Report: Assignment 1 - basic

group 27

20 April 2018

Task 2: Kaggle Competition: Titanic: Machine Learning from Disaster

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. This sensational tragedy shocked the international community and led to better safety regulations for ships.

The aim of the competition is to predict who among the passengers and crew was more likely to survive than others. Kaggle provides two datasets: *train* and *test*. While both datasets reference to passengers details, only *train* dataset contains infromation if passenger survived or not. Our goal is to predict which passenger from *test* dataset survived the sinking of Titanic.

Preparation

As mentioned before, Kaggle has provided two seperate datasets: *train* and *test*. Both datasets contain details about passengers and their trip. The only defference between them is *Survived* column in *train* dataset that indicates if passenger has survived the Disaster. That dataset will be used to train our models.

Before attempting to perform predicions, we focused on given data and tried to retrieve more interesting facts based on Feature Engineering. Because the only column that differ is *Survival*, for further processing we decided to datane both datasets into big one. Such an approach allows us to perform more adequate data analyse as we have a full insight of traveling passengers.

Data exploration

To have a better insight into assignment, we had to explore given data. That's the most important step - we have to be aware of all the details to work efficiently. Whole dataned dataset contains 1309 records (passengers) with 12 variables. In this part we will take a closer look to every attribute.

In datasets we can distinguish several (12) columns:

- Survived - indicated if given passenged survived
- PassengerId - passenger index in dataset
- Pclass - the ticket class (1,2,3)
- Name - full name of passenger, including their title
- Sex - sex of passenger
- Age - age of passenger
- SibSp - number of siblings or spouses traveling with passenger
- Parch - number of parents or childern traveling with passenger

- Ticket - ticket number
- Fare - passenger fare
- Cabin - passenger's cabin number
- Embarked - port of embarkation (C = Cherbourg, Q - Queenstown, S = Southampton)

Lets make a closer look into several variables.

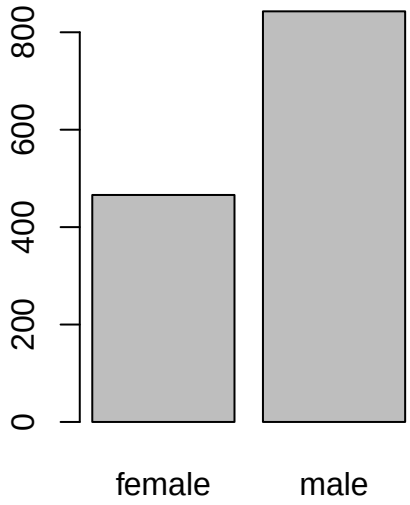
Name

In given dataset we can see that *Name* attribute contains string with passenger's name, surname and title.

example: *Allison, Master. Hudson Trevor*

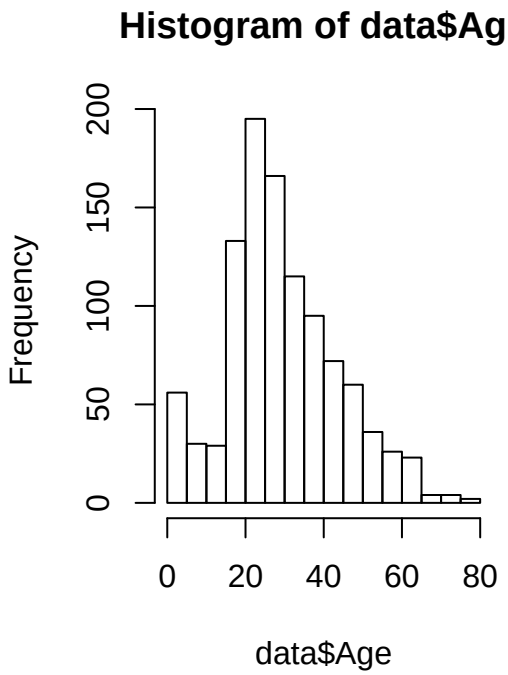
Fortunately, all rows in *Name* column follow the same string pattern (*surname*, *title first name*). Thanks to this fact, we will be able to retrieve more additional infrmation about passengers, like common surnames or titles.

Sex



Investigating Sex attribute we can see that there were 466 females and 843 males onboard. That gives us the first easy grouping of passengers.

