some relative meaning, so they're also somewhat continuous.
(2) Using regression to estimate missing values. Here too could have gone either way (see above)… but since we didn't cover multinomial logistic regression in this course, the solutions treat the data as continuous for this part. Once the missing values are estimated, the estimates are rounded (because the original values are all integer) and values larger or smaller than the extremes are shrunk to the extremes.
(3) Using regression plus perturbation.
(4) Removing rows with missing data.

(5) Adding a binary variables to indicate when data is missing, and adding the necessary interaction variables also.

Once the data sets have been created, we use KNN (for k=1,2,3,4,5) and SVM (C=0.0001,0.001,0.01,0.1,1,10) to create classification models, and measure their quality. The table below shows the results.

	Method for dealing with missing data				
Model	Impute using mode	Impute using regression	Impute using regression, then perturb	Remove rows with missing data	Add binary variable for missing data
KNN (k=1)	0.952	0.948	0.948	0.952	0.952
KNN (k=2)	0.952	0.948	0.948	0.952	0.952
KNN (k=3)	0.924	0.919	0.919	0.923	0.924
KNN (k=4)	0.924	0.919	0.919	0.923	0.924
KNN (k=5)	0.919	0.914	0.914	0.913	0.919
SVM (C=0.0001)	0.662	0.662	0.662	0.659	0.662
SVM (C=0.001)	0.943	0.943	0.943	0.942	0.943
SVM (C=0.01)	0.957	0.957	0.957	0.957	0.957
SVM (C=0.1)	0.962	0.962	0.962	0.962	0.962
SVM (C=1)	0.967	0.962	0.967	0.966	0.967
SVM (C=10)	0.967	0.962	0.967	0.966	0.967

It turns out that there isn't much difference in model performance across the five ways of dealing with missing data. The best SVM models are a little better than the best KNN models, but SVM is harder to calibrate; the worst SVM models tested are much worse than the worst KNN models.

Question 15.1

Describe a situation or problem from your job, everyday life, current events, etc., for which optimization would be appropriate. What data would you need?

Some (perhaps overzealous!) baseball fans have tried to drive around the country to see a baseball game at each of the 30 Major League stadiums, and then return home, in the shortest number of days. This can be modeled using optimization: minimize the number of days it takes, subject to the constraints that a game is seen at each stadium, and the planned sequence of games is possible given the driving times and game schedules. The necessary data would include the Major League schedule (which stadiums have games scheduled on each day, and what time they're scheduled for), and how long it takes to drive between each pair of stadiums.

Question 15.2

In the videos, we saw the "diet problem". (The diet problem is one of the first large-scale optimization *problems to be studied in practice. Back in the 1930's and 40's, the Army wanted to meet the nutritional requirements of its soldiers while minimizing the cost.) In this homework you get to solve a diet problem with real data. The data is given in the file* `diet.xls`.

Here's one possible solution. <u>Please note that a good solution doesn't have to try all of the possibilities in the code; they're shown to help you learn, but they're not necessary</u>.

1. *Formulate an optimization model (a linear program) to find the cheapest diet that satisfies the maximum and minimum daily nutrition constraints, and solve it using PuLP. Turn in your code and the solution. (The optimal solution should be a diet of air-popped popcorn, poached eggs, oranges, raw iceberg lettuce, raw celery, and frozen broccoli. UGH!)*

Algebraically, here's what the model looks like:

<u>Data</u>

c_j = cost per unit of food j

$\boldsymbol{a_{ij}}$ = amount of nutrient $\boldsymbol{i}$ per unit of food $\boldsymbol{j}$

m_i = minimum amount of nutrient i required

$\boldsymbol{M_i}$ = maximum amount of nutrient $\boldsymbol{i}$ required

<u>Variables</u>

$\boldsymbol{x_j}$ = amount of food $\boldsymbol{j}$ eaten

<u>Objective</u>

Minimize $\sum_j \boldsymbol{c_j x_j}$ (minimize total cost)

<u>Constraints</u>

$\sum_j \boldsymbol{a_{ij} x_j} \geq \boldsymbol{m_i}$ for each nutrient $\boldsymbol{i}$ (take in at least minimum amount of each nutrient)

$\sum_j \boldsymbol{a_{ij} x_j} \leq \boldsymbol{M_i}$ for each nutrient $\boldsymbol{i}$ (take in no more than max amount of each nutrient)

$x_j \geq 0$ for each food j (can't eat negative amounts)

The file `solution 15.2-1-separate.py` shows one way to solve the problem. This file deals with each nutrient separately, to show what the problem looks like.

However, an easier way to do it is shown in file `solution 15.2-1-simpler.py`, where data is read and constraints are written in a loop, so individual nutrients don't need to each be written out. This will come in handy when there are more constraints (like the optional part below).

