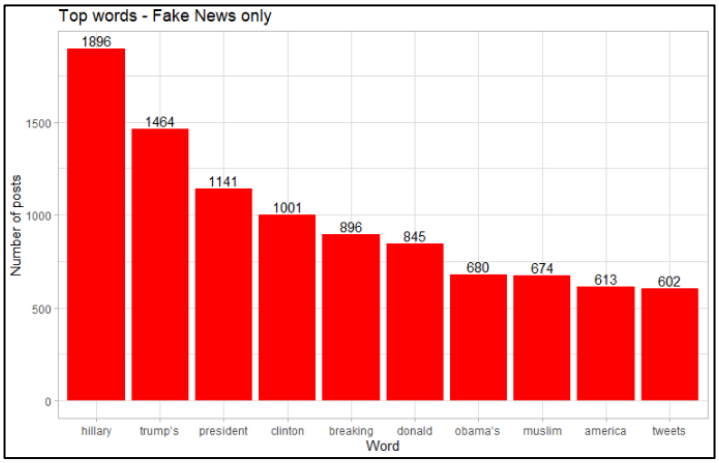
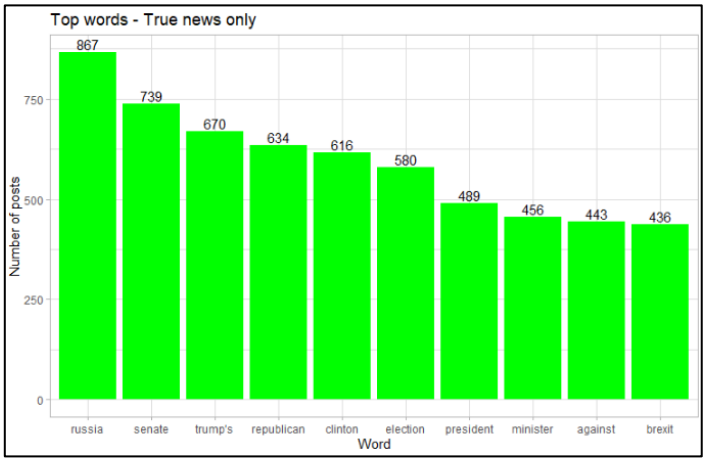
1. Exploratory Data Analysis:

- Assign new column “category” to both fake and true dataset where “0” is fake and “1” is true:

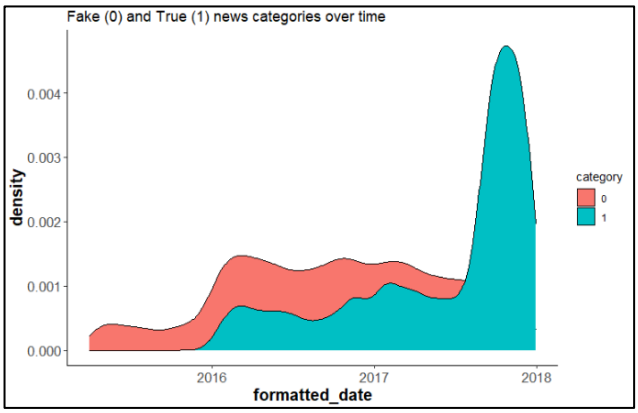
```
> glimpse(fake)
Rows: 23,481
Columns: 5
$ title    <chr> "Donald Trump Sends Out Embarrassing New Year's Eve Message; This is Disturbing", "Drunk Bragging Trump St~
$ text     <chr> "Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had to g~
$ subject  <chr> "News", "News", "News", "News", "News", "News", "News", "News", "News", "News", "News", "News", "N~
$ date     <chr> "December 31, 2017", "December 31, 2017", "December 30, 2017", "December 29, 2017", "December 25, 2017", "~
$ category <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0~
> glimpse(true)
Rows: 21,417
Columns: 5
$ title    <chr> "As U.S. budget fight looms, Republicans flip their fiscal script", "U.S. military to accept transgender r~
$ text     <chr> "WASHINGTON (Reuters) - The head of a conservative Republican faction in the U.S. Congress, who voted this~
$ subject  <chr> "politicsNews", "politicsNews", "politicsNews", "politicsNews", "politicsNews", "politicsNews", "politicsN~
$ date     <chr> "December 31, 2017", "December 29, 2017", "December 31, 2017", "December 30, 2017", "December 29, 2017", "~
$ category <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
```