Age



Regarding Age attribute, we can see that this variable varies up to 80 with mean around 23

Feature Engineering

After investigation of given dataset we can distingush columns that seem to be useful for further processing to retrieve even more data. In such an approach we are able to create additional columns with relevant variables that sould result in better prediciton accuracy.

Feature: Title

As mentioned before, Name column contains not only name and surname of passenger but also a title (like Sir., Mr., Mrs., ...). Following common pattern (*surname*, *title first name*) we can retrieve additional Column in our dataset that would group our passenger by Title. In addition, groups of unique similar titles were replaced by the tame variable (like 'Capt', 'Don', 'Major', 'Sir' => 'Sir').

As a result we obtained a fixed set of values with 11 levels.

Feature: Family

Basing on variables *SibSp* (number of siblings or spouses), *Parch*(number of parents or child) and Surnames retrieved from *Name* variable we are able to group passengers by families. Assuming that during disaster, every person takes care about their relatives, we think that it can be a significant factor in predicitons.

Our assumptions:

- the number of relatives with who each passenger was traveling is aclculated as follows: *SibSp + Parch + 1* - result is family size
- if family size is less or equal 2 we assume that the value is not revelant and we mark sucha afamily as *n/a*

As a result we obtained Family attribute with 97 levels.

Feature: Deck

Analysing *Cabin* attribute we figured out that each cabin number consists of Deck Level and Room number (like C40 => Deck C, Room 40). Because Deck Level could play important role in evacuation, we assumed thet it's a siginicant attribute. We decided to create a new attribute called Deck and we assigned relevant Deck Level to each passenger. Unfortunately, not every passenger had a Cabin number assigned, in such a case we marked Deck as 'U'.

Result:

```
##    A    B    C    D    E    F    G    T    U
##   22   65   94   46   41   21    5    1 1014
```

Feature: TicketType

Looking into ticket numbers we can see that some tickets have common prefix that could refer to Ticket Type of place of purchase (example: STON/02 42342). We decided to retrieve that ticket prefix and create a new attribute for each passenger. If ticket didn't have any prefix, we marked TicketType as 'num'.

As a result we obtained TicketType factor with 51 levels.

Missing values

We have found that some records lack in Age attribute. In such a situationw e decided to use a Decision tree to predict missing Age values. As siginicant factors we marked attributes: Pclass, Sex, FamilySize, Embarked, Title, SibSp, Parch.

Also Fare column had some missing values. In such a case we replaces missing values with median of all ticket Fares.

Classification and evaluation

By analysing our data and engineering some additional features we have enriched our dataset.

Within all calumns we decided that only few of them play siginificant role in predictions.

Chosen factors: *Pclass, TicketType, Sex, Deck, Age, SibSp, Parch, Fare, Embarked, Title, FamilySize, FamilyID*

Creating a steup

To evaluate classifiers we will need to create a proper setup. In this case we decided to use *train* data from Kaggle as it contains *Survived* column. For evaluation purposes we decided to split the data for training and testing sets with ratio 70/30 (70% - training, 30% - testing). While splitting the data we based on random ordering.

For evaluation we decided to use two non-linear algorithms: k-Nearest Neighbour Classification and Conditional inference trees. Both classifiers were trained and tested with the same sets of data. For evaluation analysis we used Confusion Matrix.

Factors that we took into account:

- Accuracy - how well results were predicted
- 95 CI - confidence intervals, our final score should match into calculated intervals
- Kappa - accuracy through random predicitons
- F1 - model that takes recall and precision into account

Evaluation of k-Nearest Neighbour Classification

Accuracy : 0.6929
95% CI : (0.6338, 0.7477)
Kappa : 0.3338
F1 : 0.5638

Evaluation of Conditional inference trees

Accuracy : 0.809
95% CI : (0.7566, 0.8543)
Kappa : 0.5929
F1 : 0.7437

Kaggle Submission

For Kaggle competition we decided to use Conditional inference trees as it gives us higher results in included evaluation factors.

We have submited our Prediction in Kaggle system and obtained satisfactory result 0.82296 which is top 3% in leaderboard. This result also matches into expected Confidence Intervals calculated during evaluation.

Task 3: Research and theory

Task 3a

Task 3b

Mean Squared Error(MSE)

The mean squared error (MSE) of an estimator measures the average of the squares of the errors that is, the difference between the actuals and predicted values.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

Where, $\hat{Y}_i$ is a vector of n predictions and Y is the vector of observed values of the variable being predicted.

