

Assignment 1-Yinan Yan

by Yinan Yan

FILE	24276_YINAN_YAN_ASSIGNMENT_1-YINAN_YAN_309231_2124548707.DOCX (637.47K)		
TIME SUBMITTED	30-NOV-2020 04:46AM (UTC+0000)	WORD COUNT	2686
SUBMISSION ID	137786953	CHARACTER COUNT	15333

Titanic Shipwreck Survivors Prediction Based on Machine Learning Algorithms

Yinan Yan 2006532

Abstract: The sinking of the Titanic is one of the most serious shipwreck in the world. This report is about using machine learning algorithms to predict the survivors of the Titanic. In this report, Decision Tree Algorithm and Random Forest Algorithm are used to build different models to train and test the Titanic data set. After modeling and evaluation, it is concluded that the Random Forest Algorithm has better performance in such data sets than the Decision Tree Algorithm. Compared with the decision tree algorithm, the Random Forest Algorithm makes up for the Decision Tree Algorithm' problem of possible overfitting.

Keywords: Machine learning, Decision Tree Algorithm, Random Forest Algorithm, Logic Regression Algorithm

1. Introduction

1.1 Machine Learning

According to [1], Machine Learning is the science (and art) of programming computers so they can learn from data.

Here is a slightly more general definition:

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.

—Arthur Samuel, 1959

And a more engineering-oriented one:

A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E .

—Tom Mitchell, 1997

As shown by [2], machine learning mainly uses experience to learn, by analyzing and learning a large amount of existing data, to obtain the internal correlation between data. In recent years. With the improvement of data foundation, algorithm performance and hardware computing power, machine learning has made rapid development, especially deep learning in image, voice and natural language processing.

As demonstrated in [3], machine learning provides a method of automatically summarizing logic or rules from data by selecting appropriate algorithms. These logic or rules can be used as models to classify and predict new data.

1.2 Dataset information

The Titanic data set is a classic data set in machine learning。 The

Titanic data set include 12 columns and 892 rows. Except the titles of the data set, it has 891 passengers' information, and except the PassengerID, the data set has 1 label (Survived) and 10 attributes (Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin and Embarked).

Obviously, we should analyze the correlation between the attributes and label, and then build a suitable model. This model will have good performance to predict the dataset, which are not only training dataset, but also testing dataset.

2. The process of data analysis

We use machine learning algorithms to analyze the passengers' data of the Titanic and build the models, in order to predict who may become survivors through the model. In the process of building the entire model, we need to perform roughly the following four steps:

- 1.read the data
- 2.explore the data
- 3.data cleaning and pre-processing
- 4.build the models

3. Data processing

3.1 Attributes information

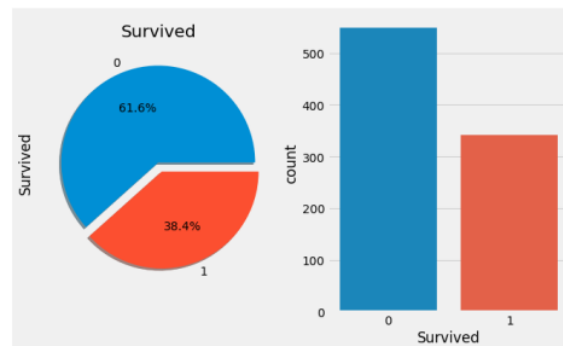
This dataset includes 10 attributes, which are Pclass, Name, Sex, Age, SibSp, Parch, Ticket, Fare, Cabin and Embarked. In these attributes, Sex, Embarked, Pclass, SibSp and Parch are discrete variables. Age and Fare are continuous variables. Mixed variables include Ticket and Cabin. As for Name, these are the passengers' name. Pclass means the class of the passenger cabin, included class 1, class 2 and class 3. Sex includes male and female. SibSp has two value (0 and 1). 0 means that the passengers do not have sisters or brothers on the ship. 1 means that the passengers have sisters or brothers. Parch has two value (0 and 1). 0 means that the passengers do not have parents on the ship. 1 means that the passengers have parents on the ship. Ticket means the ticket' number. Fare means the ticket' price. Cabin means the specific cabin. Embarked means where the passengers embarked, and its value include S, C and D.

