MIDTERM
TITANIC: MACHINE LEARNING FROM DISASTER

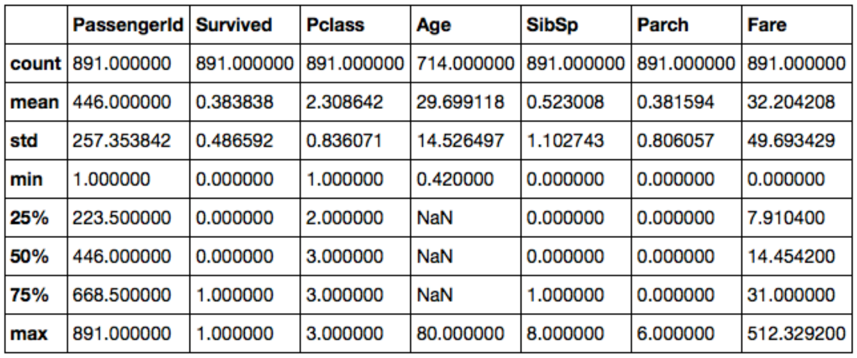
1 Introduction

This task is about using machine learning algorithm to data of passengers in Titanic to predict whether a passenger could or could not survive in titanic disaster. From the data, we may find that some groups of people had more possibility of surviving than the others. Mostly, to finish this task, steps including data exploration, feature engineering, model building, model training are introduced in the report. Two models, K nearest neighbor and random forest are applied to predict the result, results with or without feature selection are compared and discussed. From the result, approaches with feature selection have better performance.

2 Methods and Approach

DATA EXPLORATION

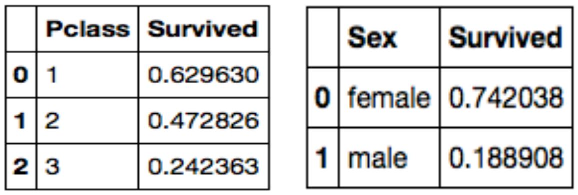
Before building models to predict for data set, first we can do data exploration to have a rough understanding of the data, to see which kind of data we have and find whether the features have some relationship, which will be an assistant to our feature selection and feature engineering. We can find that the label of our output can be represented as '0' or '1', so this problem can be considered as a classification problem.
Then using python with numpy and pandas, we can have a rough view of the data set. Using training set as example. There are some missing data in data set, we may use regression models to fill these missing, but here I only consider filling the NaN with mean value. The statistic information of training set is used when doing feature selection and feature engineering.



	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	NaN	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	NaN	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	NaN	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

Figure 1 Training Data

For the categorical or the numerical features, we can analyze each feature respectively. For example, we can easily find that passengers whose sex if female are more likely to be survived. So this might a useful feature for our classification.



	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

	Sex	Survived
0	female	0.742038
1	male	0.188908

Figure 2 Pcalss and Sex

	SibSp	Survived
1	1	0.535885
2	2	0.464286
0	0	0.345395
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

	Parch	Survived
3	3	0.600000
1	1	0.550847
2	2	0.500000
0	0	0.343658
5	5	0.200000
4	4	0.000000
6	6	0.000000

	age_band	Survived
0	[0, 7.91]	0.197309
1	(7.91, 14.454]	0.303571
2	(14.454, 31]	0.454955
3	(31, 512.329]	0.581081

	fare_band	Survived
0	[0, 7.91]	0.197309
1	(7.91, 14.454]	0.303571
2	(14.454, 31]	0.454955
3	(31, 512.329]	0.581081

Figure 3 SibSp, Parch, Age, Fare

For baseline models, using original value of the feature, changing the string features to dummy code, filling the missing data with that feature's mean value.

