STAT 457 Project II — Genre Classification on IMDB Dataset

Bofan Sun
20108143
17bs48@queensu.ca

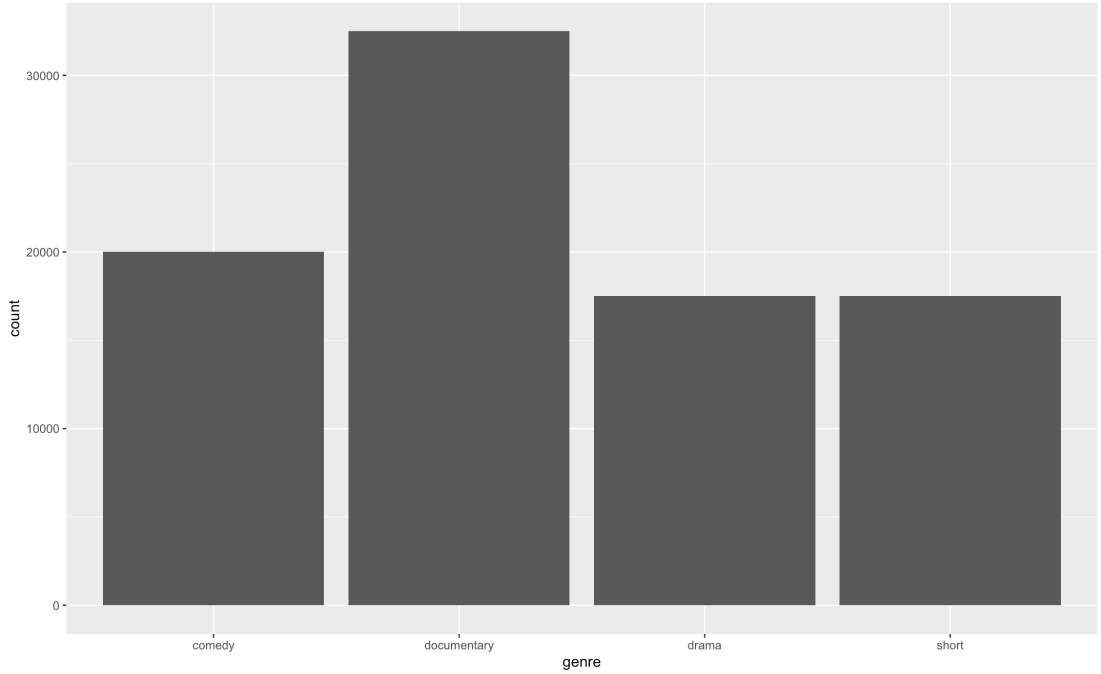
Introduction

The goal of this project is to analyze a subset of data derived from the IMDB dataset, and predict the genre of each movie in the test dataset based on a brief description provided by building models. I start by using glimpse() and summary() to have a brief summary of the dataset.

```
> glimpse(train_file)
Rows: 10,000
Columns: 3
$ id          <int> 1722, 7915, 5434, 4640, 9590, 8880, 9638, 4225, 1174, 5574, 9048, 8328, 7027, 8274, …
$ genre       <chr> " documentary ", " documentary ", " documentary ", " comedy ", " comedy ", " drama "…
$ description <chr> " Quality Control consists of a series of 16mm single take shots filmed in the summe…
> summary(train_file)
       id           genre           description
 Min.   :    1   Length:10000       Length:10000
 1st Qu.: 2501   Class :character   Class :character
 Median : 5000   Mode  :character   Mode  :character
 Mean   : 5000
 3rd Qu.: 7500
 Max.   :10000
>
```

Exploratory Data Analysis

Then, in order to closely observe the dataset, I did data visualization on the train dataset by plotting some graphs. I draw a histogram of the four types of genres: comedy, documentary, drama and short, and compare the number of movies with each genre. This is what I got:



We can see from this graph that most movies are documentary type, and the number of movies with comedy, drama or short genre are very close.
Next, I did some preprocessing to the dataset follow the example code provided. Then, I prepared training dataset and testing dataset for the following modelling by splitting the train dataset gained from preprocessing into train and test by about 20% as test and rest of them as train. The following is the dataset used in future modelling which we will discuss later.

```
#create training and test data in the form of matrices and vectors
Xtrain = as.matrix(comb_mat[1:nrow(train_file),])
Ytrain = train_file$genre

Xtest = as.matrix(comb_mat[(nrow(train_file)+1):(nrow(train_file)+nrow(test_file)), ])
testID = test_file$id #use for output

trainID = sample(1:nrow(Xtrain), floor(0.8*nrow(Xtrain)))
Xtrain_dat = Xtrain[trainID, ]
Xtest_dat = Xtrain[-trainID, ]

Ytrain_dat = Ytrain[trainID]
Ytest_dat = Ytrain[-trainID]
```