- Top Words by "Title" (only words over 5 characters allowed):



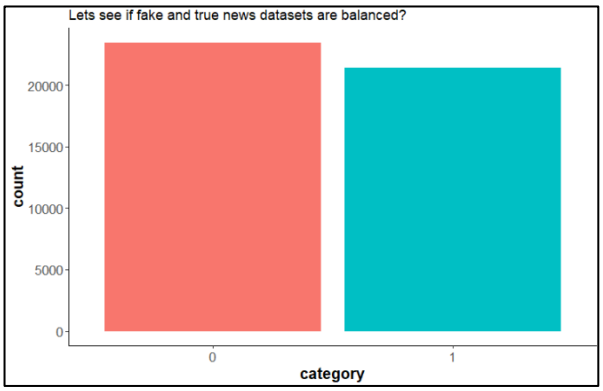


- Merge fake and true news and create a plot over time:



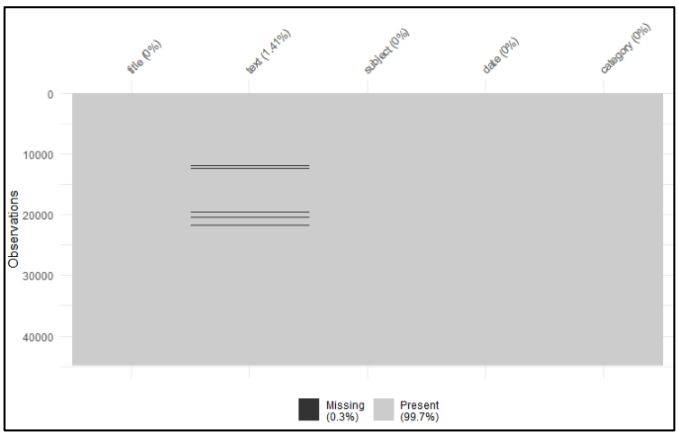
- Fake news are more frequent in 2016, 2017 and first half of 2018.
- In Q4 2018 fake and true news are balanced

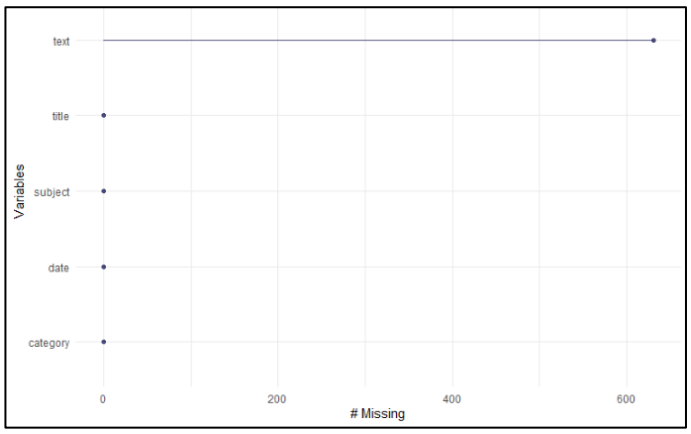
- Are datasets balanced?



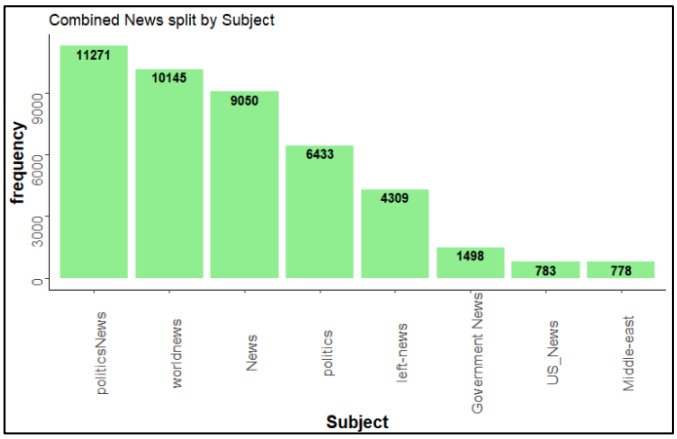
- The merged dataset is Balanced - this will make it easier for prediction

