Ordered Sets in Data Analysis Big Homework: Predicting University Tuition High Or Low

Name: Ho Lum Cheung
Email: khchun_1@edu.hse.ru
Due Date: 12 DEC 2019

Prompt: Use Lazy FCA to examine a dataset and come to some conclusions

1. Analysis Software

Python 2.7.14 with pandas, numpy, and datetime, and random

2. Background

The Integrated Postsecondary Education Data System (IPEDS) data set (https://www.kaggle.com/sumithb university-data-ipeds-dataset) contains American university enrollment data with rows representing universities and columns representing enrollment features such as the number of enrolled students (size). University enrollment data is useful for answering many questions for many groups. We provide the following examples:

1. A student might ask: Given my test scores, what universities will accept me?
2. A researcher might ask: Where do Asians like to go to school?
3. A university might ask: Should we adjust our admission standards?

We propose the following problem:

Researchers are interested in tools to determine which universities have good value for tuition costs. For instance, they work for a university ranking agency and want to write articles about good deals for high school students to consider. And so they do a small project (this big homework) on FCA to answer the following preliminary question:

Using FCA and basic demographic information, how well can we predict if a university charges 'high' or 'low' tuition?

3. Data Set

The original data set has 145 features, but we remove most of them due to being confounding, incomplete, or irrelevant.

Examples of confounding data include university name, zip code, state, and location data.

Examples of incomplete data include racial makeup of students.

Examples of irrelevant data include about 10 columns of binary variables specifying what degrees (associate's bachelor's, master's, PHD) the university offers.

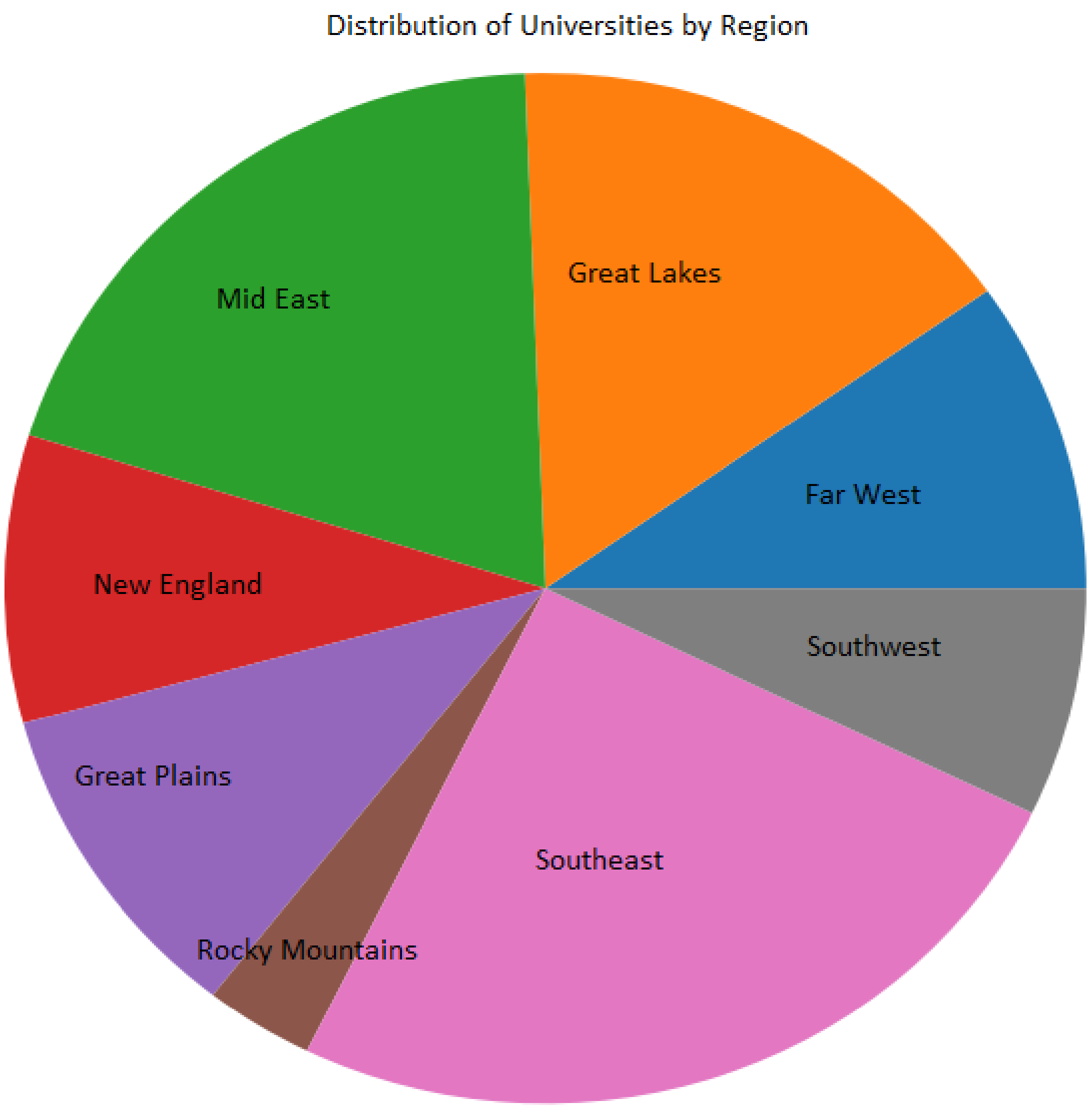
Target feature: Total price for out-of-state students living on campus 2013-14