Prediction Models and Kaggle Scores

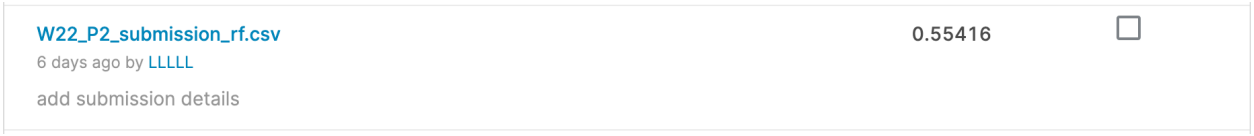
First Model —— Random Forest by Ranger

My first model is random forest. I first specify the tuning method by the trainControl() function from caret package with 5-fold cross-validation, and then set the range of

tunning parameter mtry from 2 to 20, with each of them is added by 2 by the former one. Then train the model with method “ranger”.

```
#model 1: random Forest
ctrl <- trainControl(method = "cv",
                     number = 5)

rf.Grid = expand.grid(mtry = 2*(1:10),
                      splitrule ='gini',
                      min.node.size = 1)

rf.cv.model <- train(Xtrain_dat, Ytrain_dat,
                     method = "ranger",
                     trControl = ctrl,
                     tuneGrid = rf.Grid)
rf.cv.model
```

Then we do predictions using the “best” model selected, and check predictive performance, which is around 0.55. Lastly, applying random forest to the test file, and generate a csv file and submit it to Kaggle. I got my best result for this model as 0.55416.

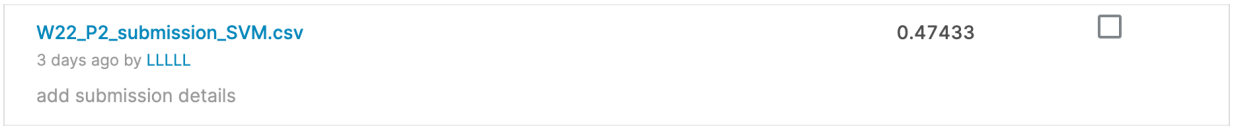


Second Model —— Support Vector Machines (SVM)

My second model is SVM with linear kernal. First, I specified the grid with 5-fold cross-validation, and use “caret” package to train the SVM model and select best tuning parameter. The optimal model have tuning parameters C = 70. This is what I got:

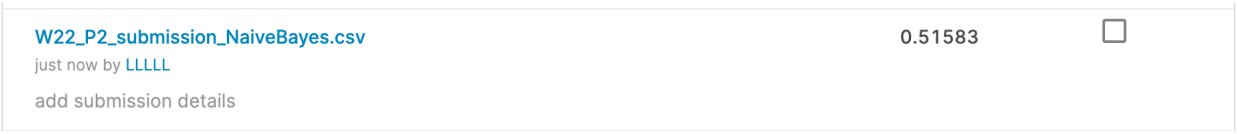
```
> lmSVM.cv.model
Support Vector Machines with Linear Kernel

7000 samples
 835 predictor
   4 classes: ' comedy ', ' documentary ', ' drama ', ' short '

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 5600, 5600, 5599, 5600, 5601
Resampling results across tuning parameters:

  C    Accuracy   Kappa
   10  0.4724286  0.2966876
   20  0.4690024  0.2921171
   30  0.4701452  0.2936522
   40  0.4718595  0.2959319
   50  0.4700026  0.2934696
   60  0.4731444  0.2976543
   70  0.4734307  0.2980359
   80  0.4674311  0.2900469
   90  0.4718571  0.2959372
  100  0.4705719  0.2942324

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 70.
>
```

Then we do predictions using the optimal model selected, and evaluate prediction performance. Lastly, applying random forest to the test file, and generate a csv file and submit it to Kaggle. I got my best result for this model as 0.47433, which has no improvement compared with previous model.



Third Model —— Naive Bayes

My third model is naive bayes. I fit the model with naiveBayes() function in "e1071" package. Then predict the genres using test dataset, and evaluate prediction performance. The best performance I got is 0.5375. Lastly, I applied naive bayes to the test file, and generate a csv file and submit to Kaggle. I got my result for this model as 0.51583, which improves a little bit compared with previous model.

