Eric_Hirsch_622_Final_Assignment

Predicting the Space Titanic Kaggle Competition

Eric Hirsch

12/11/2022

Contents

```
knitr::opts_chunk$set(echo = TRUE, warning =  FALSE, message = FALSE)
```

```
dfTable <- read.csv("D:\\Rstudio\\Cuny_622\\Space\\622a.csv")
hux <- as_hux(dfTable)
```

Discussion

Introduction

In machine learning, we predict target variables based on input variables. For this final exercise, we will apply various machine learning algorithms to a Kaggle data set (Spaceship Titanic) in order to predict which passengers have been transported to another dimension.

While it's tempting to throw as many algorithms at the problem as possible to see what sticks, the statistical fact is that while it is rare that a poor model will perform well on a holdout set, the chances of making false conclusions based on performance increases if we simply try one model after another. Besides, if we don't understand our model and our data, and the model becomes much more difficult to troubleshoot and maintain.

When choosing models, we are balancing simplicity and complexity, and therefore tendencies to underfit or overfit. When the relationships in the data are simple and certain statistical conditions are met, parametric methods like OLS work well and have the advantage of being easily interpretable. If, for example, we are predicting height from weight, the relationship is simple enough that we can create a linear regression model and capture most of the variation that can be explained for these two variables.

When we increase our dimensions and/or complexity of relationships within the dataset, parametric methods are likely to underfit the data. Even in our simple height and weight example, if the relationship between height and weight varies considerably at lower weights, medium weights and higher weights, spline regression or another nonparametric technique will be necessary. As dimensions and complexity increases, we adopt techniques that are more powerful at morphing the data shape so that we can model the underlying structure, such as trees, SVM and neural nets.

Choosing the more complex algorithm will likely fit the training data better, but may be less interpretable and more subject to overfitting. With this in mind, each of these techniques has its advantages and disadvantages. In my experience with earlier datasets in this class, trees will pick up autonomous clusters in the data set better than SVMs. For example, if there were a small but significant anomalous cluster of individuals for whom height and weight were inversely related, trees will incorporate the cluster while SVMs will ignore it. Of course, clusters like this might signal a missing variable, but not all of the necessary variables will be found in any given data set. On the other hand, when the relationships are more systematic and class boundaries are clear, SVMs may perform better because the kernel trick allows SVMs to radically change the data shape in order to find the class boundary. SVMs can also perform better when there is less data.

While decision trees are powerful, they are generally more so when bagged (e.g., Random Forest) or boosted (e.g. xgBoost). Because xgBoost is an active learner, it will often have the upper hand in fitting the training data.

One of the biggest advantages of neural networks is that they effectively do the feature engineering for you if you can apply enough layers. They are also subject to the "double descent" phenomenon, which helps with managing underfitting. However, for a student using a home computer like myself, it's often impractical to take advantage of these facts as the algorithm would run too long. Neural networks, like SVMs, also powerfully change the data shape in order to find class boundaries.

Accurate prediction depends not only on algorithm choice. We also need to engineer features (except possibly in very large neural nets) and tune hyperparameters. We also need to choose metrics that tell us whether or not our model is effective.

Prediction using the Kaggle Spaceship Titanic Data Set

For this exercise I've chosen a Kaggle Competition – the Kaggle Spaceship data set. The advantages of using this a competition data set are that we can compare our performance to those of others. Achieving 90% on a holdout set in and of itself tells us nothing - we don't know if achieving 95% would have been easy or impossible. In this competition, the 2,000 or so submitted accuracies on the leaderboard range from about 76% to 82%, which can give us a good idea of how well our model is working.

The main disadvantages of this data set are that the data is made up and the scenario a bit far-fetched. However, I wanted a data set that had a simple class as a target, as opposed to an image example. The standard Titanic data set has been over analyzed, so this was one of the few good choices left.

The Business Problem

In the year 2912, the Spaceship Titanic, an interstellar passenger liner with almost 13,000 passengers on board, collided with a spacetime anomaly hidden within a dust cloud. Though the ship stayed intact, almost half of the passengers were transported to an alternate dimension. Our job is to predict which passengers were transported by the anomaly using a set of partial records recovered from the spaceship's damaged computer system.

Data Summary and Preparation

The data set consists of 8693 records and 13 variables, including spending on the ship's various amenities (VR Deck, Spa, Room Service, Food Court, and Shopping Mall), cabin number, whether the individual was traveling with the group, whether the individual was a VIP, planet of origin and destination, and so on. These columns map to some degree with the original Titanic database.

The target variable, Transported, is roughly equally distributed between false (4315) and true (4378).

Some of the numeric variables, particularly spending variables, are highly skewed - most passengers spend no money while a few spend a great deal. We can see that spending on luxuries (the spa, room service, etc.) is strongly negatively correlated with being transported - this supports the supposition that the rich were spared. Spending on more budget–friandly amenities like the food court and shopping mall are also negatively correlated but less so. Age has a small negative correlation as well.

We decide not to log transform the numeric variables as normal distributions for predictors are not required by our models and interpretability suffers.

