Sarcasm Detector

Lidor Erez

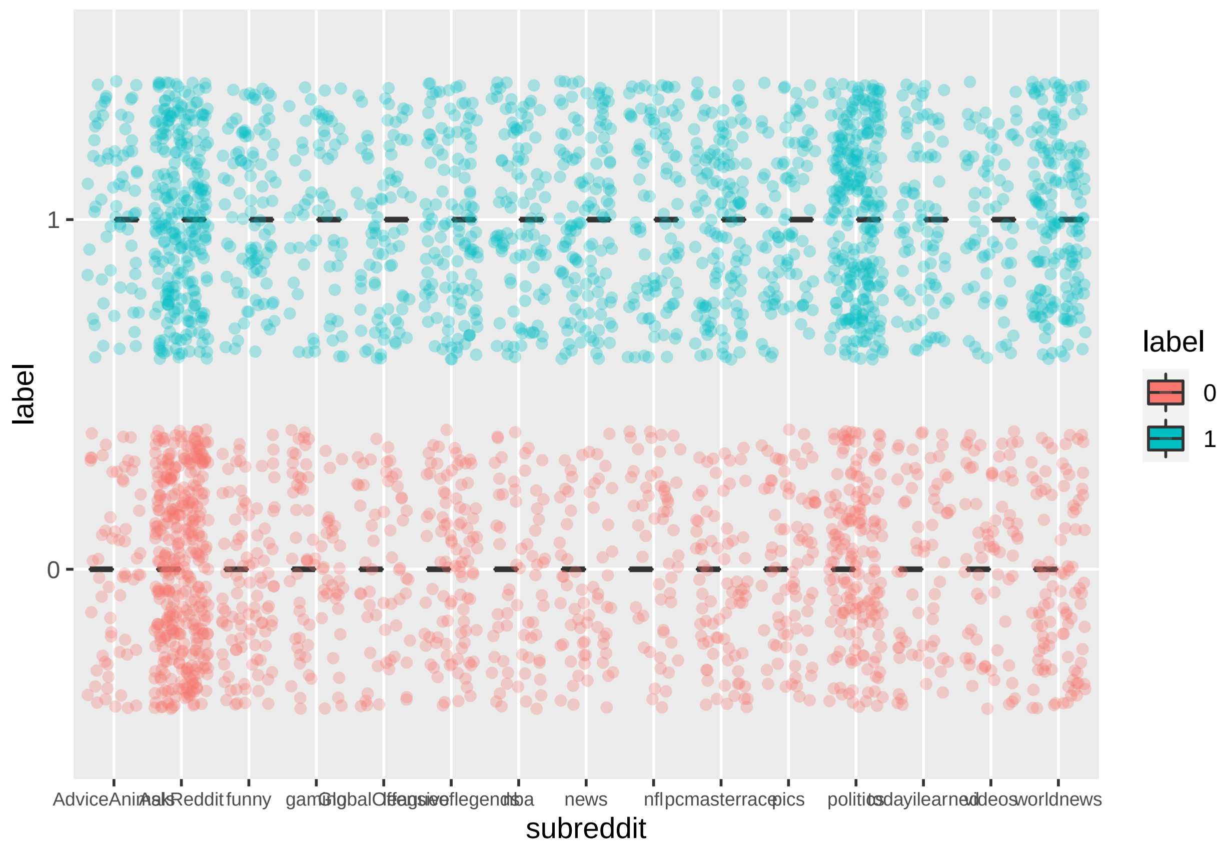
As a Statistics & Data Science student I was given a task to detect sarcasm in Reddit posts using supervised machine learning model. With the help of Kaggle as a source of data, I will try to Analyze the data and figure out which supervised machine learning model is better for this task.

```
str(sarcastic)
```

```
## 'data.frame':    3314 obs. of  29 variables:
##  $ label          : Factor w/ 2 levels "0","1": 1 2 1 1 2 2 2 1 1 2 ...
##  $ subreddit      : chr  "AskReddit" "todayilearned" "AskReddit" "videos" ...
##  $ charN          : int  96 111 82 21 113 30 13 25 46 17 ...
##  $ wordN          : int  21 18 17 4 24 6 3 5 8 4 ...
##  $ letterN        : int  75 89 62 18 83 24 10 20 38 12 ...
##  $ avg_word_length: num  3.57 4.94 3.65 4.5 3.46 ...
##  $ capsN          : int  2 3 2 1 7 2 2 1 1 4 ...
##  $ caps_ratio     : num  0.0267 0.0337 0.0323 0.0556 0.0843 ...
##  $ vowelsN        : int  30 30 22 8 30 10 4 8 16 3 ...
##  $ vowels_ratio   : num  0.4 0.337 0.355 0.444 0.361 ...