Mean Absolute Error(MAE)

The mean absolute error(MAE) measures the mean of the absolute errors that is, the absolute value of the difference between the forecasted value and the actual value. MAE tells us how big of an error we can expect from the forecast on average.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

Where, $\hat{Y}_i$ is a vector of n forcasts and Y is the vector of actual values of the variable being predicted.

MSE Vs MAE

Mean squared error has the disadvantage of heavily weighting outliers. It is a result of the squaring of each term, which effectively weights large errors more heavily than small ones. Where this kind of property is undesirable, MAE can be used in those applications by the researcher.

When dealing with outliers, it might be helpful to use MAE instead of MSE since MSE gives higher error than MAE. Yet, MSE is more popular and efficient than MAE, because MSE punishes larger errors, which tends to be useful in the real world.

The mean absolute error (MAE) has the same unit as the original data, and it can only be compared between models whose errors are measured in the same units.

Both MSE and MAE are scale-dependent.For instance, if the observed data are in km then MSE is in km^2 and MAE is always in km respectively. Often, we need to perform accuray test on predicted values across different units. In that particular context, both MSE and MAE will not be applicable because they can only be compared between models whose errors are measured in the same units.

For evenly distributed errors that is, when all of the errors have the same magnitude, then Root mean squared error(RMSE) and Mean absolute error(MAE) will give the same result.If the square of the difference between actual values and forcasted values gives a positive distance which is same as their absolute distance then, MSE = MAE.

Data collection and exploration

To calculate MSE and MAE of different regression methods we used the *Energy_efficiency.csv* dataset.This dataset has been collected from the UCI MAchine Learning *Repository*[3]. This dataset is a collection of 768 samples and 8 features, aiming to predict two real valued responses.

The dataset contains the following eight attributes or features(X1,X2,....X8) along with two response variables(Y1,Y2):

- Relative Compactness(X1)
- Surface Area(X2)
- Wall Area(X3)
- Roof Area(X4)
- Overall Height(X5)
- Orientation(X6)
- Glazing Area(X7)
- Glazing Area Distribution(X8)
- Heating Load(Y1)
- Cooling Load(Y2)

It is important to implement energy efficiency in building to mitigate the impact of climate change. Due to the high demand for energy and unsustainable supplies, energy efficiency in building plays a vital role reducing energy costs and greenhouse gas emissions. Therefore, studying this dataset to evaluate how well energy is being used there to cut out the costs which will be helpful to have a ECO-friendly environment.

Experiment and perform evaluation

We load the samples into a dataframe and took all the column attributes as factor. We randomize the data frame using . Then, we divided the dataset into a trained dataset with the top 80% of the samples, and a tested dataset with the bottom 20% of the samples respectively. So, energy train data has first *614* entries from the dataset and energy test data contains the rest *154* samples.

At first we set up a model*(rt1)* for tree regression using the *Heating.Load* as outcome variable and all the eight attribtes as input variables and fit a new dataframe with the actual and predicted value of the model based on the test data. Using Regression Tree model(rt1) and "Heating.Load" as outcome, we calculated MSE = 6.59 and MAE = 2.101.

Similarly we fit another model(rt2) for tree regression but instead of using *Heating.Load* as outcome variable now we are intersted to use *Cooling.Load* as outcome variable. And we figured out for this model(rt2), using *Cooling.Load* we got MSE = 8.461 and MAE = 2.084.

We randomize the data frame using again. Next up we fit two models namely rf1 and rf2 respectively for both *Heating.Load* and *Cooling.Load* as outcome variables using Random forest regression following the same approach as described earlier for rt1 and rt2. Then we measured the MSE and MAE and for rf1 we got, MSE = 1.36 and MAE = 0.907.

```
##                            % Inc MSE
## Glazing.Area                    74.7
## Glazing.Area.Distribution       41.0
```

```
## Relative.Compactness           24.7
## Surface.Area                   24.1
## Wall.Area                      20.8
## Roof.Area                      18.9
## Overall.Height                 17.7
## Orientation                   -19.6
```

Observing the result of *importance()* function to calculate the importance of each variable, we got to see that *Glazing.Area* was considered the most important predictor; it is estimated that, in the absence of that variable, the error would increase by 74.7%.