Missing Values 1073, or 12%, of records have missing values. The vast majority of missing values are found in the amenity expenditure columns. Oddly, the amenity expenditure rows with missing values are completely independent of each other - there are almost no records where more than one of these values is missing.

This may be an artifact of the fact that the data is manufactured. In order to confirm that there is no systematic relationship between missing data and the target variable, we look at the Chi square between the target and a flag designating missing data. We do this for each amenity expenditure column and find no relationship between missing data and the target variable. We therefore eliminate rows with missing values for the training set. For the test set, we impute the median.

There is very little, even surprisingly little, multicollinearity in the database. In the case of variables that track spending on amenities this is most surprising, and may suggest that passengers were working within a budget and only spent money on the activities they liked most.

Outliers All of the spending variables are highly skewed, with very large instances occurring at the very end of the distribution. However, as most of our techniques are robust for outliers, records with extreme values remain in the database, as there is no reason to think that the spending is a data entry error or an anomalous occurrence.

Feature Engineering The data set holds a number of opportunities for feature engineering. Through testing, it was found that the following new features were significant in predicting transportation. They are:

1. Groups - Passenger ID numbers suggest that some passengers boarded the ship as part of a group. Although we could not establish that members of a group tended to meet the same fate, being a member of a group influenced whether a passenger was transported.
2. Cabins - Cabin codes were parsed for location n the ship - this information was correlated with the target variable.
3. Interaction variables - there were a number of interactions among variables which appeared when the variables were examined in isolation. We only retained one (group passengers were more likely to be transported if they shopped at the mall) because the others did not significantly increase accuracy when running the overall model - but there would be more to explore here.

At this point, a picture of the transported passengers begins to emerge - they are the poorer passengers, most likely to shop on a budget or, even cheaper, spend the trip in cryosleep. They tend to enter the ship in groups and inhabit lower class cabins.

Modelling

Choosing and Testing Models Understanding what models are doing and how is a key part of prediction. We compared tree models, svm, neural net, and logistic regression.

The first task is to understand the requirements of the business problem. In this case, we have a partial roster of passengers where we know who was transported and who wasn't. As for the rest of the passengers, it is our job to predict which of them were transported as well. Insight into the data is necessary insofar as it helps us make better predictions, but we need no more from the data than that. Accuracy is our metric, as this is the metric used in the Kaggle (e.g., there is no penalty for false positives or false negatives which might suggest a different metric). This suggests we should use the most complex model we can that has the highest accuracy.

We used tenfold cross validation and compared models on the metric of accuracy – our results varied widely. There was some, but not perfect, match between the accuracy predicted by the cross validation, and the accuracy achieved in the Kaggle. For example, going on cross validation alone, the svm with the radial kernel underperformed compared to a linear and polynomial kernel - however, in the Kaggle it significantly outperformed both kernels. Of course, without the Kaggle we would not have known this and would not have chosen the radial kernel to put into production.

On the other hand, our best model on cross validation (xgboost) also performed best on the Kaggle. We suspect the neural net would have performed well also, but we did not have the computer power to hypertune it properly.

Hyperparameter Tuning The caret package in R automatically hypertunes models on basic parameters and chooses the most optimal based on the metric determined by the user. With the exception of xgBoost, all of our models were tuned that way. This left two tasks – tune xgBoost, and experiment with manual tuning of other models.

The tuned xgBoost model (number of rounds was reduce from 55 to 14) increased accuracy from 79% to 80%, and improved our percentile position in the Kaggle from the 73rd to the 78th percentile.

The fact that our radial-kernal SVM had lower cross validation accuracy but higher Kaggle accuracy might have a number of different causes - but one is the possibility that there are idiosyncrasies in the training set that are not found in the Kaggle set. In this case, we can increase sigma to decrease the risk of overfitting.

As we experimented by increasing sigma and decreasing sigma on the radial SVM model. Reducing sigma (tighter fit) improved accuracy during cross validation, but not when applied to the Kaggle. Because the differences were very small (.0026 improvement in accuracy), the models probably only varied by a handful of predictions. Increasing sigma (looser fit) reduced accuracy both in cross validation and on the Kaggle set.

Results

```
knitr::kable(dfTable)
```

Model	Holdout.Set.Accuracy	Kaggle.Accuracy	Kaggle.Percentile
XGBoost Tuned (14 Rounds)	0.8000	0.80383	78.2%
XGBoost Untuned (55 Rounds)	0.7900	0.80243	73.8%

Model	Holdout.Set.Accuracy	Kaggle.Accuracy	Kaggle.Percentile
SVM Rad	0.7715	0.80149	70.0%
SVM Rad Lower Sigma	0.7741	0.80032	61.1%
SVM Rad Higher Sigma	0.7669	0.79682	57.3%
SVM Linear	0.7800	0.79775	56.8%
SVM Poly	0.7800	0.79611	55.3%
Random Forest	0.7900	0.78793	34.5%
Logistic Regression with feature engineering	0.7800	0.77975	26.2%
Neural Net (severely limited due to computer power)	0.7600	0.77881	25.5%
Logistic Regression without feature engineering	0.7800	0.69227	9.5%