However you set up the problem, here's the optimal solution:

```
## ---------The solution to the diet problem is----------
## 52.64371 units of Celery,_Raw
## 0.25960653 units of Frozen_Broccoli
## 63.988506 units of Lettuce,Iceberg,Raw
## 2.2929389 units of Oranges
## 0.14184397 units of Poached_Eggs
## 13.869322 units of Popcorn,Air_Popped
##
## Total cost of food = $4.34
```

It’s a pretty cheap daily diet, but not one that looks particularly delicious!

2. *Please add to your model the following constraints (which might require adding more variables) and solve the new model:*

   a. *If a food is selected, then a minimum of 1/10 serving must be chosen. (Hint: now you will need two variables for each food i: whether it is chosen, and how much is part of the diet. You’ll also need to write a constraint to link them.)*

We need to add two new things to the model above:

<u>New variables</u>

$\boldsymbol{y_j}$ = 1 if food $\boldsymbol{j}$ is eaten, 0 if not (so it’s a binary variable)

<u>New constraints</u>

$x_j \geq 0.1\, y_j$ for each food j	(if food j is eaten, must eat at least 0.1 units)
0.0000001 $x_j \leq y_j$ for each food j	(if any of food j is eaten, must ensure $y_j = 1$)

   b. *Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.*

This requires just one constraint:

<u>New constraints</u>

$y_{frozen\ broccoli} + y_{raw\ celery} \leq 1$ (can’t eat both frozen broccoli and raw celery)

   c. *To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected. [If something is ambiguous (e.g., should bean-and-bacon soup be considered meat?), just call it whatever you think is appropriate – I want you to learn how to write this type of constraint, but I don’t really care whether we agree on how to classify foods!]*

This requires just one constraint, but it’s a longer one:

<u>New constraints</u>

$$\begin{aligned} y_{Roasted\ Chicken} &+ y_{Poached\ Eggs} + y_{Scrambled\ Eggs} + y_{Bologna,Turkey} + y_{Frankfurter,Beef} \\ &+ y_{Ham,Sliced,Extralean} + y_{Kielbasa,Prk} + y_{Pizza\ W/Pepperoni} + y_{Hamburger\ W/Toppings} \\ &+ y_{Hotdog,Plain} + y_{Pork} + y_{Sardines\ in\ Oil} + y_{White\ Tuna\ in\ Water} + y_{Chicknoodl\ Soup} \\ &+ y_{Splt\ Pea\&Hamsoup} + y_{Vegetbeef\ Soup} + y_{Neweng\ Clamchwd} + y_{New\ E\ Clamchwd,W/Mlk} \\ &+ y_{Beanbacn\ Soup,W/Watr} \geq 3 \end{aligned}$$

[Note: if you identified a different set of things that counted as meat/poultry/fish/eggs, that’s fine. I’m not worried about your ability to distinguish foods; I just want you to be able to write this constraint.]

The file `solution 15.2-2-simpler.py` shows one way to solve the problem. Here's the solution:

```
## ---------The solution to the diet problem is----------
## 0.1 units of Bologna,Turkey
## 42.423026 units of Celery,_Raw
## 82.673927 units of Lettuce,Iceberg,Raw
## 3.0856009 units of Oranges
## 1.9590978 units of Peanut_Butter
## 0.1 units of Poached_Eggs
## 13.214473 units of Popcorn,Air_Popped
## 0.1 units of Scrambled_Eggs
##
## Total cost of food = $4.51
```

It's pretty close to the previous solution, but the constraints we added have had their effect. Broccoli has disappeared (celery stayed), we have three proteins (turkey bologna, poached eggs, and scrambled eggs), and because the proteins are more-expensive, the solution uses the minimum amount of them that we allow (0.1 units).

If you want to see what a more full-sized problem would look like, try solving your models for the file `diet_large.xls`*, which is a low-cholesterol diet model (rather than minimizing cost, the goal is to minimize cholesterol intake). I don't know anyone who'd want to eat this diet – the optimal solution includes dried chrysanthemum garland, raw beluga whale flipper, freeze-dried parsley, etc. – which shows why it's necessary to add additional constraints beyond the basic ones we saw in the video!*

*[**Note**: there are many optimal solutions, all with zero cholesterol, so you might get a different one. It probably won't be much more appetizing than mine.]*

Algebraically, this model looks exactly like the one in Part 1 – there's no difference.

Data

c_j = cost per unit of food j

$\boldsymbol{a_{ij}}$ = amount of nutrient $\boldsymbol{i}$ per unit of food $\boldsymbol{j}$

m_i = minimum amount of nutrient i required

$\boldsymbol{M_i}$ = maximum amount of nutrient $\boldsymbol{i}$ required

Variables

$\boldsymbol{x_j}$ = amount of food $\boldsymbol{j}$ eaten

Objective

Minimize $\sum_j \boldsymbol{c_j x_j}$ (minimize total cost)

Constraints

$\sum_j \boldsymbol{a_{ij} x_j} \geq \boldsymbol{m_i}$ for each nutrient $\boldsymbol{i}$ (take in at least minimum amount of each nutrient)

$\sum_j \boldsymbol{a_{ij} x_j} \leq \boldsymbol{M_i}$ for each nutrient $\boldsymbol{i}$ (take in no more than max amount of each nutrient)

$x_j \geq 0$ for each food j (can't eat negative amounts)

The file `solution 15.2-large.py` shows one way to solve this problem. Note that the data file is less clean, so the code includes skipping the first empty row, replacing blank values with zero, etc. But other than that, we don't need to change much in the code from Part 1 to solve this much-larger problem; that's the benefit of having a good, general algebraic model in the code.