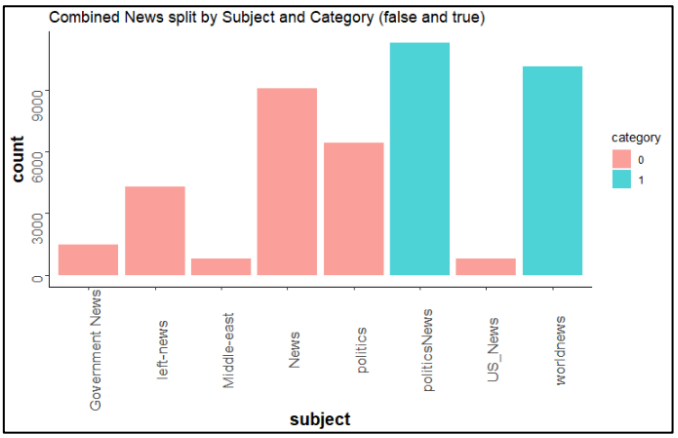
- There are 631 missing values in the “Text” column. They are removed.





- News count by each Subject, and, Subject by category plots:





- I have decided to add the “Title” inside the "Text" column for better identification and we don’t lose that data

```
> glimpse(news_combo)
Rows: 44,267
Columns: 3
$ text     <chr> "Donald Trump Sends Out Embarrassing New Year’s Eve Message; This is Disturbing Donald Trump just couldn~
$ category <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
$ ID       <chr> "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13", "14", "15", "16", "17", "18", "19",~
```

2. Pre-processing and data cleanup

- Create a corpus (type of object expected by "tm" library)
- Text to lower case
- Remove numbers
- Remove Punctuations
- Remove Stopwords
- Remove specific words (example: we should remove the name of the newspaper –“Reuters”-its on every news)

```
doc <- tm_map(doc, removeWords, c("reuters","video","image","monday","tuesday",
                                  "wednesday", "thursday","friday","saturday",
                                  "sunday","really","thing","month","year","something"))
```

- Remove Whitespace
- I have decided not to do “stem words” as it tweaks and cuts the words and they might lose their meaning
- Remove other punctuation issues (example: "[[:punct:]]")
- Lemmatization
- Create Document Term Matrix with control list (example: “wordLengths=c(5, 20)”)
- I enforced lower and upper limit to the length of the words included (between 5 and 20 characters) to speed up data processing and eliminate noise

- After that, I removed all terms whose sparsity is greater than the threshold. Sparsity dropped from 100% to 77% and term length dropped from 20 to 14 (example: “sparse = 0.85”)
- Convert DTM to matrix to DataFrame

3. Post-processing analysis

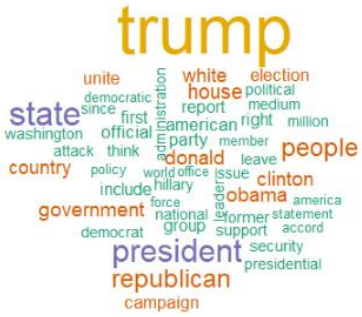
- Return all terms that occur more than 20,000 times in the entire corpus

```
> findFreqTerms(dtm.clean,lowfreq=20000)
 [1] "american"   "campaign"   "clinton"    "country"    "donald"     "election"   "government" "house"      "include"
[10] "obama"      "official"   "party"      "people"     "president"  "report"     "republican" "right"      "state"
[19] "trump"      "unite"      "white"
```

- Correlation limit inspection and associations among: “Trump”,” Obama”, “Russia”, “State” :

```
> findAssocs(dtm.clean, terms = c("trump","obama","russia", "state"), corlimit = 0.2)
$trump
       donald      campaign     president   presidential     republican          white          think administration
         0.63          0.35          0.32          0.29          0.28          0.23          0.21          0.20

$obama
     president administration
          0.37          0.25

$russia
numeric(0)

$state
     unite    include    country government   official      group washington     nation     policy   security    support
      0.51       0.28       0.26       0.26       0.26       0.24       0.24       0.23       0.23       0.22       0.22
    accord   national     public   american democratic
      0.21       0.21       0.21       0.20       0.20
```

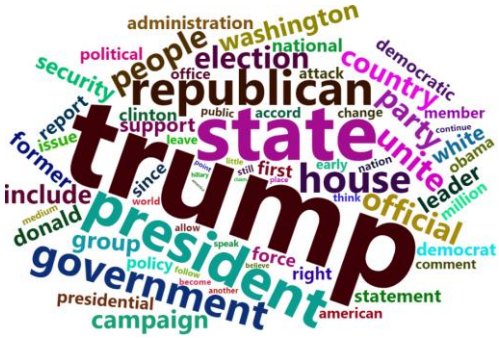
We can draw a simple word cloud by min. frequency of 10,000



