PERFORMANCE COMPARISON BETWEEN LOGISTIC REGRESSION, RANDOM FOREST CLASSIFIER AND CONVOLUTIONAL NEURAL NETWORK WITH TITANIC DATA SET

Author: Tien Tran

MATH 564 – Applied Statistics

Fall 2018

1. Introduction

   Machine learning has been around for a very long time. But only until recently, with the appearance of many massive online learning websites, machine learning has become much more accessible for a lot of people. With a certain knowledge of programming, calculus and linear algebra, a person can acquire basic knowledge about machine learning in a relatively short time. And with the growing of Big Data, there are more and more courses about Deep Learning, with various libraries made so that doing machine learning and deep learning become as convenient as possible.

   But as we have more and more techniques to use, and more data to train our models; do we tend to ignore simpler techniques? This seems to be a common mistake among newbie machine learning practitioners. From my own experience, I usually get very excited with complicated techniques or architectures that I learn and want to apply them right into the next problem. In such situation, too often I find the result from my model underwhelming. And since the model is too complex, it is hard for me to troubleshoot and fix the problem.

   In this study, I want to try different approaches to one single problem: the Titanic data set. This is a small data set used to practice machine learning. It is very popular on Kaggle and I play around with it quite a lot. My goal is to find out the strengths and weaknesses of each approach so that I can have a better understanding of each method and will be able to apply them better in the future.

2. Problem statement

   The problem appears in the Kaggle online competition "Titanic: Machine Learning from Disaster". Here is the competition link: https://www.kaggle.com/c/titanic

   According to the competition description: "The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew."

   The description also states that: "Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class."

   In this challenge, we are given a training data with nearly 900 labeled rows. Rows labels 1 mean that the passenger survived the shipwreck, while rows labels 0 mean that the passenger did not survive. We are asked to complete the analysis of what sorts of people were likely to survive.

   For evaluation, the metrics used is accuracy (the percentage of correct predictions out of all predictions made).

3. Methodology

   Looking at the data, we see that it includes training data (train.csv) and test data (test.csv) with the training data has ground truth (labels indicating passenger survived or not). Detailed analysis on the data will be conducted on the next part.

As we can see, the response variable (Y, or 'Survived' in this case) can only take value of 0 or 1. Since Y is a binary response variable, it is natural to think of Logistic Regression to create our prediction model.

Another model choice is Random Forest Classifier, since it is natural for Decision Tree to make split at each feature (predictor variable).

Finally, I want to apply a Convolutional Neural Network to this problem to see if it can learn the parameter to predict the survivability of passengers in the test data.

In order to equally compare these three models, I use the similar treatment for the data before I fit them into the models. I will use a little feature engineering to create new features for this data. For example, passenger name will be engineered to get the title of that person, e.g. Sir, Madam, or Dr. (Data featuring is inspired by a Kernel of Sina: https://www.kaggle.com/sinakhorami/titanic-best-working-classifier)

For categorical data with less than 7 levels, I use one-hot-encoding (or dummies variables) to represent them in the model. I also drop some "irrelevant" columns in the data such as PassengerId, Name (after extracted the Title), and the Cabin number. Continuous variables will be standardized before used in the model. All the details can be found in the attached codes.

I use the fastai library to conveniently preprocess the data. Shout out to Jeremy Howard and his partners for making such a great library for machine learning in general and deep learning in particular.

4. Analysis & Results

    a. Data analysis

```
In [5]: train.head()
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
In [6]: train.shape, test.shape
Out[6]: ((891, 12), (418, 11))
```

Figure 1. First look at the data

From Figure 1 we can see what the predictor variables are. We can also see the dimension of the train and test table.