DATA ENGINEERING

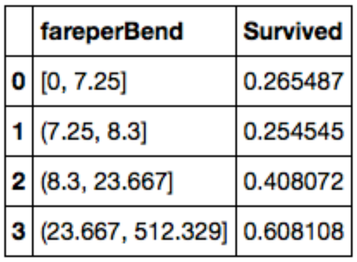
Date engineering includes feature construction, feature extraction and feature selection. For our data set, some features are not as clear as others. Such as name and tickets. For parch and sibsp, we can get a new feature of passenger's family size, and whether they are alone or not. We can separate each and see the distribution of labels. Observing that the title in passenger's name have some similarity, it might be a useful feature as well. The feature extraction and analyzing are as follows. From fare and family size, we can calculate a new feature of fare per person in that family. From the figure, the possibility of survive doesn't distribute normally for these new features. So the features may be utilized for our classification. Using sequential feature selection method, each time we add one feature, see whether the result is better or not, we may find out the most useful features for our task. Also, as we observed, features like age, fare, title, ticket length, family size, fare per person has can be turned to dummy code as their distribution differ in ranges. For example, for age, we can set age_dummy as 0 when age is in the range of (0, 8), and 1 for age in (8, 15). Here for each feature with large range, I change it to dummy value like 0, 1, 2, 3 based on 25%, 50%, 75% value of that feature from Data exploration part.

	IsAlone	Survived
0	0	0.505650
1	1	0.303538

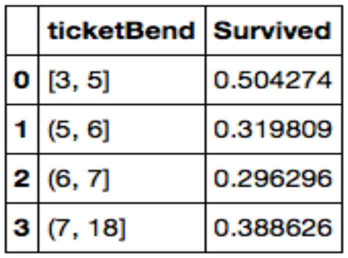
	FamilySize	Survived
3	4	0.724138
2	3	0.578431
1	2	0.552795
6	7	0.333333
0	1	0.303538
4	5	0.200000
5	6	0.136364
7	8	0.000000
8	11	0.000000

	Title	Survived
0	Master	0.575000
1	Miss	0.702703
2	Mr	0.156673
3	Mrs	0.793651
4	Other	0.347826

Figure 4 isAlone, familysize, title



	fareperBend	Survived
0	[0, 7.25]	0.265487
1	(7.25, 8.3]	0.254545
2	(8.3, 23.667]	0.408072
3	(23.667, 512.329]	0.608108



	ticketBend	Survived
0	[3, 5]	0.504274
1	(5, 6]	0.319809
2	(6, 7]	0.296296
3	(7, 18]	0.388626

Figure 5 fareperperson, ticketlength

K NEAREST NEIGHBORS

Baseline Model

First machine learning based on KNN algorithm, by choosing right features and value of K, this task can be considered as a binary classification problem. For each data in test set, we need to calculate the distance of most K nearest neighbors and decide whether this passenger can survive in the disaster. Since some features are string format, we may turn them into dummy variables. Such as sex and embarked. Right now, name and tickets seem do not have much relationship with survive, and for cabin, most data are missing, so at the baseline model, name ticket and cabin are discard. Represent sex as 0 and 1, embarked as 0, 1, 2, and use Euclidean distance to calculate the distance. Choosing k by cross validation. Try k from 5 to 20, choosing k = 9. For baseline model. Using sequential feature selection, which means adding one feature each time, keep other factors stable, observing the result of adding features. Sometimes adding new feature may increase the performance, other times adding more feature have negative influence on the performance. After experimenting, using the following six feature, which are age, pcalss, sibsp, parch, fare and sex, for the baseline KNN model. For each feature, since now they are all numerical, calculate the max and min value and normalize them, so that the range of feature won't influence the result.

Feature selected

Pclass, Sex, Age, SibSp, Parch, Fare

Result

The result of baseline model is as follows. About 73.674% accuracy.

sortedRes.csv
2 minutes ago by Yijia
add submission details
0.73684

Confusion matrix:

(predict)	['0', '1'] (truth)
0	[202, 58]
1	[58, 100]

Precision, recall and F1 score:
confusion matrix0 (Not Survived)
[202, 58]
[58, 100]

Recall	0.776923076923
Precision	0.776923076923
F1	0.776923076923

confusion matrix1 (Survived)
[100, 58]
[58, 202]

Recall	0.632911392405
Precision	0.632911392405
F1	0.632911392405

We can see that for not survival, our model has better performance.

