Bonus Project Report

CSCC11 Bonus

Zhiqian Chen 1001156900
Lixiang Wei 1001117374
2016/12/17

Titanic: Machine Learning from Disaster

Problem description

Viasulize Data
PassengerId Survived Pclass Age SibSp Parch Fare
count 891.000000 891.000000 891.000000 714.000000
891.000000
mean 446.000000 0.383838 2.308642 29.699118
0.523008
std 257.353842 0.486592 0.836071 14.526497
1.102743
min 1.000000 0.000000 1.000000 0.420000
0.000000
25% 223.500000 0.000000 2.000000 NaN
0.000000
50% 446.000000 0.000000 3.000000 NaN
0.000000
75% 668.500000 1.000000 3.000000 NaN 1.000000
max 891.000000 1.000000 3.000000 80.000000 8.000000

	Parch	Fare
count	891.000000	891.000000
mean	0.381594	32.204208
std	0.806057	49.693429
min	0.000000	0.000000
25%	0.000000	7.910400
50%	0.000000	14.454200
75%	0.000000	31.000000
max	6.000000	512.329200

Class abd Survived rate

Pclass	Survived
1	0.629630
2	0.472826
3	0.242363

Sex and Survived rate
Sex Survived

0	female	0.742038
1	male	0.188908

Family Size and Survived Rate

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

Aloneness and Survival Rate

IsAlone	Survived
0	0.505650
1	0.303538

Fare and Survived rate

CategoricalFare 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

Age and Survived Rate

CategoricalAge Survived
0 (-0.08, 16] 0.527273
1 (16, 32] 0.351648
2 (32, 48] 0.378049
3 (48, 64] 0.434783
4 (64, 80] 0.090909

From different relationships, we can conclude the Class can affect survival rate, and female has higher survived rate than male. Here we don't consider title and first name, since if there are more female than male survived, the majority of name will be female's, then looking for the relationship between name and survived rate doesn't make sense. Also, the distribution of fare resemble to class, since better classes will match higher fare.

Baseline Method

Since there are many data with null values, first we have to eliminate those data. Then we apply KNN on training set. Use false positive and true positive rate plot to to determine if ratio > 0.5. Lastly, plot error points and correct points corresponding to Class and Fare.

Improvement:

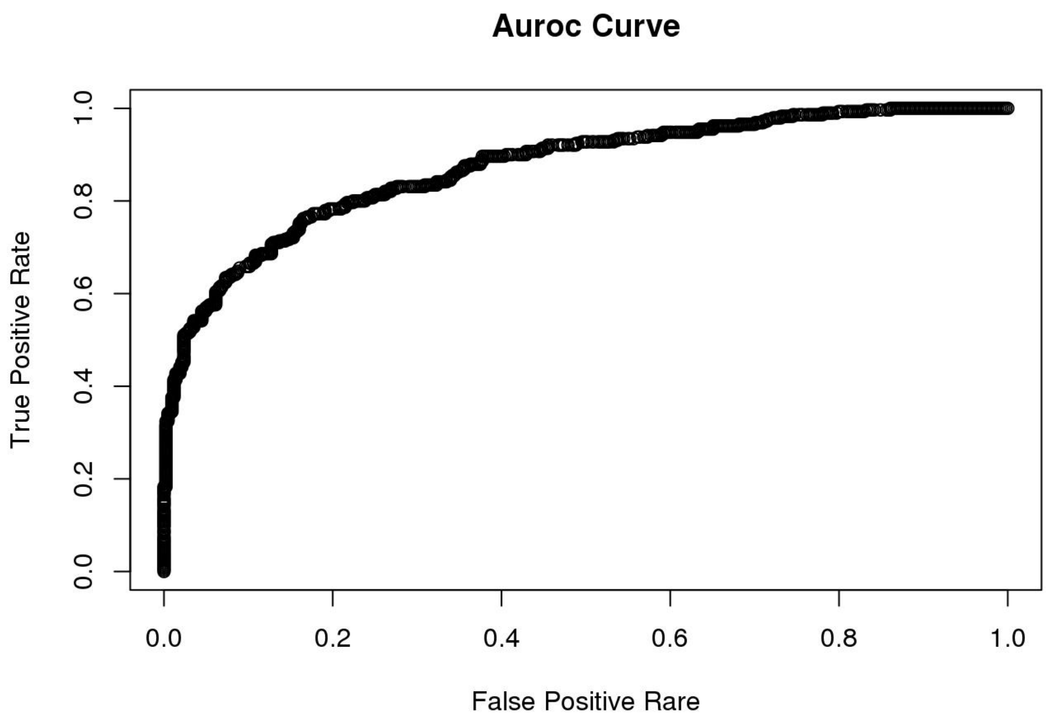
Instead of deleting data with null value, we can generate missing value randomly. For example, if data point j is missing fare, we can generate fare randomly, but within interval (mean-std,mean_std). Similarly, we can generate all missing values by this approach.