Analysis on response variable Y:

```
In [8]: train['Survived'].value_counts()

Out[8]: 0    549
        1    342
        Name: Survived, dtype: int64

In [9]: sns.countplot(x='Survived', data=train, palette='hls')
        plt.show()
```

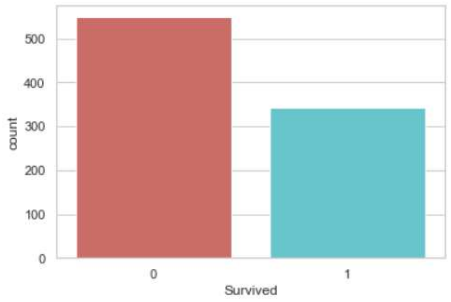

Figure 2. Distribution of Y

The distribution of response variable Y (value 0 and 1) are not too skewed, which is good.

```
In [10]: train.groupby('Survived').mean()
```

Out[10]:

Survived	PassengerId	Pclass	Age	SibSp	Parch	Fare
0	447.016393	2.531876	30.626179	0.553734	0.329690	22.117887
1	444.368421	1.950292	28.343690	0.473684	0.464912	48.395408

Figure 3. Mean of continuous data grouped by 'Survived'

From Figure 3 we can see that survived passengers tend to be younger, and also have higher fares.

Next we'll look at some distribution of some predictor variables to see whether they are good predictor of the response variable. They are "Passenger class", "Gender", and "Age group":

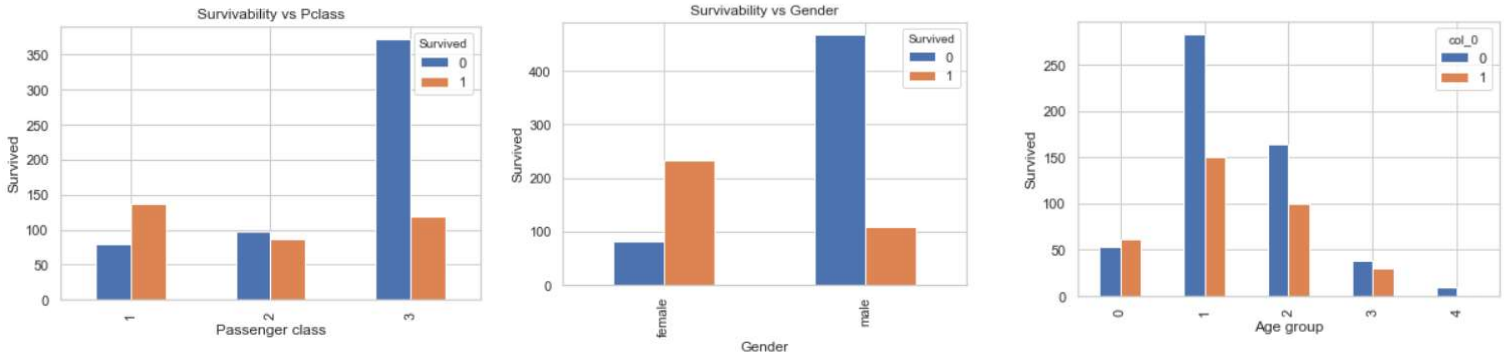

Figure 4. Distribution of Y against Pclass, Gender, and Age group

From the distribution in Figure 4, Gender and Age group can make good predictors because the distribution of survivability among different levels are quite separate from one another. Meanwhile, Passenger class may not make as good predictor because the class 1 and 2 have quite similar distribution.

b. Model results

The following result table is taken from submission score on the competition page:

	Logistic Regression	Random Forest Classifier	Fastai's default DL model
Test data accuracy (%)	0.77990	0.78947	0.77033

Some notes and comments:

- Logistic Regression was carried out using default parameters.
- Random Forest Classifier needs some tunings to get the score of 0.78947. To get that score I need to change the number of estimator (trees in the forest) to be 80 (default number is 10). I also change the max_features parameter to 0.5 so that for each decision tree, it only consider half of the features instead of all to avoid overfitting.
- Contrary to my initial thoughts, the last model does not perform well on the test set. I see similar problems on data sets with small amount of training data (several thousands). So this method may work better if we have much more data (say, at least hundred thousand samples).

c. Feature selection and model analysis

Machine learning libraries offer many convenient tools for practitioners to do feature selection. In this paper I experimented with 2 tools among them. The first is Recursive Feature Elimination (RFE) from sklearn library. The second is Random Forest Feature Importance from the fastai library.