3.2 Label information

This dataset includes one label, which is Survived. Survived is discrete variable. Survived has two value, which are 0 and 1. The meaning of 0 is that the passenger survived, and the meaning of 1 is that the passenger did not survive.

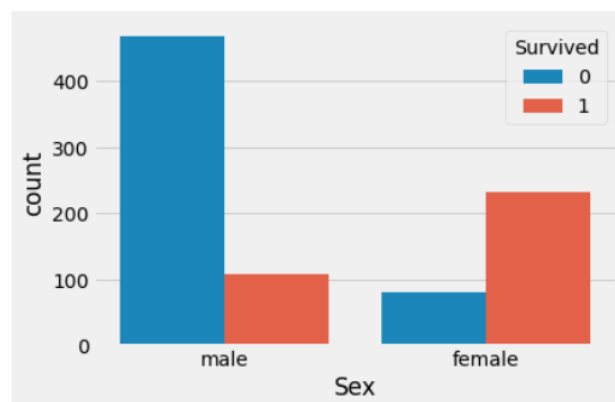
3.3 Attributes analysis

3.3.1 Label proportion



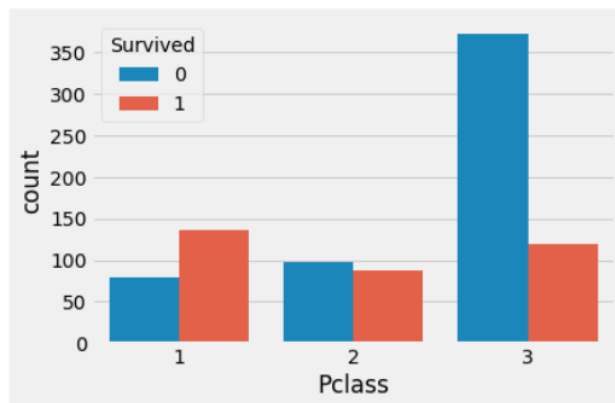
The chart shows that 61.6% passengers have survived.

3.3.2 The correlation between Sex and Survived



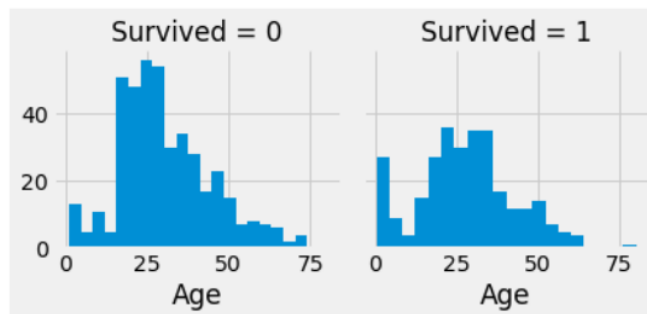
Even though there are more male than the female, the female has higher survival rate.

3.3.3 The correlation between Pclass and Survived



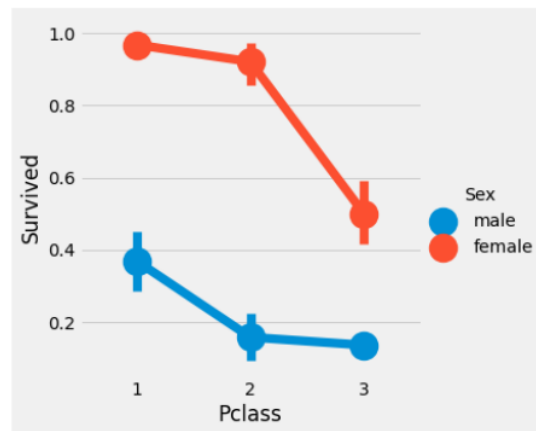
The passengers who are in passenger cabin 1 have the highest survival rate.

3.3.4 the correlation between Age and Survived



Baby and older people have higher survival rate than young adults who is about 30 years old.