Word cloud (based on term and frequency)

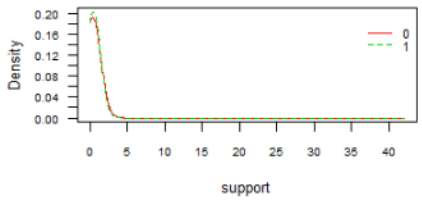
Word cloud for the Fake News (category==0)

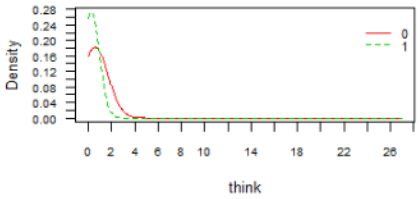
Word cloud for the True/Real News (category==1)

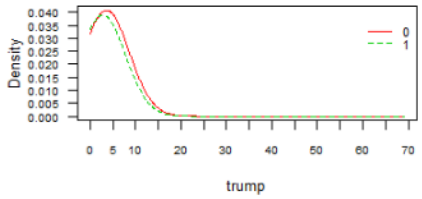


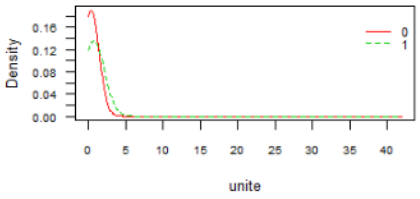
4. Modelling, Prediction, Performance

- Create 80:20 split for train and test
- Run **Naive Bayes Model**



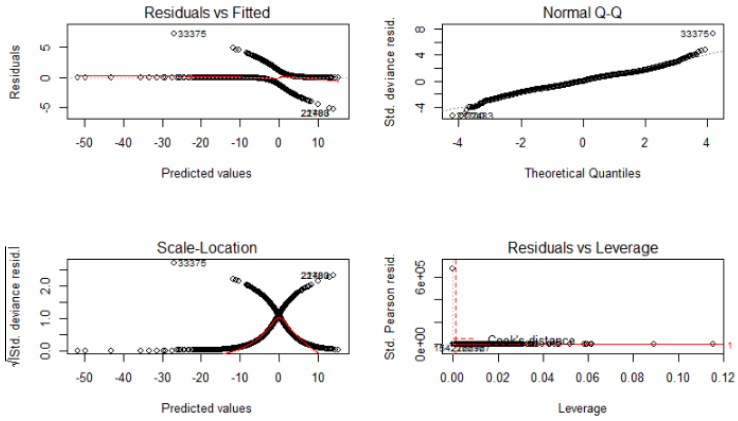






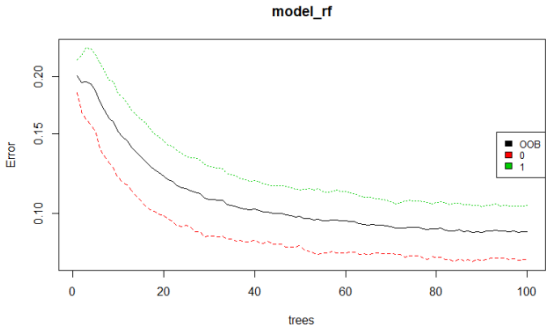

```
- Laplace: 0
- Classes: 2
- Samples: 35413
- Features: 67
- Conditional distributions:
    - Gaussian: 67
- Prior probabilities:
    - 0: 0.514
    - 1: 0.486
```

