Supervised Learning: Classification and Regression

Titanic Passenger List

Ronan McHugh s162927

November 7, 2016

1 Introduction

In this paper I present the outcome of a regression problem and a classification problem using the dataset **Titanic Passenger List** as provided by Kaggle.com[1]. For the regression problem I chose to predict *Fare*, while for the classification problem I chose to predict *Survival*. All machine learning algorithms are from Scikit-learn [7], while the graphs have been coded in `matplotlib.pyplot`.

1.1 Feature engineering and data transformations

For this assignment I decided to improve the dataset through feature engineering. Following the example of Risdal[2] and Besbes[9], I created two new features, *Title* and *FamilySize*. In addition, I performed data cleaning on the *Fare* attribute and the *Age* attribute. Note that the code for all these transformations can be seen in the `titanic_data.py` script in the GitHub repo for the project [3].

Title *Title* was created by extracting passenger title from the *Name* attribute. The *Name* attribute is well structured and always contains a title of some sort. These titles can be indicative of a passenger's age, sex, marital status, profession and social status and as such can be assumed to be of relevance in determining their fare (in the case of regression). Creating the attribute was a simple matter of identifying all possible values and parsing these out from the text of the name attribute.

FamilySize *FamilySize* was created by combining the *Parch* and *SibSp* attributes for each passenger. This gave a count for the total number of family members travelling with the passenger.

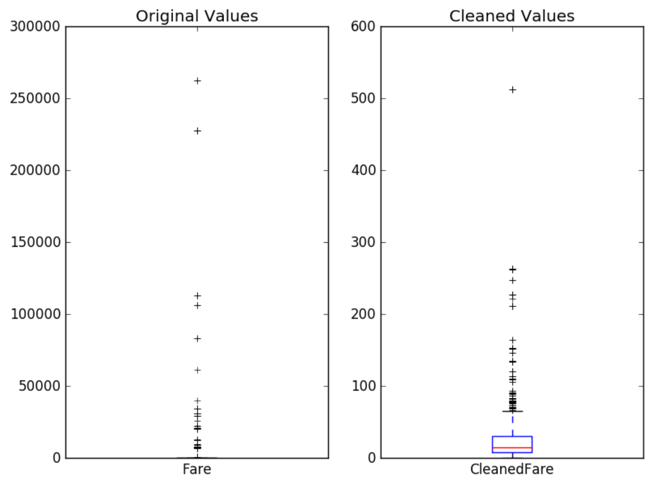

Figure 1: Distribution of Fare values before and after data cleaning. As can be seen, the cleaning process greatly reduces the range of the values in keeping with the historical data.

Fare As discussed in our previous report, the *Fare* attribute is extremely noisy[4]. After examining the values in the dataset and comparing them to information on actual ticket prices, I determined that many of the prices in the dataset were simply missing a decimal separator, for example, a passenger's fare might have the value 7925, when the actual value should be 7.925 [5][6]. Once I realised this, it was trivial to write a transformation to add the decimal point in the correct location before transforming the attribute to a numeric datatype. The outcome of this step is visualised in Figure 1.

Age There are approx 177 observations that are missing the value for *Age*. Since the engineered attribute *Title* has some relevance to *Age*, I decided to use the mean for that observation's title to fill in the *Age* variable. This was preferable to dropping the attribute as I presumed it might have some significance for both problems.

2 Regression models

2.1 Target problem

I chose to predict *Fare* for two reasons: firstly, it is the only continuous attribute. There are other interval attributes such as *Parch* (number of Parents or Children) and *SibSp* (number of siblings) but these are discrete, i.e. they increase in intervals of one. Obviously we could round the results of a linear regression to return discrete values, but I preferred not to add this layer of processing for simplicity's sake. Secondly I reasoned that *Fare* would be easier to predict based on other attributes such as *Pclass*