3.3.5 the correlation between Age Pclass and Survived



Female passengers have higher survival rate, and passengers who are in passenger cabin 1 have higher survival rate.

3.4 Missing value filling and data pre-processing

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 891 entries, 1 to 891
Data columns (total 11 columns):
#   Column      Non-Null Count  Dtype
---  -
0   Survived    891 non-null    int64
1   Pclass      891 non-null    int64
2   Name        891 non-null    object
3   Sex         891 non-null    object
4   Age         714 non-null    float64
5   SibSp       891 non-null    int64
6   Parch       891 non-null    int64
7   Ticket      891 non-null    object
8   Fare        891 non-null    float64
9   Cabin       204 non-null    object
10  Embarked    889 non-null    object
dtypes: float64(2), int64(4), object(5)
```

We can find that there are three attributes which are Cabin, Age and Embarked. Cabin missed about 700 values, so we decide to delete this attribute. As for Age, it missed about 200, we can use the average value to

update the null. About the Embarked, it just missed 2 values, so we can delete the two rows. The operation influenced the dataset a little.

After filling the missing value, we can find that there are some attributes is not meaningful for the dataset. we can delete the attribute to simplified the dataset, because it can make the work easily. Deleting the columns of Name and Ticket is a suitable way for the dataset.

4. correlation heat map

After these operations, we draw the heat map to see the correlation between label and attributes intuitively.



5. building and testing the model

5.1 choosing the algorithms

According to the dataset, we should choose the supervised learning. Supervised learning includes Logic Regression, KNN, SVM, Decision Tree and Random Forest. In this case, I decide to choose Logic Regression, Decision Tree and Random Forest Algorithm.

5.2 Introduction of choosing algorithms

5.2.1 Logic regression algorithm

As shown by Hailong Duan[5], logistic regression algorithm is a widely used classification algorithm. The logistic regression algorithm trains the positive and negative samples in the data set to obtain the hypothesis function between the sample label and the sample feature. Due to the low complexity and easy implementation of the logistic regression algorithm, it is widely used in the industry.

5.2.2 Decision Tree and Random Forests Algorithm

According to [6], a decision tree is a tree structure that includes three parts: a root node, a branch, and a leaf. The root node represents an attribute of the tree, the leaf represents the classification mark, and the branch represents the output result. The method starts from the root node and

traverses repeatedly. According to the results of the test, the instance is assigned to its child nodes. Each child node will have a value corresponding to the feature. Through the recursive method, continue to test and assign the instance. Until the leaf node is reached, the instance is finally divided into the class of the leaf node.

As demonstrated in [4]. The Random Forests Algorithm is an integrated learning method. Through the self-service resampling technique, n samples are randomly selected from the original training sample set N with replacement to generate a new training sample set to train a decision tree, and then m trees are generated according to the above steps. The decision tree constitutes a random forest, and the classification result of the new data is determined by the score formed by the voting of the classification tree.

5.3 Logic Regression Algorithm

I use the Logic Regression Algorithm in this article to discuss the survived situation, code is shown below.

```
#Logistic regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(Xtrain, Ytrain)
Y_pred = logreg.predict(Xtest)
acc_log = round(logreg.score(Xtrain, Ytrain), 2)
acc_log
```

0.79

Run the code, and then we get the value of `acc_log`, which is 0.79. The result of those code is that the Logic Regression model' accuracy is 0.79.

5.4 Decision Tree and Random Forest Algorithms

5.4.1 Decision Tree Algorithm

I use the Decision Tree Algorithm to discuss the survived situation, code is shown below.

```
#cross value
from sklearn.model_selection import cross_val_score
score = cross_val_score(clf,X,y,cv=10).mean()

score
```

```
0.7739274770173645
```

Run the code, and then we get the value of `score_`, which is 0.7715. The result of those code is that the Logic Regression model' accuracy is 0.7715. Because the system divides the training and testing data sets randomly, the division of the data set may influence the model' accuracy. Considering above condition, we use the way of cross validation to check the model. The code of cross validation is shown below.