- sklearn's RFE:

Susan Li, author of the article "Building a Logistic Regression in Python, Step by Step", wrote: "Recursive Feature Elimination (RFE) is based on the idea to repeatedly construct a model and choose either the best or worst performing feature, setting the feature aside and then repeating the process with the rest of the features. This process is applied until all features in the dataset are exhausted. The goal of RFE is to select features by recursively considering smaller and smaller sets of features." This idea is like backward stepwise model selection. However, since sklearn does not provide a similar method, we'll use RFE for this problem.

After the feature engineering process, my training set has 18 predictor variables. I used RFE to reduce the number of variables to 8 to see whether that helps create a better model.

RFE helps me choose 8 best predictors out of 18. They are: ['Pclass', 'Age', 'SibSp', 'Title', 'Has_Cabin', 'Sex_female', 'Sex_male', 'Embarked_C']. I then create a reduced logistic regression model with those 8 predictor variables and make predictions on the test set.

However, this time the score on the test set is only 0.77511, which is not an improvement of the full model.

- fastai's Feature Importance:

```
In [10]: # Feature importance
         fi = rf_feat_importance(m, x); fi[:10]
```

Out[10]:

	cols	imp
11	Sex_male	0.184560
10	Sex_female	0.144180
19	Age*Class	0.091317
6	Ticket	0.082780
18	Fare_Per_Person	0.082400
7	Fare	0.063425
2	Name	0.063419
0	PassengerId	0.054728
3	Age	0.052950
1	Pclass	0.051714

Figure 5. Feature Importance values (in descending order)

A couple of plots to better illustrate those values:

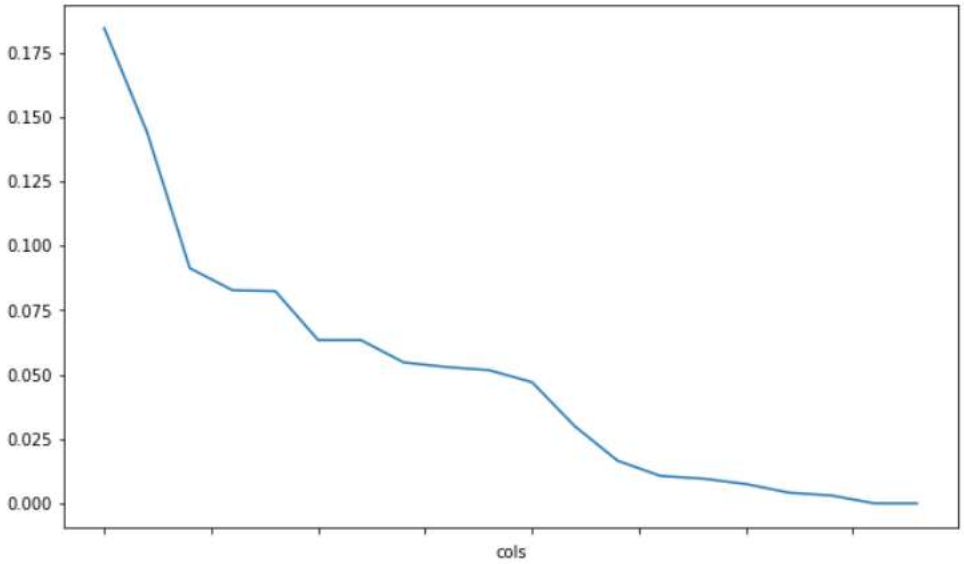

Figure 6a. Line chart of feature importance

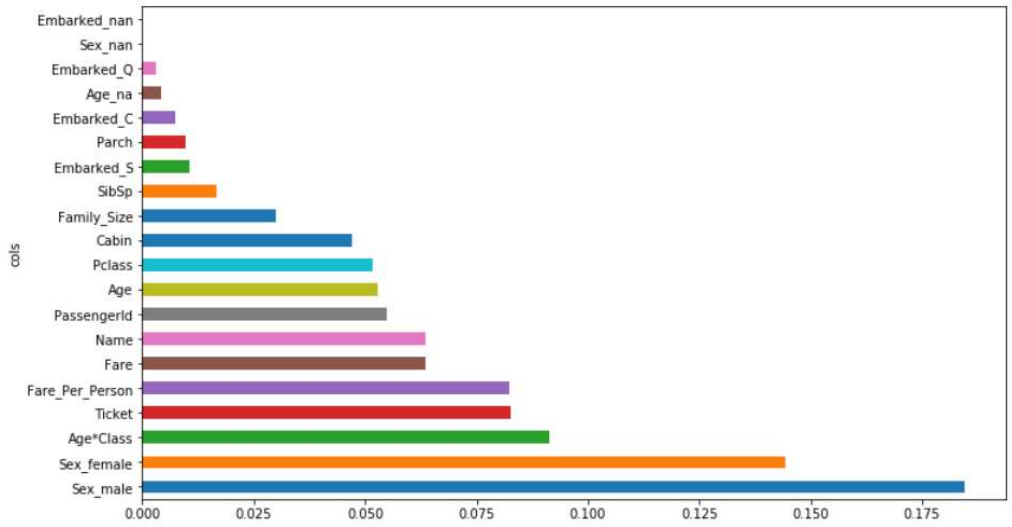

Figure 6b. Bar chart of Feature importance

From this information we can see that the Decision Trees in Random Forest Classifier use the gender information as one of the most important things to decide on the outcome of the label (Survived or not). This is quite intuitive and it follows our findings on the data analysis part. Age*Class is also a feature used frequently to decide the label of the data.

In the next experiment, I also deleted some features (columns) that have relatively low importance, according to the above values and charts. However, just as in the RFE case, the score on the test data decreases (0.78468 compared to 0.78947).

- Note that in this example, the number of data sample (nearly 900 data points) is quite large in comparison with the number of columns (or predictor variables). So creating more features seem to work better than reducing number of features. More features mean that the model can have more thing to "learn" from the training data and thus can provide more accurate prediction.

5. Conclusion

- Simpler algorithms do not mean worse algorithms. As we can see from the results presented above, a logistic regression with no tuning perform quite well on the test set, and only slightly worse than a carefully tuned Random Forest Classifier. In cases when the data is not very large (like in many data sets or playground competitions on Kaggle), simpler approach can provide very good results.
- Simpler methods such as Logistic Regression (or linear models in general) and Decision Tree ensemble can be used to interpret the data effectively. They will be very helpful if we want to gain more insights about the problems, not just get the best prediction results possible.
- About Deep Learning: one of my favorite quotes is Andrew Ng's: "AI is akin to building a rocket ship. You need a huge engine and a lot of fuel. The rocket engine is the learning algorithms but the fuel is the huge amounts of data we can feed to these algorithms."

- This is true when we have to build a deep learning model from scratch. In addition to a good architecture, we need a lot of data for the model to learn in order for it to be effective in making predictions. However, in real life this limit can be overcome if we use transferal learning. For many problems we can use a pretrained model and train last layer of the convolutional neural network and can come up with very good predicting model for a small amount of training data.
- What can be done in future studies? Hypothesis testing for goodness-of-fit; Model selection with AIC, BIC; and model diagnostics (e.g. detection of influential observations) are some things that I can think of. One major difficulty for me in conducting this study is the discrepancy between different platforms. R has many interesting statistical tools and procedures, while Python's libraries are very convenient to apply many machine learning and deep learning methods. There is a Python library (RPy2) to serve as an interface between both languages. But due to the time limit for this study, I haven't implemented RPy2 successfully.
- Final thoughts: This course, and this project has inspired me to dig deeper into the theory and logic behind many machine learning models' implementations. This will help me acquire better understanding of machine learning methods, and apply them with higher efficiency in the future.

APPENDIX

1. Titanic competition on Kaggle: https://www.kaggle.com/c/titanic
2. “Building a Logistic Regression in Python, Step by Step”, Susan Li, Towards Data Science: https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becd4d56c9c8
3. “Titanic best working Classifier”, Sina, Kaggle kernel: https://www.kaggle.com/sinakhorami/titanic-best-working-classifier
4. Introduction to Machine Learning for Coders, fastai: https://course.fast.ai/ml
5. Practical Deep Learning for Coders, part 1, fastai: https://course.fast.ai/
6. Scikit-learn, Machine Learning in Python: https://scikit-learn.org/stable/