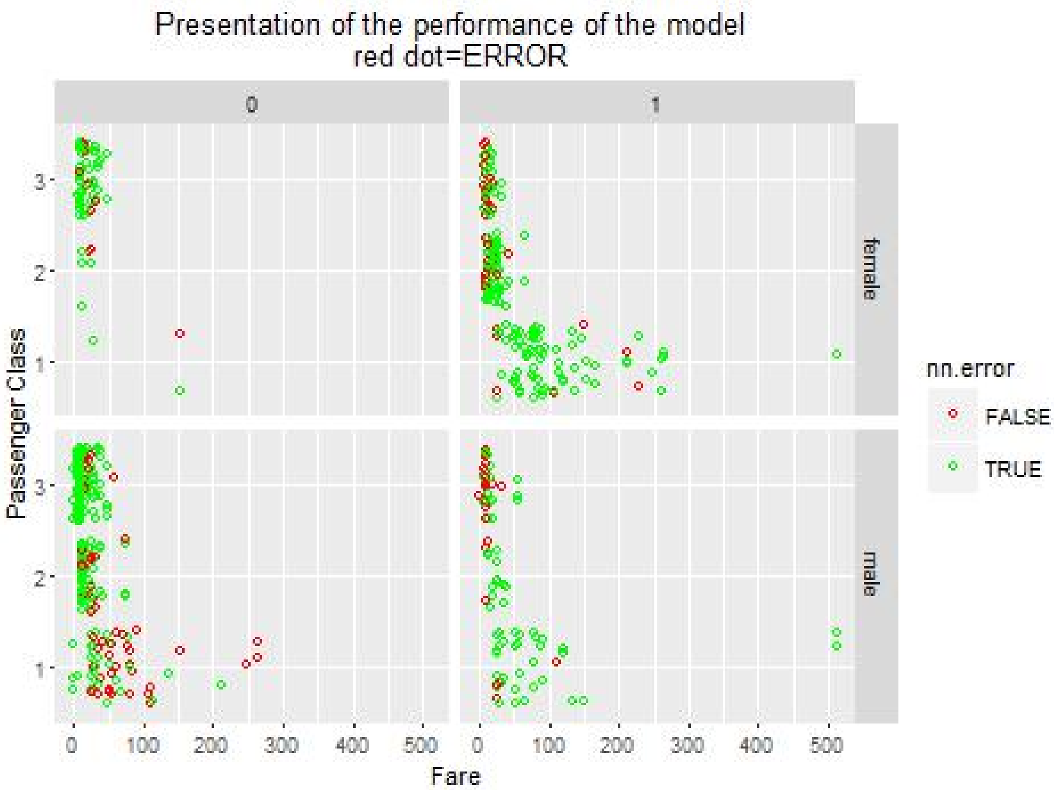
R Code:

```
options(warn=-1)
library(ggplot2)
library(nnet)
library(ROCR)
train <- read.csv("../input/train.csv", header=TRUE, stringsAsFactors = FALSE)
# Set Survived, Pclass, Sex, Embarked as factors
for(i in c(2,3,5, 12)){
  train[,i] <- as.factor(train[,i])
}
# NullValue   handling... Deletion of all rows with a NA
#It mainly concerns Age variable
for (i in 1:dim(train)[2])
{
  train <- train[!is.na(train[,i]),]
}
nn <- nnet(Survived ~ Age + Fare + Embarked+ SibSp   + Parch, train, size =
100)
nn.prediction <- predict(nn, newdata = train)
pred <- prediction(nn.prediction, train$Survived)
tpr <- unlist(performance(pred, "tpr")@y.values)
fpr <- unlist(performance(pred, "fpr")@y.values)

plot(fpr, tpr, main="Auroc Curve", xlab="False Positive Rate", ylab="True
Positive Rate", lt=1)
train$nn.error = (train$Survived != ifelse(nn.prediction <.4,1,0))
evels(train$Survived) <- c("Did not Survive", "Survived")
ggplot(train, aes(x=Fare, y=Pclass, colour=nn.error)) +
```

```
geom_point(shape=1, position = "jitter") + facet_grid(Sex~Survived) +
scale_colour_manual(values=c("red", "green")) +
ggtitle("Presentation of the performance of the model \n red dot=ERROR") +
ylab("Passenger Class") +
xlab("Fare")
```