##  $ questionN      : int  0 0 0 0 0 1 0 0 1 0 ...
##  $ exclamationN   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ punctuationN   : int  1 3 2 0 1 1 1 1 1 2 ...
##  $ questions_ratio: num  0 0 0 0 0 ...
##  $ excs_ratio     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ puncs_ratio    : num  0.01042 0.02703 0.02439 0 0.00885 ...
##  $ fBombN         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ f_ratio        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ anger          : num  4.55 0 0 0 4 ...
##  $ anticipation   : num  9.09 4.76 5.26 0 4 0 0 0 0 0 ...
##  $ disgust        : num  9.09 0 0 0 4 ...
##  $ fear           : num  0 4.76 0 0 4 ...
##  $ joy            : num  9.09 4.76 0 25 4 0 0 0 0 0 ...
##  $ negative       : num  9.09 4.76 0 0 8 ...
##  $ positive       : num  9.09 9.52 0 25 4 0 0 0 0 0 ...
##  $ sadness        : num  0 4.76 0 0 8 ...
##  $ surprise       : num  9.09 0 0 0 8 0 0 0 0 0 ...
##  $ trust          : num  9.09 14.29 0 0 4 ...
##  $ concrete       : num  2.48 2.29 2.34 2.6 2.51 3.12 2.33 2.58 2.68 2.49 ...
```