4. Cleaning and Descriptive Statistics

The original data set has 1534 objects. We simply dropped any university with any missing data for any of our features, leaving 1326 objects (and 7 features).

Feature	1QR	Median	3QR
Percent Admitted	54	66	77
Undergraduate Enrollment	1476.75	2687.5	6844.75
Financial Aid	87	96	99
Pell Grant	26	37	48
Price (OOS)	31842	37902	46323

Feature	Public(Yes)	Private(No)
Public	850	476



5. Scaling

All of our features (numeric, categorical) required scaling. In general, to keep the weight of each feature the same in the case that the size of the intersect matters, we scaled each feature to 2-3 new binary attributes. After scaling, we have about 20 attributes on 5 features. In general, we scaled our numerical data **ordinally**, and our categorical data to better categories when dealing with tuition. (For example, we know that New England schools usually charge high tuition) We list our attributes below:

Public	School is public
Private	School is private
Very Selective Admissions	$<$ 40% of students are admitted
Selective Admissions	$<$ 65% of students are admitted
Regular Admissions	$\geq$ 65% of students are admitted
New England	School is in the New England region
West Coast	School is in the Far West region
Inland	School is in the Rocky Mountain, Great Lakes, or Great Plains regions
South and Southwest	School is in the South(east) or Southwest
Small Enrollment	$<$1000 undergraduate students enrolled
Regular Enrollment	$\geq$ 1000 undergraduate students enrolled
Low Aid	$<$ 90% of students are on financial aid
Regular Aid	$\geq$ 90% of students are on financial aid
Low Pell	$<$ 30% of students have Pell Grants
Medium Pell	$<$ 45% of students have Pell Grants
High Pell	$\geq$ 45% of students have Pell Grants
Low Price	Average Tuition for Out-of-state students living on campus is $<$\$38,000

6. Cross-Validation

We do a random split of our data 10 times with 30% being test data and 70% being training data. We end up with 929 training cases and 399 test cases for each split.

7. FCA Algorithm

In general, we translated into code the task prescribed by Fedor Strok and kept the infrastructure as provided with the following changes:
We remove unnecessary functions for this task.
Because we use different scoring, we remove threshold testing.
We report on all of our test files at once.
We rewrote infrastructure to make intersections and track common intersections.
It may happen in our code that we labeled intersections and hypotheses interchangeably. This final report should always use the term 'intersection' to reduce ambiguity.
We add our own scoring system.

Intersection Scoring

We have simple intersection scoring and more advanced scoring. If the intersection is valid (more support than counter-support), then we give one vote to the respective score (positive or negative). This is regardless of the length of the intersection or other factors. For advanced scoring, we considered taking the square of the intersection length and requiring intersection to be at least a length of n=4 (or 5,6 etc.) This provides approximately 3% improvement in accuracy (77 to 80).

Overall Scoring

For the overall voting, we simply sum scores for positive and negative contexts, divide by the number of examples, and see which support is higher.