```
#cross value
from sklearn.model_selection import cross_val_score
score = cross_val_score(clf,X,y,cv=10).mean()

score
```

```
0.7739274770173645
```

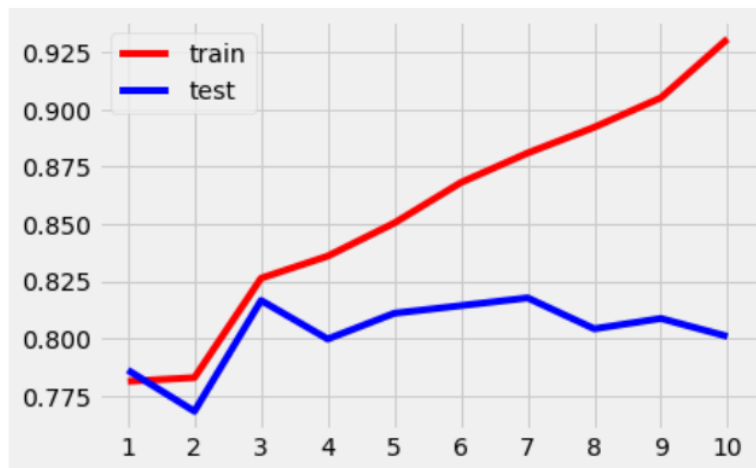
We get the score' value, which is 0.7739. The accuracy of this model

is not enough for the requirement, so we choose to change the max depth of the decision tree. We can get a most suitable model for the data set. The code of the method is shown below.

```
#check the model' fitting condition at different max_depth
from sklearn.tree import DecisionTreeClassifier
tr = []
te = []
for i in range(10):
    clf = DecisionTreeClassifier(random_state=25
                                ,max_depth=i+1
                                )
    clf = clf.fit(Xtrain, Ytrain)
    score_tr = clf.score(Xtrain,Ytrain)
    score_te = cross_val_score(clf,X,y,cv=10).mean()
    tr.append(score_tr)
    te.append(score_te)
print(max(te))
plt.plot(range(1,11),tr,color="red",label="train")
plt.plot(range(1,11),te,color="blue",label="test")
plt.xticks(range(1,11))
plt.legend()
plt.show()
```

0.8177860061287026

We get the better model' accuracy is 0.8178. We can find that the better suitable max_depth is 3.



Even though, the accuracy of this model is good now, we also have a

method to improve the model's accuracy. We can use the method of grid searching to get a better model. Through the method, we can get the parameter in the better situation. The code of this method is shown below.

```
#grid search
from sklearn.model_selection import GridSearchCV
gini_thresholds = np.linspace(0,0.5,20)

parameters = {'splitter':('best','random')
              , 'criterion':('gini',"entropy")
              , "max_depth":[*range(1,10)]
              , 'min_samples_leaf':[*range(1,50,5)]
              , 'min_impurity_decrease':[*np.linspace(0,0.5,20)]
              }

clf = DecisionTreeClassifier(random_state=25)
GS = GridSearchCV(clf, parameters, cv=10)
GS.fit(Xtrain,Ytrain)

GS.best_params_

{'criterion': 'entropy',
 'max_depth': 6,
 'min_impurity_decrease': 0.0,
 'min_samples_leaf': 11,
 'splitter': 'best'}

GS.best_score_

0.821479774705581
```

The best parameters of the model is shown below.

Criterion = entropy,

Information entropy is the criterion for evaluating impurity.

Max_depth = 6,

Max_depth is the maximum depth of Decision Tree.

Min_impurity_decrease = 0.0,

After a node is branched, the decrease in impurity is greater than or equal to the value of min_impurity_decrease, then the branch will not continue.

min_samples_leaf = 11,

Min_samples_leaf is the minimum sample size required for an intermediate node to continue branching.

Splitter = best

Decision Tree have the best branches.

And in this situation, the better accuracy is 0.8215.

5.4.2 Random Forest

In this part, I want to compare the performance of Decision Tree and RF, So I run the two together. The code of RF algorithm is shown below.