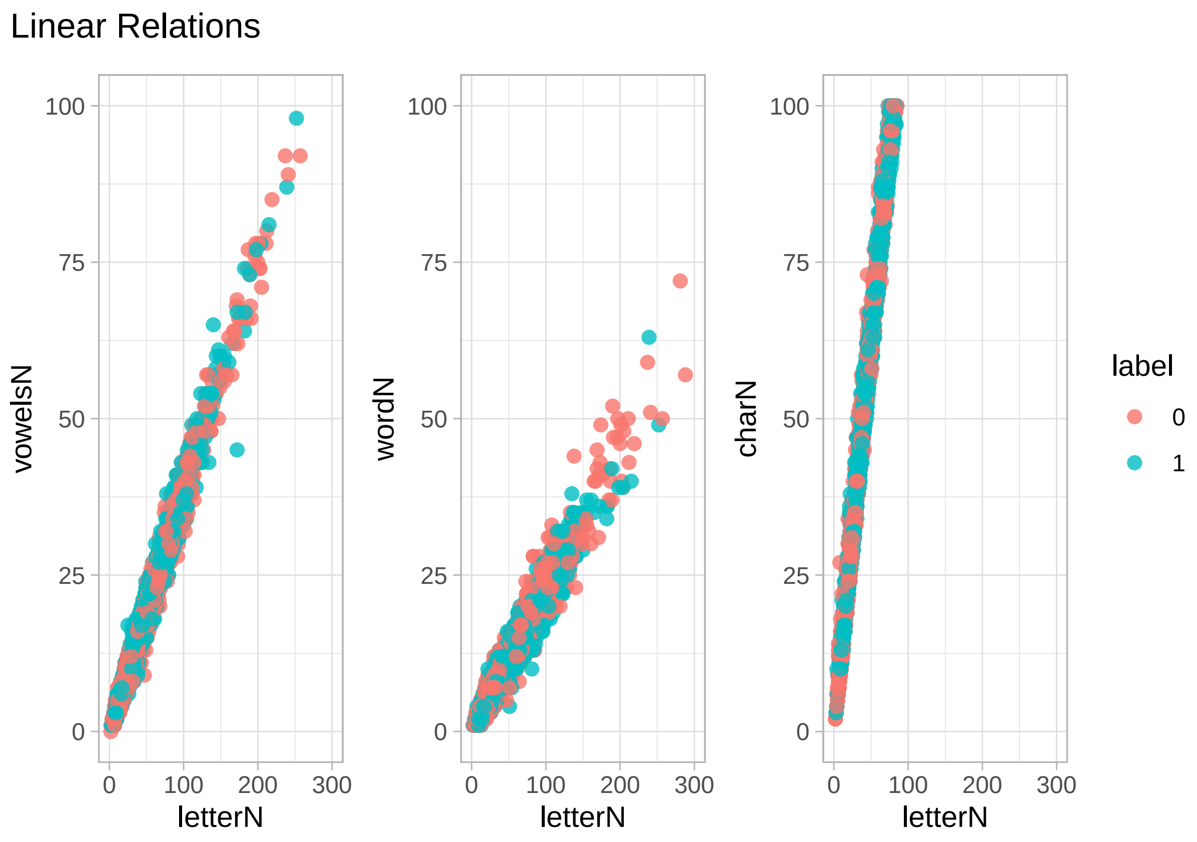
Subreddit Graph:

```
subreddit_graph <- sarcastic %>%
  ggplot(., aes(subreddit, label, fill = label)) +
  geom_boxplot() +
  geom_jitter(aes(color = label), alpha = .3) +
  theme(axis.text.x = element_text(size = 7))

subreddit_graph
```