Scoring: Mathematical Notation

We present the following mathematical explanation for clarification of our scoring: Let $h_{\tau,n}$ be the intersection of the test case g_τ with an example from G_+,G_-:

$$h_{\tau,n} = g'_n \cap g'_\tau \text{ where } g_n \in G_+ \text{or } g_n \in G_- \tag{1}$$

And let $l_{\tau,n}$ be the length of the intersection:

$$l_{\tau,n} = |h_{\tau,n}| \tag{2}$$

And let $v_{\tau,n}$ be the validity of this intersection, or the support in the contexts:

$$v_{\tau,n} = \frac{\sum_{i \in G_+} \mathbb{1}_{(h_{\tau,n} \subseteq i)}}{|G_+|} - \frac{\sum_{i \in G_-} \mathbb{1}_{(h_{\tau,n} \subseteq i)}}{|G_-|} \tag{3}$$

We will end up using this validity/weakness function only for whether it is positive or negative, but we provide it because it's another point where a researcher could modify things.

Next, our simple scoring for one element can be represented as some product of indicator functions. So over all the examples in K_+:

$$S_{\tau,+} = \frac{\sum_{g_n \in G_+} \mathbb{1}_{(l_{\tau,n} > 0)} * \mathbb{1}_{(v_{\tau,n} > 0)}}{|G_+|} \tag{4}$$

$$S_{\tau,-} = \frac{\sum_{g_n \in G_-} \mathbb{1}_{(l_{\tau,n} > 0)} * \mathbb{1}_{(v_{\tau,n} < 0)}}{|G_-|} \tag{5}$$

Note that we divided by the number of elements to in unbalanced data (we are calculating support). More advanced scoring also requires an indicator function, but it factors in minimum intersection length and the actual length of the hypothesis (squared). In this equation, minlen is the researcher-specified length for which we will accept the intersection as meaningful and score it:

$$S_{\tau,+} = \frac{\sum_{g_n \in G_+} l_{\tau,n}^2 \mathbb{1}_{(l_{\tau,n} \geq minlen)} * \mathbb{1}_{(v_{\tau,n} > 0)}}{|G_+|} \tag{6}$$

$$S_{\tau,-} = \frac{\sum_{g_n \in G_+} l_{\tau,n}^2 \mathbb{1}_{(l_{\tau,n} \geq minlen)} * \mathbb{1}_{(v_{\tau,n} < 0)}}{|G_+|} \tag{7}$$

Formal result classification is as follows:

$$\text{Classify } g_\tau as \begin{cases} Undefined : S_{\tau,+} = S_{-,\tau,-} = 0 \\ Contradictory : S_{\tau,+} - S_{\tau,-} = 0, S_{\tau,+} > 0 \\ Positive : S_{\tau,+} - S_{\tau,-} > 0 \\ Negative : S_{\tau,+} - S_{\tau,-} < 0 \end{cases} \tag{8}$$

A note about maximizing accuracy vs. honesty

While removing or modifying some of these factors (especially not considering support) will improve the accuracy of this particular data set, it will not make the algorithm useful in general.

Other Modifications

We added some timing analysis code, modified code to remove the column headers (the source of 10 'contradictions'), and some other minor things. Lastly, we note that "contradictory"scores from the original code are not contradictions, but rather "unknown"or "unclassifiable". But we only change the end-user display so as to rewrite minimal code.

7. Results

With positive results being that tuition is < \$38,000, and n = minimum hypothesis length:

	Simple	n ≥ 4	n ≥ 6	n ≥ 8	**n ≥ 12**	n ≥ 16
True Positive	1353	1432	1432	1459	**1540**	1479
True Negative	1703	1674	1675	1671	**1638**	1561
False Positive	295	324	323	327	**360**	361
False Negative	629	550	550	523	**442**	414
Accuracy	**0.768**	**0.780**	**0.781**	**0.786**	**0.798**	**0.797**
True Positive Rate (Sensitivity, Recall)	0.683	0.723	0.723	0.736	**0.777**	0.781
True Negative Rate (Specificity)	0.852	0.838	0.838	0.836	**0.820**	0.812
Positive Predictive Value (Precision)	0.821	0.815	0.816	0.817	**0.811**	0.804
Negative Predictive Value	0.730	0.753	0.753	0.762	**0.788**	0.790
False Positive Rate	0.148	0.162	0.162	0.164	**0.180**	0.188
False Negative Rate	0.317	0.277	0.277	0.264	**0.223**	0.219
False Discovery Rate	0.179	0.185	0.184	0.183	**0.189**	0.196
F1 Score	0.745	0.766	0.766	0.774	**0.793**	0.792

Results for Tic-tac-toe data

We are not quite sure what the tic-tac-toe data is for, but we ran our code against it with the following results (TTT $\geq$ n means a minimum hypothesis length was required):