```
#check the accuracy of decision tree and random forest in the same condition
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3)

clf = DecisionTreeClassifier(random_state=0)
rfc = RandomForestClassifier(random_state=0)
clf = clf.fit(Xtrain,Ytrain)
rfc = rfc.fit(Xtrain,Ytrain)
score_c = clf.score(Xtest,Ytest)
score_r = rfc.score(Xtest,Ytest)

print("Single Tree:{}".format(score_c)
      , "Random Forest:{}".format(score_r)
      )
```

Single Tree:0.7528089887640449 Random Forest:0.7902621722846442

We get the result. the result is that decision tree is 0.7528 and the random forest is 0.7902. in the same situation, random forest performs better than decision tree.

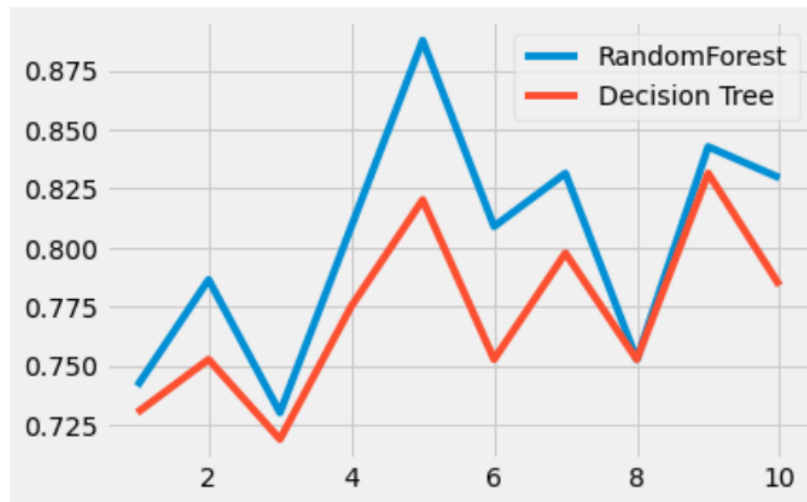
To check the performance of Decision Tree and Random Forest, we use cross validation and draw a figure to observe it intuitively. the code is shown below.

```
#cross value about decision tree and random forest
rfc = RandomForestClassifier(n_estimators=25)
rfc_s = cross_val_score(rfc,X,y,cv=10)

clf = DecisionTreeClassifier()
clf_s = cross_val_score(clf,X,y,cv=10)

plt.plot(range(1,11),rfc_s,label = "RandomForest")
plt.plot(range(1,11),clf_s,label = "Decision Tree")
plt.legend()
plt.show()
```

we get the figure that is shown below. we can get a conclusion from the figure, which is that, decision tree and random forest have same tendency and random forest perform better than decision tree basically.



6. Conclusions and Perspectives

6.1 Conclusion

This essay uses three algorithms to build the model, and estimate the model' accuracy respectively. Through the comparison, we find that decision tree shows the best performance, which is 0.8215. Because we use

the grid searching method to find the best parameters of decision tree in this condition. After this, we compare the decision tree and random forest, we find that random forest always performs better than decision tree. In summary, random forest have the best performance in these algorithms in the same condition.

6.2 Perspective

Considering the practical factors of this coursework, this article is just simple. If need to improve the model, we can improve the model through several aspect below.

1. When we process the data, we can process the data more careful.
2. We can use more algorithms, and we can find the best in those algorithms.
3. We can use grid searching for every algorithm, and find the best parameters for every algorithm. The result from this method will be more convincing.

Reference

- [1] Géron. Aurélien, "The Machine Learning Landscape" in Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems, 2nd ed, USA: O'Reilly Media, Inc, 2019, pp, 4-4
- [2] T. Li, J. J. Zhang, W. Zhao, Z. Z. Hou, "Power Consumption Data Analysis Based on Machine Learning," (基于机器学习的用电数据分析), Hebei Electric Power, vol. 39, no. 5, p.17-21, 2020
- [3] F. Wang, H. Fan, "Teaching Research on Machine Learning Courses for Data Science Graduates," (面向数据科学研究生的机器学习课程教学研究), Computer Education, vol. 11, p.135, 2020
- [4] Q. Li, Y. Yi, Z. D. Wu, T. Ding, "Review of gut microbiome analysis prediction models and algorithms," (基于机器学习的肠道菌群数据建模与分析研究综述), Microbiology Bulletin, vol. 1, p.17, Nov, 2020.
- [Online]. Available: <https://doi.org/10.13344/j.microbiol.china.200346>
- [5] H. L. Duan, "Research on Prediction of User Buying Behavior Balance and Model Fusion," M.S. thesis, CIE, NCU, NC, JX, 2020
- [6] H. Q. Zhang, "Design and implementation of R&D project management system based on decision tree algorithm," M.S. thesis, B UCEA, BJ, BJ, 2020

Appendix