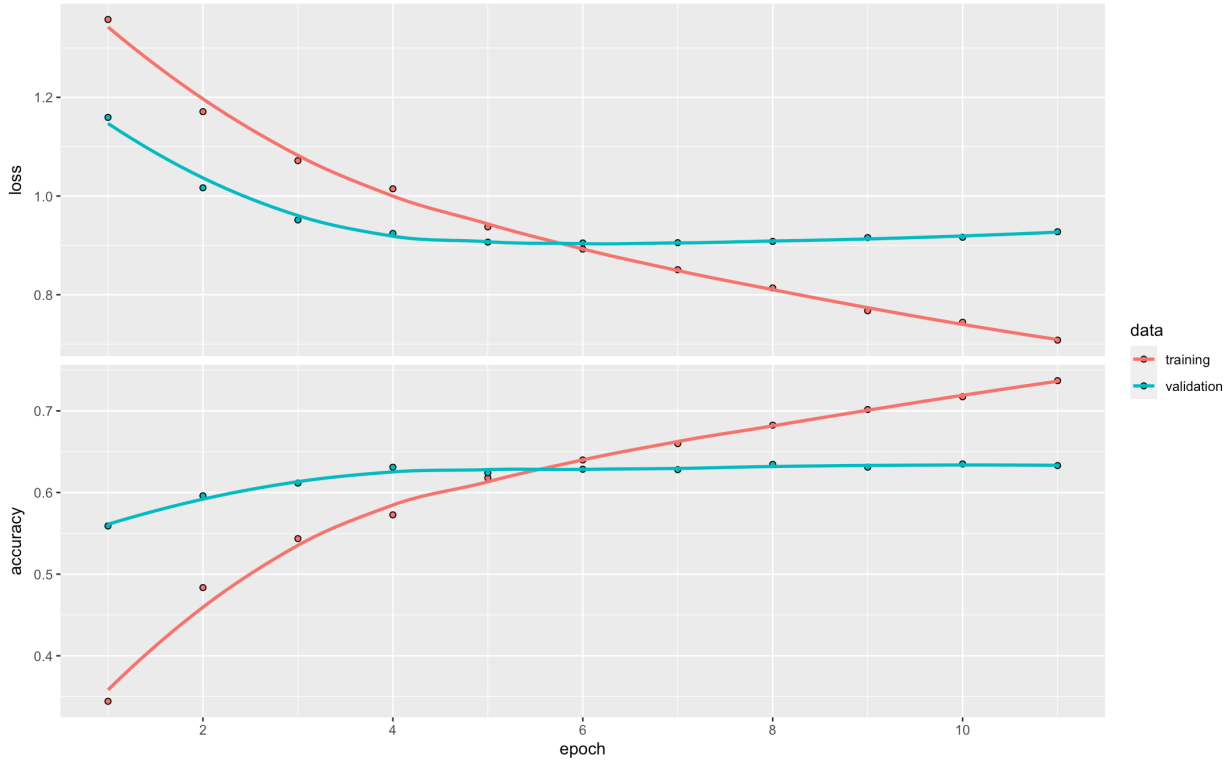


Fourth Model —— Neural Network

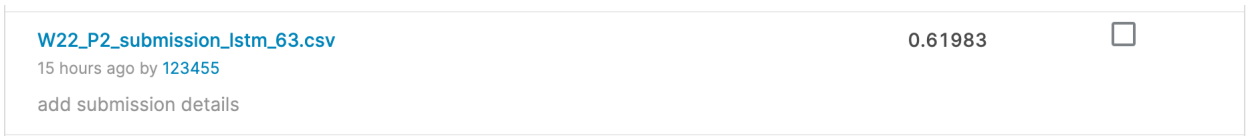
My fourth model is neural network. Since neural network only accept integer values, I first transform labels Ytrain into integers. With 0 as "comedy", 1 as "documentary", 2 as "drama" and 3 as "short". Then padding sequences Xtrain and Xtest to a matrix formed by integers and have the same length 6800. Then I build a Neural Network model using the keras_model_sequential() and then add layers. The first hidden layer has 512 units, the activation function is ReLU (rectified linear unit), and the second hidden layer has 128 units, with the same activation function as above. The last output layer has 4 units, the activation is the softmax function because our label "genre" has 4-classes.

```
> model
Model: "sequential_50"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_100 (Dense)           (None, 512)               3482112

 dropout_67 (Dropout)        (None, 512)               0

 dense_99 (Dense)            (None, 128)               65664

 dropout_66 (Dropout)        (None, 128)               0

 dense_98 (Dense)            (None, 4)                 516

=================================================================
Total params: 3,548,292
Trainable params: 3,548,292
Non-trainable params: 0
_________________________________________________________________
>
```

Then compile the model with “sparse_categorical_crossentropy” loss function since we encoded the labels as integers. Lastly, fit the model with training dataset. I set the total number of iterations over all training data “epochs” as 10 in order to avoid overfitting. And set the number of training examples in one iteration of a NN “batch_size” as 64. Then plot the model we got.



Next, do predictions on testing dataset. What we got here is a matrix with each row as a movie and each column represents a genre. The (i,j)-th entry is the probability for i-th movie with j-th genre. Use apply() function to choose the genre of each movie with largest probability and transform them as an integer (0,1,2,3 as before). Then, transform 0, 1, 2, 3 back to characters same as before, with 0 as “comedy”, 1 as “documentary”, 2 as “drama”, and 3 as “short”. Do prediction on the original test file again and write a csv file for kaggle submission. I got my best result for this model as 0.61983.



Summary

Model	Best Kaggle Score
Random Forest	0.55416
SVM	0.47433
Naive Bayes	0.51583
Recurrent Neural Network	0.61983

By the comparison above, we can see that Neural Network has the best performance.