	TTT Simple	TTT $\geq$ 6	TTT $\geq$ 7
True Positive	477	540	613
True Negative	272	328	332
False Positive	60	4	0
False Negative	149	86	13
Accuracy	0.782	0.906	**0.986**
True Positive Rate (Sensitivity, Recall)	0.762	0.863	0.979
True Negative Rate (Specificity)	0.819	0.988	1.000
Positive Predictive Value (Precision)	0.888	0.993	1.000
Negative Predictive Value	0.646	0.792	0.962
False Positive Rate	0.181	0.012	0.000
False Negative Rate	0.238	0.137	0.021
False Discovery Rate	0.112	0.007	0.000

(We note that nothing was classified when setting minimum hypothesis length to be $\geq$ 8) It appears that considering hypothesis of exactly length 7 is extremely useful (we had 98.6% accuracy). We further note in the appendix that the initial evaluation code is biased and should only have achieved 60% accuracy.

8. Conclusion

We are able to predict with about 80% accuracy, whether a particular school's tuition will be above or below $38,000.
While having a minimum hypothesis length improves our accuracy, tuning it seems to give lackluster results. Requiring it to be too high will also cause us to have un-classifiable data. Also, because we chose to split near the median, our accuracy is not as high as possible, but other indicators of predictive value are reasonable. Our predictor is about as accurate as the random forest predictor, so our results are expected.

9. Final Remarks

If we were to improve on this project, we propose the following 3 tasks:
1. Preprocess and process additional features related to student test scores.
2. Modify the process to predict a tuition fee rather than do a binary classification task.
3. Improve run time with comparison and hypothesis-checking enhancements.

Appendix

A final note on 'bias' in the original code

We believe the original code attempts to do something similar (perhaps with a simpler comparison) to our code. However, we point out that the code has bugs and 'cheats'. We provide code comments as to how this is the case, but also claim the following:
The main bug is that we think the variable 't' in 'test_ impl.py' is some quasi-hypothesis made from the example, but it does not look like a hypothesis but rather a subset of [o,x,b,'positive','negative'].
The main 'cheating' is that this when this 't' contains 'positive' or 'negative', counterexamples will never be found and examples are way more likely to be found.
We demonstrate this fact by discarding (removing) 'positive' and 'negative' from t. We cannot think of a scenario where it should be in this set. Rerunning the algorithm gives an approximately 60% accuracy.

A manual look at Pell data

Supposing we only used the "Low Pell"split, we would arrive at 69.5% accuracy:
True Positive = (Low Pell, High Price)

	Low Pell	High Pell
High Price	339	319
Low Price	86	582

Accuracy = .695, Sensitivity = 0.515, Specificity = 0.871

A manual look at region data

Region	Low Split	High Split	Predictive Value
Far West	45	78	0.634
Great Lakes	91	110	0.547
Mid East	94	170	0.644
New England	32	93	0.744
Plains	91	48	0.655
Rocky Mountains	23	13	0.639
Southeast	223	122	0.646
Southwest	69	24	0.742

One way to interpret this is that given a school is in New England, we would be able to guess with 74.4% accuracy that tuition costs 38,000 or more

Sanity Check: Random Forest Classifier

Using a very naive random forest classifier, we have accuracy of about 80%, which informs us that our methods are probably reasonable.

True Positive	1589
True Negative	1578
False Positive	404
False Negative	409
Accuracy	0.796
True Positive Rate (Sensitivity, Recall)	0.795
True Negative Rate (Specificity)	0.796
Positive Predictive Value (Precision)	0.797
Negative Predictive Value	0.794
False Positive Rate	0.204
False Negative Rate	0.205
False Discovery Rate	0.203

Sanity Check: Excluding 'Useful' Features

Excluding region and financial aid (Pell, especially) data leads to accuracy rates around 55%. A quick glance at the data indicates that those are our best indicators.

A quick reference relating states to regions

```
region_SE = 'Southeast AL AR FL GA KY LA MS NC SC TN VA WV'
region_W = 'Far West AK CA HI NV OR WA'
region_SW = 'Southwest AZ NM OK TX'
region_MNT = 'Rocky Mountains CO ID MT UT WY'
region_NE = 'New England CT ME MA NH RI VT'
region_ME = 'Mid East DE DC MD NJ NY PA'
region_MW = 'Great Lakes IL IN MI OH WI'
region_GP = 'Plains IA KS MN MO NE ND SD'
```

Run-time Analysis

Runs are timed with an update every 100 test cases examined. Currently this takes about 15 seconds. A full run presented in the final commit takes approximately 15 minutes.