I wrote above that the optimal solution included some odd stuff, like dried chrysanthemum garland, raw beluga whale flipper, etc. It turns out that there's more than one optimal solution – both solutions (and others) are able to find a completely cholesterol-free diet. I got the solution above on a different machine using commercial-grade optimization software (CPLEX); using my laptop and PuLP, I got this solution:

```
## ---------One solution to the diet problem is----------
## 0.059863415 units of Beans,_adzuki,_mature_seeds,_raw
## 0.069514608 units of Broccoli_raab,_raw
## 0.42866218 units of Cocoa_mix,_no_sugar_added,_powder
## 0.14694398 units of Egg,_white,_dried,_flakes,_glucose_reduced
## 0.73805891 units of Infant_formula,_MEAD_JOHNSON,_ENFAMIL,_NUTRAMIGEN,_with_iron,_p
## 0.4258564 units of Infant_formula,_NESTLE,_GOOD_START_ESSENTIALS__SOY,__with_iron,
## 0.050114149 units of Infant_formula,_ROSS,_ISOMIL,_with_iron,_powder,_not_reconstitu
## 0.15033656 units of Margarine_like_spread,_approximately_60%_fat,_tub,_soybean_(hyd
## 0.25918767 units of Mung_beans,_mature_seeds,_raw
## 0.18052856 units of Nuts,_mixed_nuts,_dry_roasted,_with_peanuts,_with_salt_added
## 1.184482 units of Oil,_vegetable,_sunflower,_linoleic,_(hydrogenated)
## 0.10375187 units of Seeds,_sunflower_seed_kernels,_dry_roasted,_with_salt_added
## 0.031866196 units of Snacks,_potato_chips,_fat_free,_made_with_olestra
## 0.070710308 units of Spices,_paprika
## 0.55106575 units of Tomatoes,_sun_dried
## 9999.6864 units of Water,_bottled,_non_carbonated,_CALISTOGA
##
## Total cholesterol = 0.000000
```

It's also a pretty odd diet; in addition to eating three kinds of infant formula, you'd have to drink more water than a human body can hold!

I also attached the file `solution 15.2-protein.py`, to see what would happen if, using the large data set, we wanted to maximize protein intake rather than minimize cholesterol. Here too, the diet looks pretty weird: lots of water, three types of beluga whale meat, and a frozen dinner!

```
## ---------One solution to the diet problem is----------
## 7.0117007 units of BANQUET_Salisbury_Steak_Meal,_Gravy_and_Salisbury_Steak_with_Ma
## 0.20365743 units of Cereals_ready_to_eat,_KASHI_Heart_to_Heart_by_KELLOGG
## 0.23412086 units of Collards,_raw
## 25.855235 units of Fish,_devilfish,_meat_(Alaska_Native)
## 31.46708 units of Fish,_lingcod,_meat,_raw_(Alaska_Native)
## 0.02 units of Fish_oil,_cod_liver
## 2.2140307 units of Gelatins,_dry_powder,_unsweetened
## 0.037489833 units of Mollusks,_oyster,_eastern,_canned
## 57.437865 units of Rhubarb,_wild,_leaves_(Alaska_Native)
## 621.79859 units of Sweeteners,_tabletop,_aspartame,_EQUAL,_packets
## 9.5089609 units of Tea,_brewed,_prepared_with_distilled_water
## 9552.2849 units of Water,_bottled,_non_carbonated,_CALISTOGA
## 276.5536 units of Water,_bottled,_non_carbonated,_DANNON
## 0.076732592 units of Whale,_beluga,_flipper,_raw_(Alaska_Native)
## 9.6405544 units of Whale,_beluga,_liver,_raw_(Alaska_Native)
## 1.7353546 units of Whale,_beluga,_meat,_air_dried,_raw_(Alaska_Native)
##
## Total protein = 2994.899576
```

IMPORTANT: These odd solutions might make you wonder whether optimization solutions are really useful/realistic. But that actually is looking in the wrong direction. The problem with the solutions in this homework is that the *model* is too simplistic. Obviously, none of us eats this way; if we wanted to find a realistic high-protein diet, for example, we'd have to write more-complicated constraints – we wouldn't allow more water than we can drink, we'd require that the food be easily obtainable (so, probably no beluga whale meat), we'd add constraints for our personal taste, for price, for variety of diet, etc. As the model would get more and more realistic, the solution would too. We didn't see much detail about that in this course, because we didn't have time; modeling is probably at least 1/3 of the Georgia Tech's elective course on optimization.