Advanced baseline from the Kaggle community and Improvement

Python code:

```
# Imports

# pandas
import pandas as pd
from pandas import Series,DataFrame

# numpy, matplotlib, seaborn
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#Print you can execute arbitrary python code
train = pd.read_csv("../input/train.csv", dtype={"Age": np.float64}, )
test = pd.read_csv("../input/test.csv", dtype={"Age": np.float64}, )

#Print to standard output, and see the results in the "log" section below after
running your script
print("\n\nTop of the training data:")
print(train.head())

print("\n\nSummary statistics of training data")
print(train.describe())
```

```
train.to_csv('copy_of_the_training_data.csv', index=False)

# Check the relation between survived and CLass
print (train[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean())
# Check the relation between survived and Sex
print(train[['Sex', 'Survived']].groupby(['Sex'], as_index=False).mean())

full_data = [train, test]
for dataset in full_data:
        dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1
print (train[['FamilySize', 'Survived']].groupby(['FamilySize'],
as_index=False).mean())


for dataset in full_data:
        dataset['IsAlone'] = 0
        dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1
print (train[['IsAlone', 'Survived']].groupby(['IsAlone'], as_index=False).mean())


#Fare also has some missing value and we will replace it with the median. then
we categorize it into 4 ranges
# relation between fare and survived rate
for dataset in full_data:
      dataset['Fare'] = dataset['Fare'].fillna(train['Fare'].median())
      train['CategoricalFare'] = pd.qcut(train['Fare'], 4)
print (train[['CategoricalFare', 'Survived']].groupby(['CategoricalFare'],
as_index=False).mean())

# relation between age and survived
for dataset in full_data:
    age_avg    = dataset['Age'].mean()
    age_std    = dataset['Age'].std()
    age_null_count = dataset['Age'].isnull().sum()
    # Fill out null value with randon generated age
    age_null_random_list = np.random.randint(age_avg - age_std, age_avg +
age_std, size=age_null_count)
    dataset['Age'][np.isnan(dataset['Age'])] = age_null_random_list
    dataset['Age'] = dataset['Age'].astype(int)
train['CategoricalAge'] = pd.cut(train['Age'], 5)
print (train[['CategoricalAge', 'Survived']].groupby(['CategoricalAge'],
as_index=False).mean())
```

```
# Data Cleaning


for dataset in full_data:
    # Mapping Sex
    dataset['Sex'] = dataset['Sex'].map( {'female': 0, 'male': 1} ).astype(int)

    # Mapping titles
    title_mapping = {"Mr": 1, "Miss": 2, "Mrs": 3, "Master": 4, "Rare": 5}
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

    # Mapping Embarked
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q':
2} ).astype(int)

    # Mapping Fare
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare']                          =
0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] = 1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare']   =
2
    dataset.loc[ dataset['Fare'] > 31, 'Fare']                             =
3
    dataset['Fare'] = dataset['Fare'].astype(int)

    # Mapping Age
    dataset.loc[ dataset['Age'] <= 16, 'Age']                        = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age']                            = 4

# Feature Selection
drop_elements = ['PassengerId', 'Name', 'Ticket', 'Cabin', 'SibSp','Parch',
'FamilySize']
train = train.drop(drop_elements, axis = 1)
train = train.drop(['CategoricalAge', 'CategoricalFare'], axis = 1)

test  = test.drop(drop_elements, axis = 1)

print (train.head(10))
```

```
train = train.values
test    = test.values
```