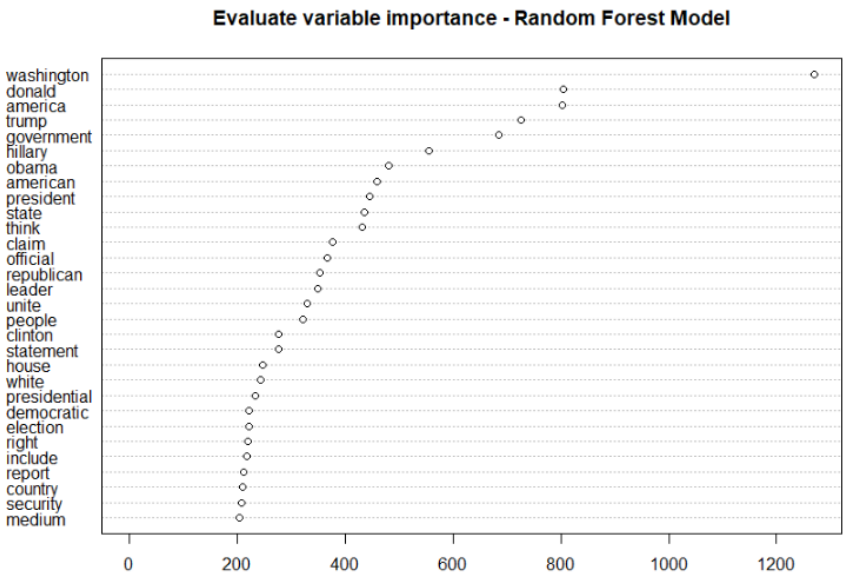
- Run **Logistic Regression Model**




```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-5.3118  -0.6676  -0.0142   0.7010   7.3274

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.0029757  0.0241568  -0.123  0.90196
accord         -0.0546252  0.0238420  -2.291  0.02196 *
administration  0.1473053  0.0207004   7.116 1.11e-12 ***
allow          -0.0833852  0.0279444  -2.984  0.00285 **
america        -0.9901574  0.0348893 -28.380  < 2e-16 ***
american       -0.3777000  0.0183395 -20.595  < 2e-16 ***
```

- Run **Random Forest Model**

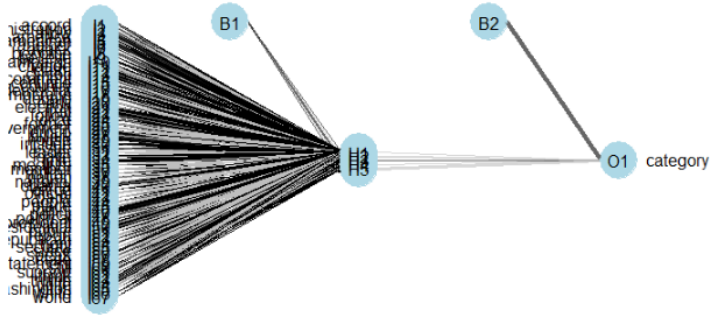




- Run **SVM**

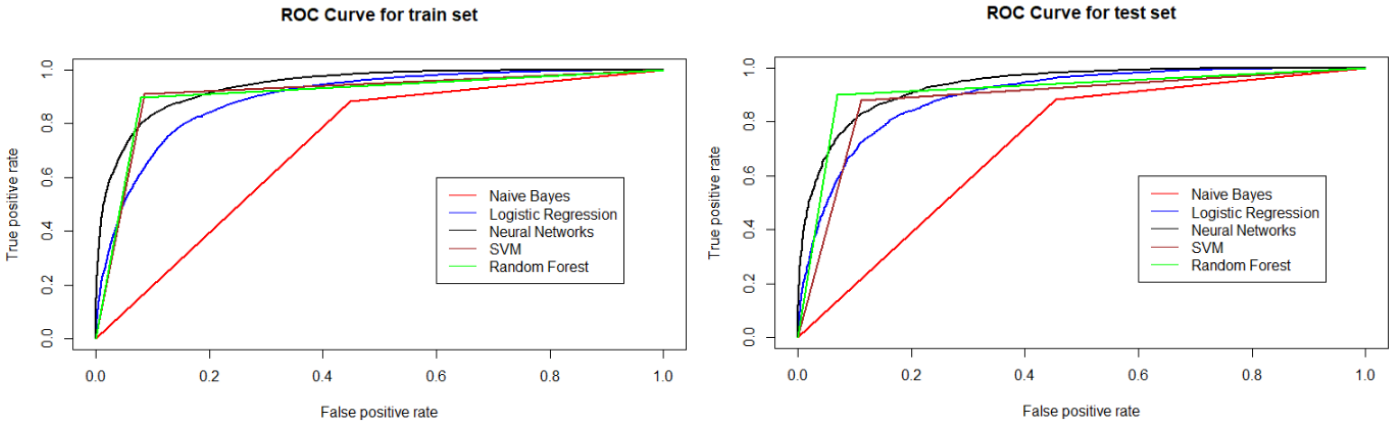
```
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1

Number of Support Vectors:  14862

 ( 7448 7414 )


Number of Classes:  2

Levels:
 0 1
```