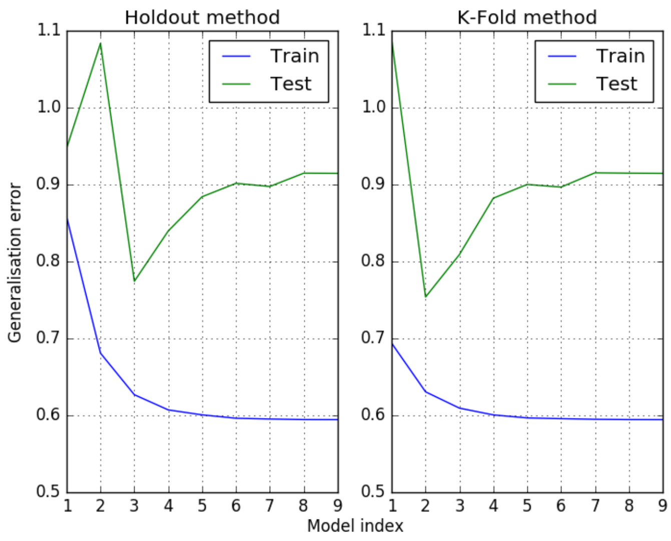

Figure 2: Two layer cross validation of a linear regression model selected using *Holdout* and *K-Fold* methods.

and *Embarked* than the other attributes mentioned.

2.2 Estimation method

Before attempting to solve the regression problems, I wanted to compare the value of the two different cross validation methods, *K-Fold* and *Holdout*. To do this, I used forward selection to choose the model's predictor features, first using *Holdout* to partition the dataset once, then using *K-Fold* to test the model against *K=3* different partitions, choosing a model based on the best average score. I then estimated the generalisation error by by testing the same models against a new section of the dataset that had not been tested. The results are shown in Figure 2 [1]. The differences between the train and test errors in both figures demonstrates the effect of overfitting, as the generalisation error on the test set quickly increases as the model becomes more complex. Interestingly, the second feature chosen when using the forward select algorithm with *Holdout* increased the generalisation error dramatically.

2.3 Linear Regression Coefficients

Using two-layer cross validation, the optimal model for linear regression used only *Pclass* and *FamilySize*. The coefficients of the learned model were *[-34.41, 7.89]*, respectively

[1]Note that in this instance and all other graphs I have computed the error rate as $1 - R^2$ where R^2 represents the coefficient of determination [8]. The best possible value for R^2 is 1 and values can be negative. Thus in our graphs the y scale ranges from 0 to above 1, where 0 represents flawless prediction.

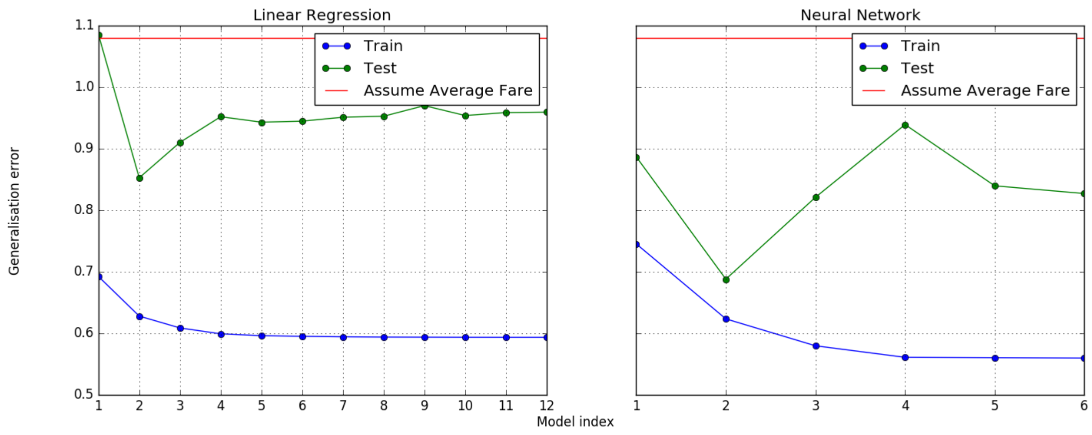

Figure 3: Two layer cross validation comparison of linear regression and artificial neural network.

while the intercept is *105.05*. This fits our intuition of the result set; *Pclass* has a very strong negative relationship as a high *Pclass* (i.e. a 3rd class ticket) will cost much less than a low one. Similarly, travelling with family will naturally cost more if the ticket includes their fares. Using an example containing normalised values for *Pclass* and *FamilySize* we can walk through the model formally as follows:

$$
\begin{aligned}
x_i &= [0.8264756, 0.06972867] \\
y &= b_0 + (b_1 * x_1) + (b_2 * x_2) \\
y_i &= 105.05 + (-34.41 * .82) + (7.89 * .069) \\
y_i &= 77.16
\end{aligned}
$$