House Prices: Advanced Regression Techniques

Problem description

Everyone wants a dreamed house, but different levels of houses have various prices. Housing prices are determined by multiple factors including the environmental condition, location, neighborhood and so on. Within the dataset, there are 79 explanatory variables can be used to predict the price of each house in Ames, Iowa.

The following is a simple description provided for these 79 variables:

MSSubClass: Identifies the type of dwelling involved in the sale.

MSZoning: Identifies the general zoning classification of the sale.

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Alley: Type of alley access to property

LotShape: General shape of property

LandContour: Flatness of the property

Utilities: Type of utilities available

LotConfig: Lot configuration

LandSlope: Slope of property

Neighborhood: Physical locations within Ames city limits

Condition1: Proximity to various conditions

Condition2: Proximity to various conditions (if more than one is present)

BldgType: Type of dwelling

HouseStyle: Style of dwelling
OverallQual: Rates the overall material and finish of the house
OverallCond: Rates the overall condition of the house
YearBuilt: Original construction date
YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)
RoofStyle: Type of roof
RoofMatl: Roof material
Exterior1st: Exterior covering on house
Exterior2nd: Exterior covering on house (if more than one material)
MasVnrType: Masonry veneer type
MasVnrArea: Masonry veneer area in square feet
ExterQual: Evaluates the quality of the material on the exterior
ExterCond: Evaluates the present condition of the material on the exterior
Foundation: Type of foundation
BsmtQual: Evaluates the height of the basement
BsmtCond: Evaluates the general condition of the basement
BsmtExposure: Refers to walkout or garden level walls
BsmtFinType1: Rating of basement finished area
BsmtFinSF1: Type 1 finished square feet
BsmtFinType2: Rating of basement finished area (if multiple types)
BsmtFinSF2: Type 2 finished square feet
BsmtUnfSF: Unfinished square feet of basement area
TotalBsmtSF: Total square feet of basement area
Heating: Type of heating
HeatingQC: Heating quality and condition
CentralAir: Central air conditioning
Electrical: Electrical system
1stFlrSF: First Floor square feet
2ndFlrSF: Second floor square feet
LowQualFinSF: Low quality finished square feet (all floors)
GrLivArea: Above grade (ground) living area square feet
BsmtFullBath: Basement full bathrooms
BsmtHalfBath: Basement half bathrooms
FullBath: Full bathrooms above grade
HalfBath: Half baths above grade
Bedroom: Bedrooms above grade (does NOT include basement bedrooms)
Kitchen: Kitchens above grade
KitchenQual: Kitchen quality
TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
Functional: Home functionality (Assume typical unless deductions are warranted)
Fireplaces: Number of fireplaces
FireplaceQu: Fireplace quality

GarageType: Garage location
GarageYrBlt: Year garage was built
GarageFinish: Interior finish of the garage
GarageCars: Size of garage in car capacity
GarageArea: Size of garage in square feet
GarageQual: Garage quality
GarageCond: Garage condition
PavedDrive: Paved driveway
WoodDeckSF: Wood deck area in square feet
OpenPorchSF: Open porch area in square feet
EnclosedPorch: Enclosed porch area in square feet
3SsnPorch: Three season porch area in square feet
ScreenPorch: Screen porch area in square feet
PoolArea: Pool area in square feet
PoolQC: Pool quality
Fence: Fence quality
MiscFeature: Miscellaneous feature not covered in other categories
MiscVal: $Value of miscellaneous feature
MoSold: Month Sold (MM)
YrSold: Year Sold (YYYY)
SaleType: Type of sale
SaleCondition: Condition of sale

There are 3 datasets
test csv file: 1459 observations and 80 variables(including Id)
train csv file: 1460 observations and 81 variables(including Id and SalePrice)
sample_submission: two columns(Id and SalePrice)
In this report, I will use randomforest to predict the SalePrice for test and using sample_submission to store my prediction.