```
1. import numpy as np
2. import pandas as pd
3. import matplotlib.pyplot as plt
4. import seaborn as sns
5. plt.style.use('fivethirtyeight')
6. import warnings
7. warnings.filterwarnings('ignore')
8. data = pd.read_csv(r"Taitanic data.csv",index_col = 0)
9. data.head(10)
10. data.info()
11. # the proportion of the label-Survival
12. f,ax = plt.subplots(1,2,figsize=(10,6))
13. data['Survived'].value_counts().plot.pie(explode=[0,0.1],
14.                                           autopct='%1.1f%%',
15.                                           ax=ax[0],shadow=True)
16. ax[0].set_title('Survived')
17. sns.countplot('Survived',data=data,ax=ax[1])
18. # the relation between different genders and Survival
19. data.groupby(['Sex','Survived'])['Survived'].count()
20. sns.countplot('Sex',hue='Survived',data=data)
21. # the relation between the Pclass and Survival
22. pd.crosstab(data.Pclass,data.Survived,margins=True).style.background_gradient(cmap='GnBu_r')
23. sns.countplot('Pclass',hue='Survived',data=data)
24. #the relation between Age and Survival
25. g = sns.FacetGrid(data, col='Survived')
26. g.map(plt.hist, 'Age', bins=20)
27. plt.show()
28. #delete the columns which have too many missing values and are irrelevant with the labels from observation
29. data.drop(["Cabin","Name","Ticket"],inplace=True,axis=1)
30. #process the missing values
31. data["Age"] = data["Age"].fillna(data["Age"].mean())
32. data = data.dropna()
33. #transfer binary viaribles(male,female) into numerical viariables(0,1)
34. data["Sex"] = (data["Sex"]=="male").astype("int")
35. #transfer tricategorical variable(S,C,D) into numerical variable(0,1,2)
36. labels = data["Embarked"].unique().tolist()
37. data["Embarked"] = data["Embarked"].apply(lambda x: labels.index(x))
38. data.head(10)
39. # correlation heat map
```

```

40. sns.heatmap(data.corr(),annot=True,linewidths=0.2,cmap='summer_r') #data.co
    rr()return the value of correlation
41. fig = plt.gcf() #establish canvas
42. fig.set_size_inches(8,6) #canvas size
43. #separate the label and attributes
44. X = data.iloc[:,data.columns != "Survived"]
45. y = data.iloc[:,data.columns == "Survived"]
46. #separate the training dataset and testing dataset
47. from sklearn.model_selection import train_test_split
48. Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3)
49.
50. #correct the index of training dataset and testing data set
51. for i in [Xtrain, Xtest, Ytrain, Ytest]:
52.     i.index = range(i.shape[0])
53.
54. #check the training data set' information
55. Xtrain.head()
56. #check the testing dataset' information
57. Xtest.head()
58. #check the training data set' size
59. Xtrain.shape
60. #logistic regression
61. from sklearn.linear_model import LogisticRegression
62. logreg = LogisticRegression()
63. logreg.fit(Xtrain, Ytrain)
64. Y_pred = logreg.predict(Xtest)
65. acc_log = round(logreg.score(Xtrain, Ytrain), 2)
66. acc_log
67. #decision tree
68. from sklearn.tree import DecisionTreeClassifier
69. clf = DecisionTreeClassifier(random_state=25)
70. clf = clf.fit(Xtrain, Ytrain)
71. score_ = clf.score(Xtest, Ytest)
72. score_
73. #cross value
74. from sklearn.model_selection import cross_val_score
75. score = cross_val_score(clf,X,y,cv=10).mean()
76.
77. score
78. #check the model' fitting condition at different max_depth
79. from sklearn.tree import DecisionTreeClassifier
80. tr = []
81. te = []
82. for i in range(10):

```