Whereas for model rf2, using *Cooling.Load* we got MSE = 3.698 and MAE = 1.348.

```
##                            % Inc MSE
## Glazing.Area                   75.51
## Glazing.Area.Distribution      29.36
## Relative.Compactness           24.03
## Surface.Area                   22.02
## Roof.Area                      21.45
## Wall.Area                      20.14
## Overall.Height                 18.78
## Orientation                    -4.36
```

If we look into the *importance()* function to calculate the importance of each variable, we can see that The *Glazing.Area* was considered the most important predictor for *rf2.* it is estimated that, in the absence of that variable, the error would increase by 75.51%.

If we perform overall evaluation and compare MSE and MAE for all four models we can see that using random forest regression the model, rf1 with *Heating.Load* as response variable has lower error rate for both MSE = 1.36 and MAE = 0.907 compared to other models. For regression tree model both rt1 and rt2 produced relatively higher MSE values though MAE values did not varry significantly.

Task 3c

Theory: Analyze a less obvious dataset : SMS Spam Filtering

Text message classification requires supervised Natural language processing techniques to filter messages with respect to it's types and maps inputs to it's targeted varaibles based on the learning information which it gets from trained data.

Our aim is to predict the probabilities of a message being spam or ham. Therefore, we need to perform text mining on unstructured data, fit a predictive model on top of that and suggest improvement if any to increase our proposed model's performance.

Data Colection

The dataset: *SmsCollection.csv* has been collected from the course website. This dataset is a collection of 5574 text messages in English provided by a UK forum for research purpose. In this

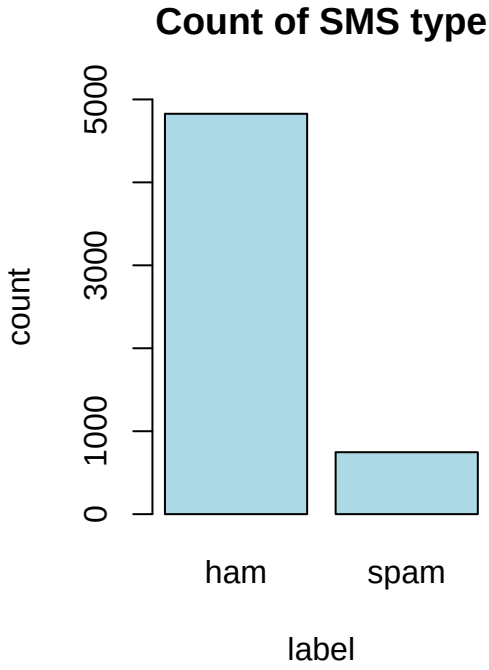
dataset, messages are labeled as either *spam* or *ham*. *Ham* stands for legitimate message whereas the type *spam* is used for trashed or unwanted message.

At first we load the data from the source. Then we split label and text and bind them into a dataframe.

Data exploration

The *SmsCollection* dataset contains text messages only. Since we are only dealing with text messages which are unstructured in nature, so we will need to perform some basic natural language processing technique in order to tokenize those texts, computing the frequencies of words, calculating document-feature matrix and so on.

In general, almost all the classifiers use a conditional probability model to classify data. Looking at the samples we can see that they are mainly concerning about classifying the messages into a two class problem as spam or ham. Among 5574 text messages there are 4827 messages categorized as ham and the rest 747 messages are classified as spam. We generate a barplot of it.



As we can observe there are more ham messages than spam There are various classifier algorithms to solve this but we found Naive Bayes as the most suitable one for this purpose. Naive Bayes is a simple yet powerful classifier based on Bayes probability theorem which uses conditional probabilty model. It is more suited to categorial variables although it can also be used for continuous variables. Text messages are often noisy and the amount of predictors are way more than the actual samples. Naive Bayes classifier follows conditional independence theorem. Therefore, it assumes that features are independent of one another which is a high bias and this introduced strong bias might be helpful in reducing the variance to achieve better predictions.

Data Processing and transformation

We load the samples into a dataframe and use the *label* as a factor while on the otherhand we are using attribute *text* as character.And then we randomize the data frame using . To process the text data we transformed the data frame into a volatile corpus as they cannot be directly handled by a data frame. VCorpus converted each of the messages as a document.

In the VCorpus text document each SMS has it's content in raw formatted way. So, before applying Naive Bayes classification algorithm we need to clean up data. It will help the algorithm to perform more efficiently which will eventually increase the accuracy of predictions.