Baseline method

```
library(randomForest)


train <- read.csv("train.csv",header=TRUE,stringsAsFactors=FALSE)
test <- read.csv("test.csv",header=TRUE,stringsAsFactors=FALSE)
submission <-
read.csv("sample_submission.csv",header=TRUE,stringsAsFactors=FALSE)

# Deal with missing values
variables <- names(train)
num_var<-names(data.frame(train[which(sapply(train,is.numeric))]))
```

```
cat_var<-names(data.frame(train[which(sapply(train,is.character))]))
num_var = num_var[num_var!='SalePrice']

for(i in num_var)
{
  if(any(is.na(train[[i]])))
  {
    train[[i]][is.na(train[[i]])] <- mean(train[[i]],na.rm=TRUE)
    }
  if(any(is.na(test[[i]]))){
      test[[i]][is.na(test[[i]])] <- mean(test[[i]],na.rm=TRUE)
    }
}


for(i in cat_var)
{
  if(any(is.na(test[[i]]))){
    test[[i]][is.na(test[[i]])] <- "MISSING_VALUE"
  }
  if(any(is.na(train[[i]]))){
    train[[i]][is.na(train[[i]])] <- "MISSING_VALUE"
  }

}

# for category variable, us it as factor
for(i in cat_var)
{
    levels <- sort(unique(c(train[[i]],test[[i]])))
    train[[i]] <- factor(train[[i]],levels=levels)
    test[[i]] <- factor(test[[i]],levels=levels)
}

Randf <- randomForest(SalePrice~.,train)

submission$SalePrice <- predict(Randf,test)

write.csv(submission,file="submission.csv",row.names=FALSE)
```

description of my baseline: first fill in the missing data, and then change categorical variable as factors since when we use randomForest regression to predict the SalePrice, we need numerical data.

Advanced baseline from the Kaggle community and Improvement

https://www.kaggle.com/nithum/house-prices-advanced-regression-techniques/simple-linear-regression-demo

```
corr = train.select_dtypes(include = ['float64', 'int64']).corr()
corr['logSalePrice'].sort_values(ascending = False)

model = LinearRegression()

model_rf = RandomForestRegressor(n_estimators= 25)
```

In this advanced baseline, the author using linear regression and RandomForest (2 models) to predict the sale price. It seems that in this analysis, depending on the correlation, the author only choose 5 variables to predict. However, with so many variables, the price may be predicted well by combination of many variables.

Lesson Learned

For different situation, we need to use different algorithms, for example, the Housing Price Project, we need to calculate a real value from given variables, which means we should use regression to predict SalePrice. During our course, we have the opportunity to know a stronger regression model – random forest, so we use it here. Also, for Titanic project, we have to deal with data with missing values. We can either delete those data points, or we can generate a random value according to its mean and standard deviation. After this project we also got more understanding about random forest method. Also, we learnt that before using these machine learning algorithms, it is more important to analyze variables, e.g. which variable should be used to predict dependent variable , which variable should be ignored and how to deal with missing data.

NOTE: Since we did this project in campus lab together, all work and submissions in github are done by both of us, and for sure we contributed equally.

Tasks:
Zhiqian Chen: Design and improve the baseline solution for HousePricing project, and wrote R code to apply RandomForest, and made improvement on Titanic project's R code.

Lixiang Wei: Wrote Python code to visualize data. Twisted R code to apply KNN, and came up with the idea of generating radom values.