2.4 Linear Regression vs Artificial Neural Network

My next step was to test the linear model against an artificial neural network, the outcome of this is shown in Figure 3, with the error rate caused by assuming the average fare marked in red. In terms of model selection we can see that the simpler models performed better in the generalisation estimate, as shown in Table 1. The best performing model in both cases used *Pclass* and *FamilySize* attributes but the neural network achieved greater accuracy with these features than the linear regression model did. The linear regression model also developed more elaborate models that were shown by cross-validation to significantly overfit the training data 3.

Model index	**Features**	**Training Error**	**Estimated Generalisation Error**
-	**Linear Regression**	-	-
1	Pclass	0.6923	1.0853
2	Pclass, FamilySize	0.6280	0.8526
3	Pclass, FamilySize, EmbC	0.6087	0.9102
4	Pclass, FamilySize, EmbC, TitMiss	0.599	0.9519
5	Pclass, FamilySize, EmbC, TitMiss, TitMr	0.5961	0.9431
6	Pclass, FamilySize, EmbC, TitMiss, TitMr, EmbQ	0.5951	0.9447
7	Pclass, FamilySize, EmbC, TitMiss, TitMr, EmbQ, Sex	0.5943	0.9511
8	Pclass, FamilySize, EmbC, TitMiss, TitMr, EmbQ, Sex, TitMajor	0.5938	0.9527
9	Pclass, FamilySize, EmbC, TitMiss, TitMr,EmbQ, Sex, TitMajor, Age	0.5936	0.9699
10	Pclass, FamilySize, EmbC, TitMiss, TitMr, EmbQ, Sex,Tit Major, Age, Survived	0.5935	0.9539
11	Pclass, FamilySize, EmbC, TitMiss, TitMr, EmbQ, Sex, TitMajor, Age, Survived, TitDon	0.5934	0.9587
12	Pclass, FamilySize, EmbC, TitMiss, TitMr, EmbQ, Sex, TitMajor, Age, Survived, TitDon, TitCol	0.5934	0.9594
-	**Neural Network**	-	-
1	Pclass	0.7454	0.8868
2	Pclass, FamilySize	0.6234	0.6880
3	Pclass, FamilySize, EmbC	0.5797	0.8210
4	Pclass, FamilySize, EmbC, TitMiss	0.5609	0.939
5	Pclass, FamilySize, EmbC, TitMiss, TitMme	0.5603	0.8396
6	Pclass, FamilySize, EmbC, TitMiss, TitMme, TitMajor	0.5597	0.8271

Table 1: Comparison of model selection by linear regression and artificial neural networks

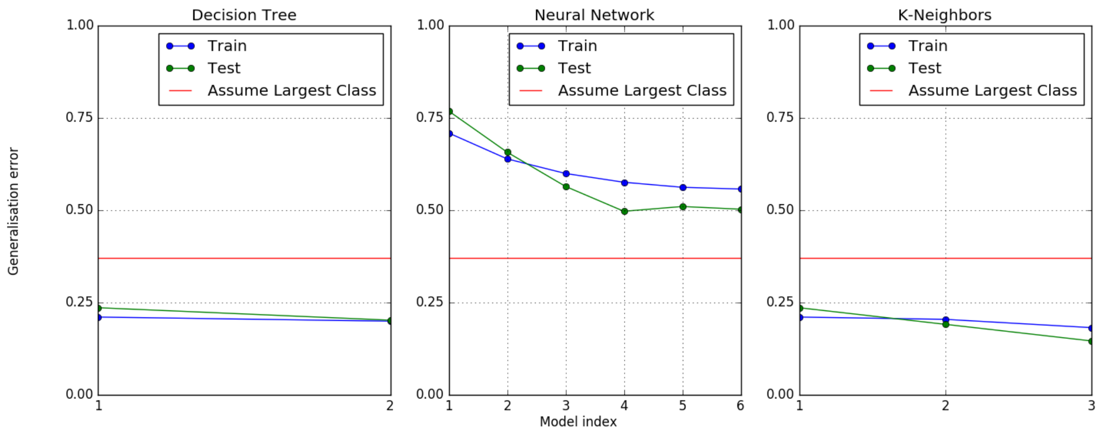

Figure 4: Two layer cross validation comparison of classification models

3 Classification models

3.1 Target Problem

For the classification exercise I have chosen to predict the *Survived* attribute. This is the intended problem of the Titanic dataset and as such I have based my feature engineering decisions on the literature dealing with the problem [2] [9].