Model with Feature Engineering

As discussed in part of Data Engineering, adding new features to baseline models and discard some old features. Here adding family size, is along, fare per person, ticket length, title and drop sibsp, parch. And for age, fare, sex, embarked, using dummy value. Using sequential feature selection, and cross validation, finally choosing the following features: age, pclass, isalone, fare, sex, embarked, title. For each feature, normalize it to 0-1 bond. Keep k = 9 stable, we can see there is an increase in accuracy, proving that feature engineering is successful.

Feature selected

Pclass, Sex, Age, Fare, isAlone, Embarked, Title

Result

Increase accuracy from about 73% to 75%.

4809 new Yijia 0.75598 1 1m

Confusion matrix:

(predict)	['0', '1'] (truth)
0	[203, 57]
1	[53, 105]

Precision, recall and F1 score:
confusion matrix0 (Not Survived)
[203, 57]
[53, 105]

recall	0.780769230769
precision	0.79296875
F1	0.786821705426

confusion matrix1 (Survived)
[105, 53]
[57, 203]

recall	0.664556962025
precision	0.648148148148
F1	0.65625

Both performance improved after feature engineering.

RANDOM FOREST MODELS

Baseline model

Random forest constructs a number of decision trees and get the result from voting the results of the decision trees. By choosing data randomly for each tree and choosing features randomly for each node, the random forest trees can prevent overfitting in some degree. For the baseline random forest model, from the experiments, choosing 2/3 from training set for each decision tree and k –

2 features for each node, assuming we have k features in total. Choosing N = 10 trees in the forest. Similarly, using sequential feature selection and cross validation, choose several features for baseline model. Here we do not need normalization, and each time we calculate entropy of features to choose the one with large information entropy.

Feature selected

Pclass, Sex, Age, SibSp, Parch, Fare, Embarked

Result

About 74% accuracy.

RF_Titanic.csv
just now by Yijia
add submission details
0.74163

Confusion matrix:

(predict)	['0', '1'] (truth)
0	[207, 53]
1	[60, 98]

Precision, recall and F1 score:

confusion matrix0 (Not Survived)

[197, 63]

[55, 103]

recall	0.757692307692
precision	0.781746031746
F1	0.76953125

confusion matrix1 (Survived)

[103, 55]

[63, 197]

recall	0.651898734177
precision	0.620481927711
F1	0.635802469136

The performance of survived is worse than not survived.

Model with Feature Engineering

By using feature engineering's result described as above, adding new feature like title, ticket length, is alone, family size and fare per person. Discard parch and sibsp. After choosing features by experiments, we can find the prediction result has a higher accuracy than baseline model. For random forest we don't need normalization.

Feature selected

Pclass, Sex, Age, Fare, isAlone, Title, ticket length,

Result

Improve accuracy by about 2%

4541 new Yijia 0.76077 4 ~10s

Confusion matrix:

(predict)	['0', '1'] (truth)
0	[210, 50]
1	[53, 105]

Precision, recall and F1 score:

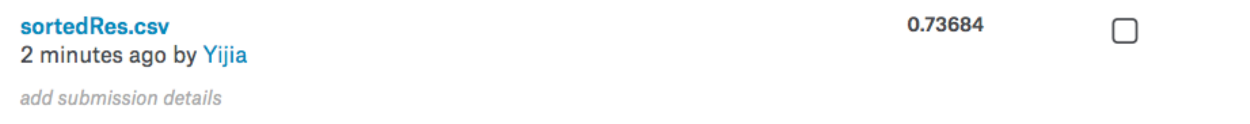
```
confusion matrix0 (Not Survived)
[210, 50]
[53, 105]
recall          0.807692307692
precision       0.798479087452
F1              0.803059273423
confusion matrix1 (Survived)
[105, 53]
[50, 210]
recall          0.664556962025
precision       0.677419354839
F1              0.670926517572
```

Both performance increased.

