Course 5, Task 3: Data Science and Big Data

Lessons Learned Report

Anders Dowd

Sometimes Simpler is Better

On the WiFi locationing project, the POA called for testing at least three algorithms and suggested a list that included the very same algorithms required for this task. I ended up trying 12 different algorithms and the best performance came from Random Forest. On this project, I stuck to just the four required algorithms and again Random Forest was the most effective for both models. I'm glad I took the initiative to experiment with other algorithms since I am still so new to machine learning and data analytics, but this project reinforced the notion that for a lot of tasks, a straight forward, simple algorithm is probably sufficient to do the job. I know there will be instances where a highly specialized algorithm is needed, but one thing that has been consistent throughout this entire program is that Random Forest is a pretty good catch-all machine learning algorithm for classification and should probably be a baseline reference for most classification tasks I encounter in the future.

Model	Accuracy (Train)	Kappa (Train)	Accuracy (Test)	Kappa (Test)
C5.0	0.7700645	0.5523425	0.7753213	0.5619559
Random Forest	0.7733677	0.5597573	0.7760925	0.5654067
K-Nearest Neighbors	0.3648004	0.1793220	0.3609254	0.1817561
Support Vector Machines	0.7084131	0.4062715	0.7151671	0.4162251

Table 1: Comparison of Modeling Algorithm Performance (iPhone)

Model	Accuracy (Train)	Kappa (Train)	Accuracy (Test)	Kappa (Test)
C5.0	0.7651510	0.5281260	0.7659520	0.5303090
Random Forest	0.7679725	0.5341508	0.7698269	0.5375984
K-Nearest Neighbors	0.7564675	0.5145780	0.7563937	0.51410720
Support Vector Machines	0.7120026	0.3929717	0.7036941	0.3772725

Table 2: Comparison of Modeling Algorithm Performance (Galaxy)

Feature Selection Improves Run Time If Not Performance

After experimentation with correlation, Near Zero Variance, and Recursive Feature Elimination, I proceeded with RFE to reduce the size of my data sets for training my models. Running RFE resulted in a reduction from 58 variables to 25 for iPhone and from 58 to 26 for Galaxy. While it did not improve the accuracy or Kappa score of my models by a significant amount, it greatly sped up the run time of my model training without losing any accuracy/Kappa.

iPhone Features			Galaxy Features		
iphone	iphonedisneg	iphonecamneg	iphone	htccampos	htcperneg
googleandroid	iphoneperunc	ios	samsunggalaxy	htcperpos	samsungcamunc
samsunggalaxy	iphoneperneg	htccamunc	htcphone	samsungperpos	samsungcampos
htcphone	htcdisneg	samsungperpos	googleandroid	iphoneperunc	iphonecamneg
iphonedispos	htcperneg	iphonecampos	sonyxperia	iphonedisneg	iphonecampos
sonyxperia	htcdispos	htcperunc	iphonedispos	iphoneperneg	htccamneg
iphoneperpos	htcperpos	googleperpos	iphonedisunc	htcperunc	iosperneg
iphonedisunc	htccamneg	iphonecamunc	iphoneperpos	ios	samsungdisunc
htccampos			htcdispos	htccamunc	

Table 3: Features Selected through Recursive Feature Elimination

When It Comes to Classification, Less Is More

One of the very first truths that became apparent to me in this program was that classification models are most effective at predicting binary variables. While it's not always possible to reduce your target to two classes, any simplification of the dependent variable can improve model performance. In this task, I was able to reduce the number of classes from 6 to 4 and it had a dramatic improvement on classification performance. For both data sets, I saw an improvement of nearly 8 percentage points in accuracy. This provided me with enough wiggle room to make other adjustments to the data sets that would improve other metrics at the expense of accuracy while producing more useful results, namely...

Method	Accuracy (Train)	Kappa (Train)	Accuracy (Test)	Kappa (Test)
Near Zero Variance	0.7581757	0.5248325	0.7640103	0.5378806
Recursive Feature Elimination	0.7732021	0.5601608	0.7773779	0.5683583
RFE w/ Undersampling	0.4593214	0.3511832	N/A	N/A
RFE w/ Oversampling	0.5614980	0.4737977	N/A	N/A
RFE w/ Recoded Dependent	0.8510435	0.6321705	0.8473008	0.6208318
RFE w/ Rec. Dep. & Oversamp.	0.6656846	0.5542470	0.6673784	0.5565045