3.2 Comparison of classifiers

I chose to predict the outcome using Decision Trees, K-Nearest Neighbours and Artificial Neural Networks. I used forward selection to choose the optimal models, scoring first using K-Fold cross validation before testing them against a test dataset to estimate the generalisation error. The results of this process are summarised in Figure 4 and Table 2. The estimated generalisation error resulting from assuming all values to be equal to the largest class is shown in red. As can be seen, the performance on the classification problem is significantly better than that of the regression problem. In all cases, the gap between the test error and the training error is much smaller and in some cases the models actually perform better on the test set. The two best performing method are the Decision Tree and the K-Neighbours, the Neural Network performs significantly worse than presuming the largest target class.

3.3 Interpreting the models

As it is not possible to observe the inner workings of neural networks, I will only look at the best models returned by the Decision Tree method and by the K-Nearest Neighbours method.

Features	**Training Error**	**Estimated Generalisation Error**
Decision Tree	-	-
Sex	0.2107	0.2359
Sex, FamilySize	0.1995	0.2022
Neural Network	-	-
Sex	0.7121	0.7839
Sex,Pclass	0.6367	0.6673
Sex, Pclass, TitMaster	0.6080	0.5862
Sex, Pclass, TitMaster, FamilySize	0.5701	0.4581
Sex, Pclass, TitMaster, FamilySize, TitMlle	0.5607	0.5707
K-Neighbours	-	-
Sex	0.2107	0.2359
Sex, FamilySize	0.2045	0.1910
Sex, FamilySize,TitMaster	0.1820	0.1460

Table 2: Performance of classification methods

3.3.1 Decision Tree

The default implementation of the Decision Tree method provided by Scikit-learn uses a Gini function to measure split quality and has no depth limits. When applied to our dataset, this returns a highly complex tree with 18 levels in some places, thus for brevity's sake I only show 4 layers in Figure 5. Given a male passenger travelling with 1 family member, we can walk through the Decision Tree as follows:

$$x_i = [0, 1] \tag{1}$$
$$Sex <= 0.5 = True \tag{2}$$
$$FamilySize <= 0.5 = False \tag{3}$$
$$FamilySize <= 3.5 = True \tag{4}$$
$$FamilySize <= 1.5 = True \tag{5}$$
$$y_i = Perished \tag{6}$$

As we can see, the Decision Tree predicts that this passenger perished in the disaster, correctly in this case. The visualisation of the Decision Tree shows that the main group of survivors are women with a relatively small number of family members.

3.3.2 K-Neighbours

We will need to write some code to explain the workings of the K-Neighbours algorithm. Given $x_i = [1, 3, 0]$, i.e. a woman travelling with three family members, with a title other than Master:

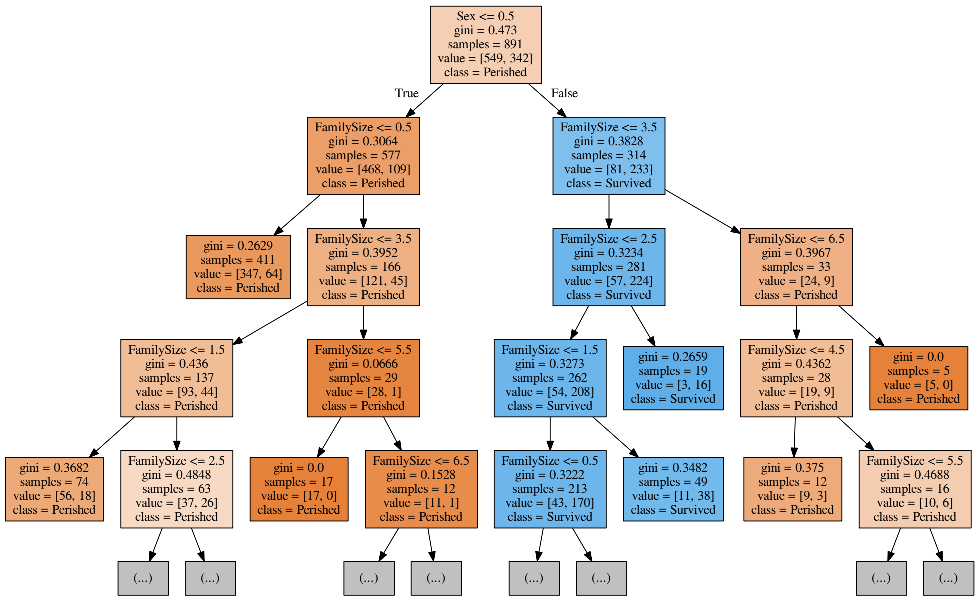

Figure 5: Visualisation of Decision Tree based on Sex and FamilySize

```
# ...
# we assume the relevant imports and data creation
clf = neighbors.KNeighborsClassifier(5, weights='distance')
clf.fit(X_train, y_train)
new = array([[ 1.,  3.,  0.]]) # our new sample
# Returns a distance array followed by an array indexes for the new sample's kneighbours
clf.kneighbors(new)
# (array([[ 0.,  0.,  0.,  0.,  0.]]), array([[498,  43, 448, 297, 726]]))
# Let's lookup those neighbors
X_train[498], y_train[498] # (array([1, 3, 0]), 0)
X_train[43], y_train[43]   # (array([1, 3, 0]), 1)
X_train[448], y_train[448] # (array([1, 3, 0]), 1)
X_train[297], y_train[297] # (array([1, 3, 0]), 0)
X_train[726], y_train[726] # (array([1, 3, 0]), 1)

# Get the prediction
clf.predict(new) # array([1])
```

As we can see, the model finds five neighbours with a distance of 0, i.e. they have the exact same attributes as the new observation. To predict the class, the classifier simply takes the majority class of the nearest neighbours, in this case, 1 for Survived, which is incorrect for this observation.

3.4 Comparison with other work

Risdal [2] performs more elaborate engineering (she creates a Mother and a Child variable for example) and uses a Random Forest algorithm to get an overall error rate of slightly under 20%. Besbes also performs substantial engineering before employing a Random Forest algorithm to return an error rate of 0.1866 [9]. Note though that both Besbes and Risdal derive their error rates from a different test set[2].

References

[1] Kaggle, *Titanic: Machine Learning From Disaster*, `https://www.kaggle.com/c/titanic/`

[2] Megan Risdal, *Exploring the Titanic Dataset*, `https://www.kaggle.io/svf/198371/166ea2e9c1074ca9cd2447c7ee27cf10/__results__.html`

[3] Ronan McHugh, *Machine Learning*, `https://github.com/ronan-mch/machine_learning`

[4] Ali Chegini, Ronan McHugh, *Feature Extraction and Visualisation - Introduction to Machine Learning - Project 1*, `https://docs.google.com/document/d/1QSV2kJBotRCUvSiBF7-Z3Jjz6BoKB2H9TaV4PGRheTk/edit?usp=sharing`

[5] Reference.com, *What were the prices of Titanic Tickets in 1912?*, `https://www.reference.com/history/were-prices-titanic-tickets-1912-53fecb79d8634272`

[6] Encylopedia Titanica, *Titanic Passenger List*, `https://www.encyclopedia-titanica.org/titanic-passenger-list/`

[7] Scikit-learn: Machine Learning in Python Pedregosa et al, JMLR 12, pp. 2825-2830, 2011 `http://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html`

[8] Scikit-Learn: R2 Score, the coefficient of determination scikit-learn, `http://scikit-learn.org/stable/modules/model_evaluation.html#r2-score-the-coefficient-of-determination`

[9] How to score 0.8134 in Titanic Kaggle Competition Ahmed Besbes, 10-08-2016, `http://ahmedbesbes.com/how-to-score-08134-in-titanic-kaggle-challenge.html`

[2]They use Kaggle's online test set and testing tool which I have avoided as it it cannot easily be documented or integrated into my code