Our data cleaning process includes : conversion of all texts to lowercase, removal of numbers that is neither a spam nor a ham, removal of some common stop words in english such as: "a","an","the","for" etc. that neither indicate spam or ham, punctuation and extra whitespace removal. Finally after completing data cleaning task, final version of the VCorpus were transformed into a Document-Term-Matrix (DTM) that will be taken into account as the basis for the classification.Doing so, we found 7713 unique terms in total for all 5574 entries.

Generating training and testing dataset

We divided the DTM to generate our training and testing dataset. The Document-term-matrix is splited into a trained dataset with the top 75% of the raw sms data, and a tested dataset with the bottom 25% of the raw sms data using the *createDataPartition()* function. Since, we only need "label" attribute of the raw sms dataset we created two classifier labels namely "sms train labels"" and "sms test labels" by splitting with exact same proportions of row that we used before. We made these two classifier labels to use them for Naive Bayes model later on.

Table 5: Frequency comparison among different datasets based on SMS label

	Raw Dataset	Training Dataset	Test Dataset
ham	86.6	86.6	86.6
spam	13.4	13.4	13.4

Using prop.table() we converted number of spam/ham messages of both sms train and test labels into fractional values and preserved those proportions into our train and test dataset. Looking into the above table we can see that 86.6% of the messages correspond to legitimate messages (ham) and 13.4% to spam messages which follows the same proportion in each of our dataset perfectly.

We created a wordcloud from the cleaned vcorpus to look at the most frequent words in the available bag of words. We also created seperate wordclouds for spam and ham messages where most frequent words appeared in larger font and less frequent words in smaller font.

Looking the above wordclouds we found that the spam contains “call”, “now”, “free”, “mobile” as most frequent words whereas ham contains frequent words such as “will”, “get”, “now”,“just”, “can”. Also spam(on left) showed extreme frequency in it’s wordcloud. Since, it seemed that our datasets contains distintive words, we hope our choosen classifier algorithm(Naive Bayes) will be a good fit for sms prediction.

We removed most of the least frequent words from the DTM and created final train and test dataset that we should be using for the training using only the most frequent words that appeared at least 5 times in datasets. The number of columns for each trained and tested datasets are then shrink from 7713 terms to 1193 column (words).

Training the data using Naive Bayes Model

we already have trained and tested labels respective to the datasets.And we used naive_bayes() to train and build model based on the trained dataset along with it’s trained label. Our trained model contained information from both trained and tested DTM whcih have 1193 distict words(possibilities of either spam/ham).

Evaluate performance

evaluating it’s performance we can see that Naive Bayes has accuracy rate 96.77% with sensitivity 78.49% and there are 5 *spam* text messages wrongly classified as ham and 40 *ham* text examples wrongly classified as spam. In order to improve it’s performance we used Laplace along with it and laplace lowered both of the false positive and false negative values and increased our accuracy up to 97.56%. The summarised version of their performances are given below into tabular form.

	True_Neg	True_Pos	False_Neg	False_Pos	accuracy	sensitivity

Table 6: Performance Table for two models

	True_Neg	True_Pos	False_Neg	False_Pos	accuracy	sensitivity
Model-1:[Naive Bayes]	1201	151	35	5	0.9713	0.8118
Model-2:[Naive Bayes+laplace]	1204	163	23	2	0.982	0.8763

Conclusion

To solve this task we classified text messages as ham or spam using some basic natural language processing and then model a naive Bayes text classifier. There are numerous ways of doing this but using the Naived Bayes classfication algorithm, we obtained more than 97% accuracy in predicting whether a new incoming message is a spam or not based on it's training data.

References

1. https://en.wikipedia.org/wiki/Naive_Bayes_classifier
2. https://en.wikipedia.org/wiki/Naive_Bayes_spam_filtering
3. https://www.r-bloggers.com/understanding-naive-bayes-classifier-using-r/
4. https://datascienceplus.com/text-message-classification/
5. https://en.wikipedia.org/wiki/Mean_squared_error
6. https://en.wikipedia.org/wiki/Mean_absolute_error
7. http://archive.ics.uci.edu/ml/datasets/Energy+efficiency#
8. A. Tsanas, A. Xifara: 'Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools', Energy and Buildings, Vol. 49, pp. 560-567, 2012 (the paper can be accessed from weblink)