Table 4: Results of Feature Selection and Engineering Methods with Random Forest Classifier for iPhone

Method	Accuracy (Train)	Kappa (Train)	Accuracy (Test)	Kappa (Test)
Near Zero Variance	0.7556376	0.5042898	0.756652	0.5062507
Recursive Feature Elimination	0.7683591	0.5360192	0.7687936	0.5359937
RFE w/ Undersampling	0.4529992	0.3435224	N/A	N/A
RFE w/ Oversampling	0.5433774	0.4520528	N/A	N/A
RFE w/ Recoded Dependent	0.8451899	0.6018153	0.8470679	0.603060
RFE w/ Rec. Dep. & Oversamp.	0.6644438	0.5525917	0.6540550	0.538740

Table 5: Results of Feature Selection and Engineering Methods with Random Forest Classifier for Galaxy

As In All Things, Balance Is Important

…balancing the dependent. In both original training data sets, there was a heavy over representation of positive sentiment. As a result, when I trained on the data sets out of the box, the models were inevitably biased toward positive at the expense of other classes. Now, because most of the instances in the training sets were, in fact, positive sentiment, the models still returned high accuracy even though they were terrible at predicting 3 out of 4 classes. In order to correct for this, I experimented with both undersampling and oversampling the data sets to balance the classes of the dependent variable. Oversampling proved to be the more effective strategy for this project. The resulting models still overpredicted positive sentiment, but not nearly so egregiously as before.

iPhone RFE w/ Recoded Dependent

Prediction \ Reference	0	1	2	3
0	376	1	2	9
1	0	15	0	1
2	7	2	231	9
3	205	235	123	2674

Accuracy: 0.8473 Kappa: 0.6208

Statistics By Class

	0	1	2	3
Sensitivity	0.6395	0.0593	0.6489	0.9929
Specificity	0.9964	0.9997	0.9949	0.5297
Balanced Accuracy	0.8179	0.5295	0.8219	0.7613

iPhone RFE w/ Recoded Dependent Oversampled

Prediction \ Reference	0	1	2	3
0	1958	8	8	72
1	191	1271	201	570
2	14	7	1955	46
3	530	1407	529	2005

Accuracy: 0.6674 Kappa: 0.5565

Statistics By Class

	0	1	2	3
Sensitivity	0.7271	0.4720	0.7260	0.7445
Specificity	0.9891	0.8809	0.9917	0.6948
Balanced Accuracy	0.8581	0.6764	0.8588	0.7196

Galaxy RFE w/ Recoded Dependent

Prediction \ Reference	0	1	2	3
0	346	2	1	34
1	1	16	1	3
2	3	3	200	26
3	158	228	150	2699

Accuracy: 0.8424 Kappa: 0.5906

Statistics By Class

	0	1	2	3
Sensitivity	0.6811	0.0643	0.5682	0.9772
Specificity	0.9890	0.9986	0.9909	0.5167
Balanced Accuracy	0.8351	0.5314	0.7795	0.7469

Galaxy RFE w/ Recoded Dependent Oversampled

Prediction \ Reference	0	1	2	3
0	2078	25	19	53
1	153	1220	219	527
2	21	32	1836	90
3	510	1485	688	2092

Accuracy: 0.6541 Kappa: 0.5387

Statistics By Class

	0	1	2	3
Sensitivity	0.7524	0.4417	0.6647	0.7574
Specificity	0.9883	0.8915	0.9827	0.6762
Balanced Accuracy	0.8703	0.6666	0.8237	0.7168

Table 6: Comparison of Model Metrics With and Without Oversampling Employed

Accuracy Isn't Everything

As I mentioned above, when I recoded the dependent to reduce the number of classes to predict, I was able to improve the accuracy of my model to roughly 85%, but the ability to reliably predict anything other than positive sentiment was abysmal. I addressed this issue by balancing the dependent variable in my training model via oversampling. Table 6 breaks down the metrics of the models with and without oversampling. The oversampled model demonstrates an increase in True Positive Rate for classes 0-2 as well as an increase in the True Negative Rate for class 3, resulting in a much more balanced model. While the overall accuracy of each model hovers around a pedestrian 65-66%, the predictive value of the oversampled model is, in my view, much greater than the model with the unbalanced target. This is particularly true for negative sentiment, which, for Helio's purposes, is every bit as important as positive sentiment. While accuracy in the 80s sure looks pretty, at the end of the day the less overall accurate model produced more useful results for our purposes.