- Run **NNET**

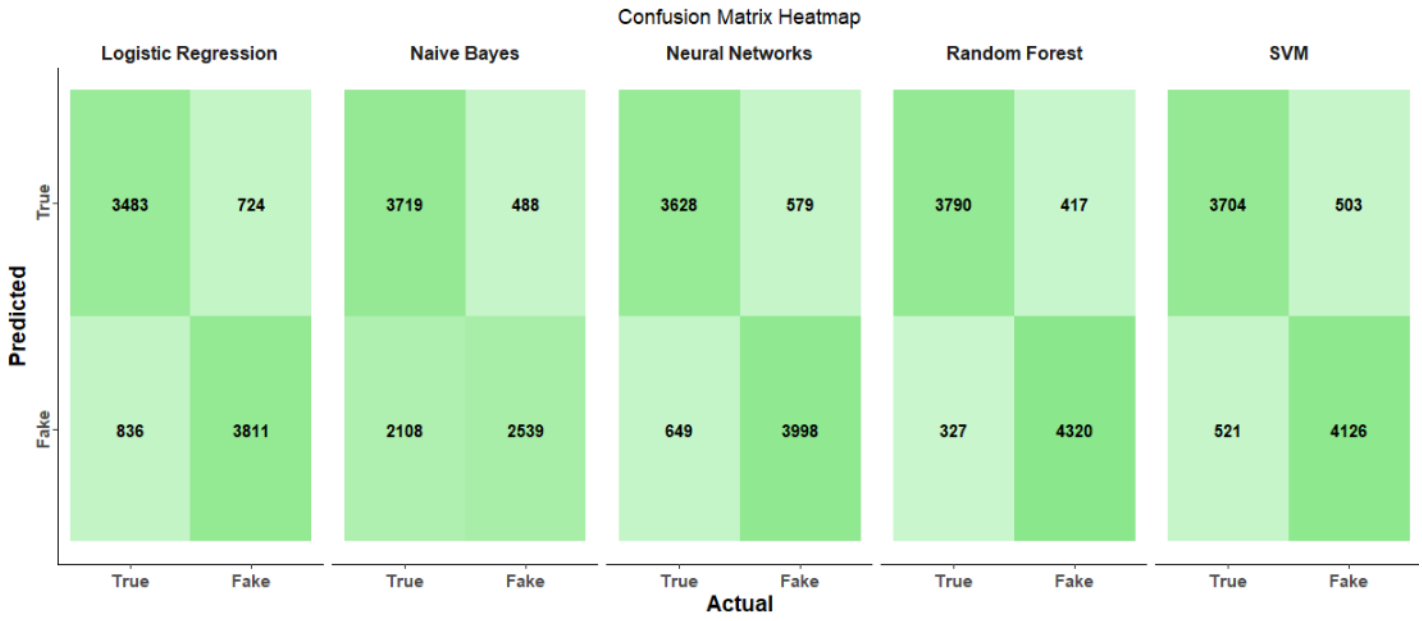



```
iter 330 value 11131.545766
iter 340 value 11121.944151
iter 350 value 11108.509880
iter 360 value 11081.698324
iter 370 value 11045.305201
iter 380 value 11019.456384
iter 390 value 10995.583696
iter 400 value 10982.555516
iter 410 value 10973.709274
iter 420 value 10968.158688
iter 430 value 10964.480650
iter 440 value 10962.553806
iter 450 value 10961.682231
iter 460 value 10960.679128
iter 470 value 10958.416922
iter 480 value 10955.502049
iter 490 value 10952.714821
iter 500 value 10952.036867
final  value 10952.036867
stopped after 500 iterations
```

- Evaluation: ROC



- Evaluation: Confusion Matrix



- Evaluation: Summary Table

```
> score
                Model  Accuracy  F1_Score
1         Naive Bayes 0.7067992 0.6617149
2 Logistic Regression 0.8238084 0.8301024
3      Neural Networks 0.8613056 0.8668690
4                 SVM 0.8843461 0.8896076
5       Random Forest 0.9159702 0.9207161
```

Table for Accuracy and F1 Score

Model	Accuracy	F1_Score
Naive Bayes	0.7067992	0.6617149
Logistic Regression	0.8238084	0.8301024
Neural Networks	0.8613056	0.8668690
SVM	0.8843461	0.8896076
Random Forest	**0.9159702**	**0.9207161**

Note:

Random Forest has the highest accuracy and F1 score.