```

83.     clf = DecisionTreeClassifier(random_state=25
84.                                   ,max_depth=i+1
85.                                   )
86.     clf = clf.fit(Xtrain, Ytrain)
87.     score_tr = clf.score(Xtrain,Ytrain)
88.     score_te = cross_val_score(clf,X,y,cv=10).mean()
89.     tr.append(score_tr)
90.     te.append(score_te)
91.     print(max(te))
92.     plt.plot(range(1,11),tr,color="red",label="train")
93.     plt.plot(range(1,11),te,color="blue",label="test")
94.     plt.xticks(range(1,11))
95.     plt.legend()
96.     plt.show()
97. #grid search
98. from sklearn.model_selection import GridSearchCV
99. gini_thresholds = np.linspace(0,0.5,20)
100.
101. parameters = {'splitter':('best', 'random')
102.               , 'criterion':("gini", "entropy")
103.               , "max_depth":[*range(1,10)]
104.               , 'min_samples_leaf':[*range(1,50,5)]
105.               , 'min_impurity_decrease':[*np.linspace(0,0.5,20)]
106.               }
107.
108. clf = DecisionTreeClassifier(random_state=25)
109. GS = GridSearchCV(clf, parameters, cv=10)
110. GS.fit(Xtrain,Ytrain)
111.
112. GS.best_params_
113. GS.best_score_
114. #check the accuracy of decision tree and random forest in the same condition
115. from sklearn.tree import DecisionTreeClassifier
116. from sklearn.ensemble import RandomForestClassifier
117. Xtrain, Xtest, Ytrain, Ytest = train_test_split(X,y,test_size=0.3)
118.
119. clf = DecisionTreeClassifier(random_state=0)
120. rfc = RandomForestClassifier(random_state=0)
121. clf = clf.fit(Xtrain,Ytrain)
122. rfc = rfc.fit(Xtrain,Ytrain)
123. score_c = clf.score(Xtest,Ytest)
124. score_r = rfc.score(Xtest,Ytest)
125.

```

```
126. print("Single Tree:{}".format(score_c)
127.       , "Random Forest:{}".format(score_r)
128.       )
129. #cross value about decision tree and random forest
130. rfc = RandomForestClassifier(n_estimators=25)
131. rfc_s = cross_val_score(rfc,X,y,cv=10)
132.
133. clf = DecisionTreeClassifier()
134. clf_s = cross_val_score(clf,X,y,cv=10)
135.
136. plt.plot(range(1,11),rfc_s,label = "RandomForest")
137. plt.plot(range(1,11),clf_s,label = "Decision Tree")
138. plt.legend()
139. plt.show()
```

Assignment 1-Yinan Yan

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

83/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

GRADING FORM: SMU GENERIC_GJ

YINAN YAN

82.50

RESEARCH-INFORMED LITERATURE

Some very good work here indeed. References ok but it would be nice to see some more papers related to your dataset also, if only for comparison.

6

KNOWLEDGE AND UNDERSTANDING

Some excellent work here. Process clear. Data pre-processing good. Algorithms - you could have included a little more detail on these and an explicit discussion of parameters for each, though I accept the word count may limit this.

16.5

ANALYSIS

Some very good work here. Choices firmly informed by data and its limitations. Application of cross-validation and hyper-parameter tuning well considered. Results discussed clearly.

30.5

PRACTICAL APPLICATION AND DEP.



Appropriate libraries selected and used to good effect. Well done.

26.5

SKILLS FOR PROFESSIONAL PRAC.



Refs appropriate - please use references to back up assertions, definitions, etc. Well structured and graphs, in general helpful.

3