3 Result and Conclusion

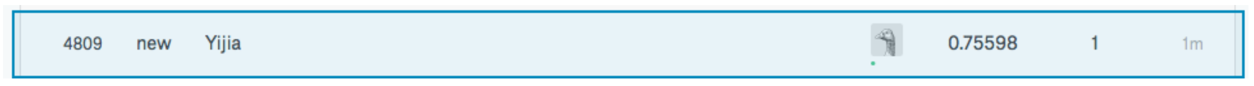
Screenshot of Rank

KNN baseline & with feature engineering



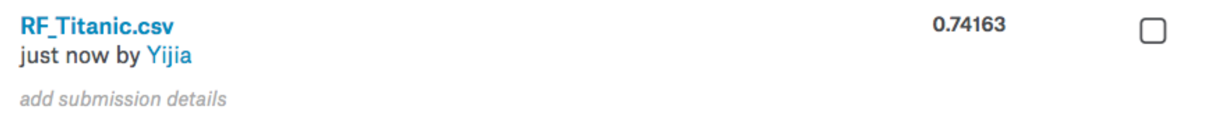
sortedRes.csv
2 minutes ago by Yijia
add submission details

0.73684



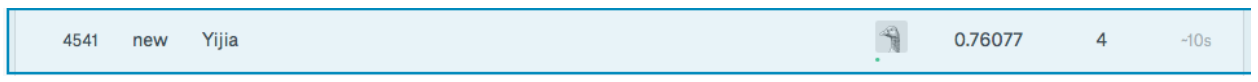
4809 new Yijia 0.75598 1 1m

Random Forest baseline & with feature engineering



RF_Titanic.csv
just now by Yijia
add submission details

0.74163



4541 new Yijia 0.76077 4 ~10s

Conclusion

From the experiments, we can have conclusion as follows:

- Some groups of people had more possibility of surviving than the others, and the survival can be predicted.
- Both KNN and Random Forest model can predict survive with almost similar accuracy.
- With feature engineering, the performance of prediction increases.

4 How to Run on Aspen

KNN – using java, command line as follows:

```
javac KNN_copy.java -cp $(hadoop classpath)

jar cf KNN_Titanic.jar *.class

jar tf KNN_Titanic.jar

hdfs dfs -rm /user/yijiaj/hadoop/input/*

hdfs dfs -mkdir /user/yijiaj/hadoop/input

scp /Users/yijia/IdeaProjects/KNNr/src/main/java/train.csv yijiaj@aspen.lti.cs.cmu.edu:/home/yijiaj

hdfs dfs -put /home/yijiaj/train.csv /user/yijiaj/hadoop/input/train.csv

hdfs dfs -rm -r /user/yijiaj/hadoop/output

hadoop jar KNN_Titanic.jar KNN_copy /user/yijiaj/hadoop/input /user/yijiaj/hadoop/output
```

Random forest – using python, command line as follows:

```
scp /Users/yijia/PycharmProjects/DecisionTree/train.csv yijiaj@aspen.lti.cs.cmu.edu:/home/yijiaj

scp /Users/yijia/PycharmProjects/DecisionTree/test_new.csv yijiaj@aspen.lti.cs.cmu.edu:/home/yijiaj

scp /Users/yijia/Desktop/RFMapReduce_copy.py yijiaj@aspen.lti.cs.cmu.edu:/home/yijiaj

[yijiaj@aspen ~]$ mkdir input

scp /Users/yijia/Desktop/input_copy/* yijiaj@aspen.lti.cs.cmu.edu:/home/yijiaj/input

[yijiaj@aspen ~]$ python RFMapReduce_copy.py /home/yijiaj/input
```