By dividing subreddits into groups of sarcastic and non-sarcastic I can tell where users post their sarcastic posts/ non-sarcastic posts. Which can show me which of the subreddit is better for my predictions. # Linear Relation Graphs:

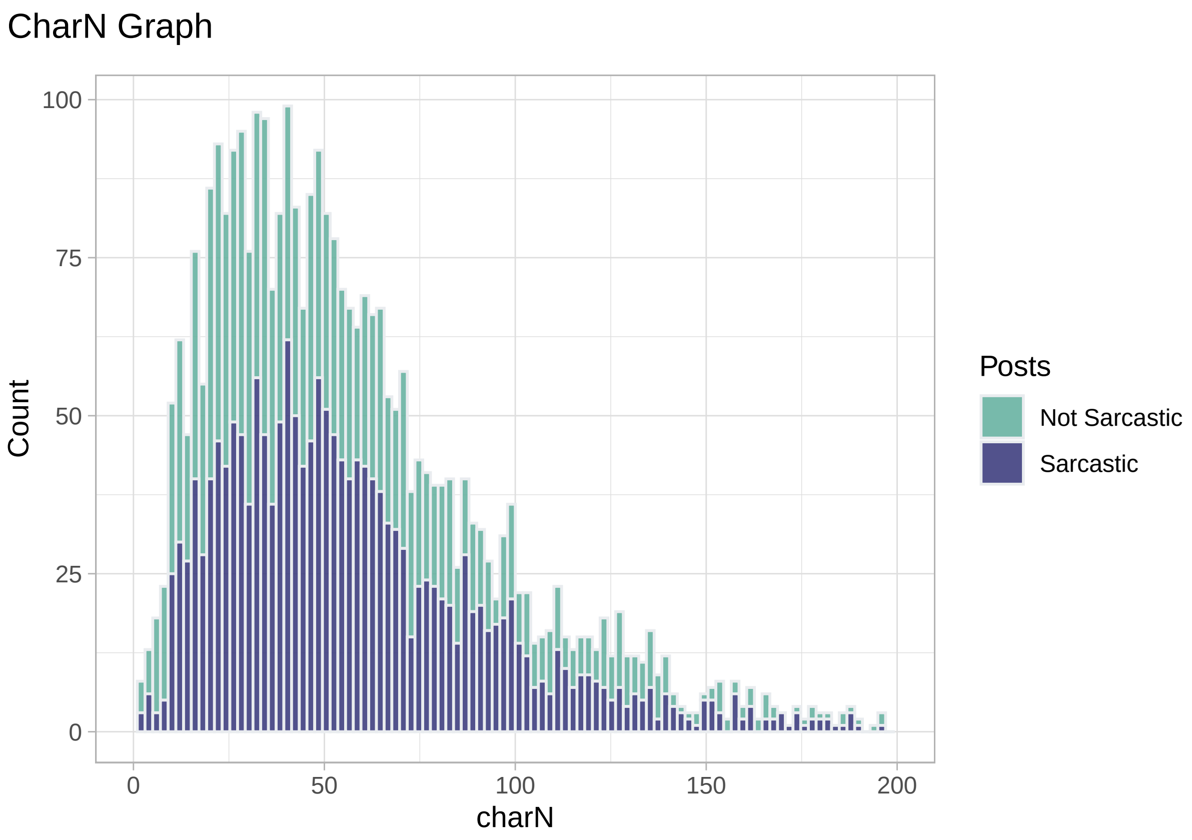
```
linear_relation_letterN_vowelsN <- sarcastic %>%
  ggplot(.,aes(letterN,vowelsN, color = label)) +
  geom_point(size = 2, alpha = .8) +
  theme_light()+
  ylim(0,100)+
  xlim(0,300)

linear_relation_letterN_wordN <- sarcastic %>%
  ggplot(., aes(letterN, wordN, color = label)) +
  geom_point(size  = 2,alpha = .8) +
  theme_light() +
  ylim(0,100) +
  xlim(0,300)

linear_relation_LetterN_charN <- sarcastic %>%
  ggplot(., aes(letterN, charN, color=label))+
  geom_point(size=2, alpha = .8) +
  theme_light() +
  ylim(0,100)+
  xlim(0,300)


linear_relation_letterN_vowelsN +
  linear_relation_letterN_wordN +
  linear_relation_LetterN_charN +
  plot_annotation(title = "Linear Relations") +
  plot_layout(guides = "collect")
```



After inspecting the data I figured there might be overlapping columns. Therefore I made these graphs to check the linear relations between letterN and other predictors that has relation to letters. Since the growth of the predictors depends on each other I can tell that there's some connection between these predictors that I might need to handle before training my model.

CharN Graph:

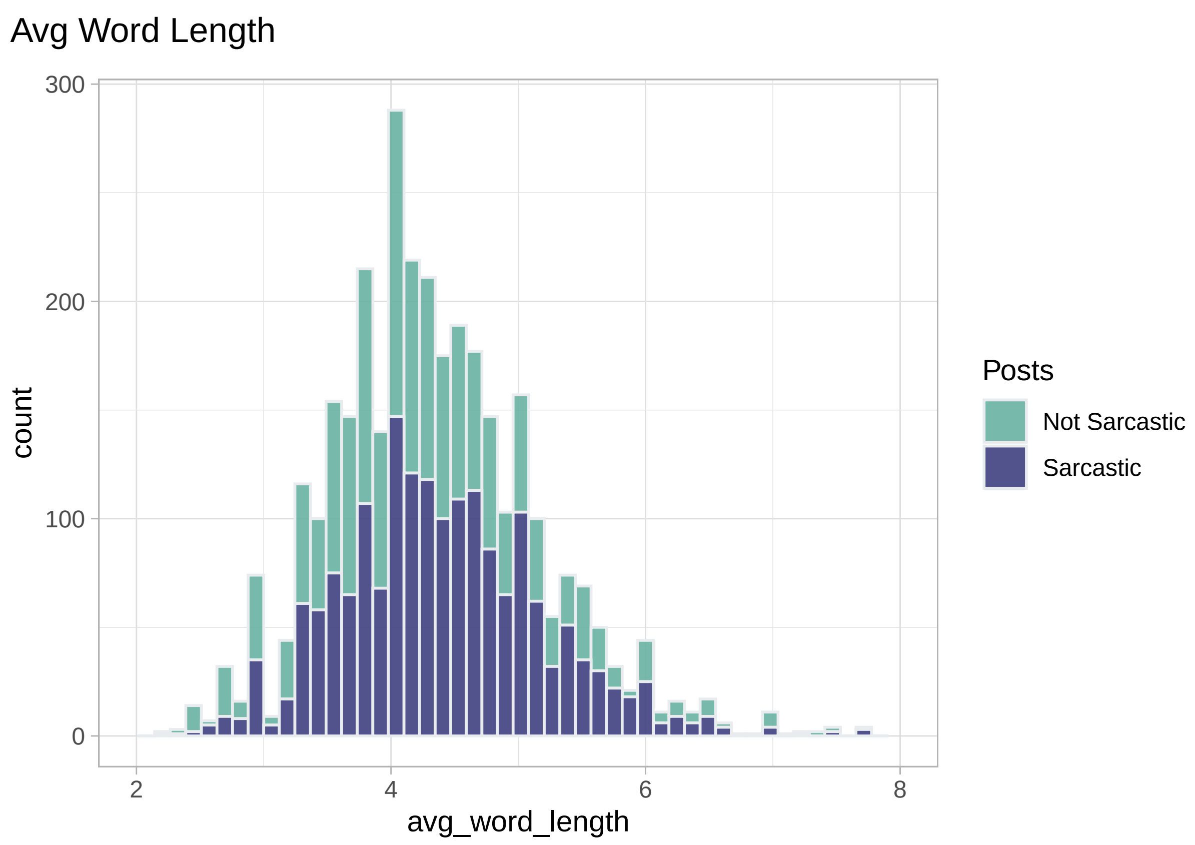
```
charN_histogram <- sarcastic %>%
  ggplot( aes(x=charN, fill=label)) +
    geom_histogram( color="#e9ecef", alpha=0.9,bins=100) +
    scale_fill_manual(labels = c("Not Sarcastic","Sarcastic"),
                      values=c("#69b3a2", "#404080")) +
  theme_light()+
    plot_annotation(title= "CharN Graph")+
    labs(y = "Count")+xlim(0,200) + labs(fill="Posts")
charN_histogram
```



It seems that there's a different between sarcastic post and non-sarcastic post when we're looking at the number of characters in the post. As the graph shows, in non sarcastic posts there are more characters than in sarcastic post. Therefore CharN might help me to determine whether a user's post is sarcastic.

Avg Word Length Histogram:

```
avg_word_length_histogram <- sarcastic %>%
  ggplot( aes(x=avg_word_length, fill=label)) +
    geom_histogram( color="#e9ecef", alpha=0.9,bins=50) +
    scale_fill_manual(labels = c("Not Sarcastic","Sarcastic"),
                      values=c("#69b3a2", "#404080")) +
  theme_light()+
    xlim(2,8)+
    plot_annotation(title="Avg Word Length")+
    labs(fill="Posts")
avg_word_length_histogram
```



When users post a non sarcastic post they use longer words.Even though I don't have the text of the posts I can still tell that there might be more characters in non sarcastic posts because of the high usage of letters. This rises another question: are there more letters in non sarcastic posts?

Number of Letters Comparsion:

```
non_sarcastic_letters <- mean(sarcastic["letterN"][sarcastic["label"] == 0])
sarcastic_letters <- mean(sarcastic["letterN"][sarcastic["label"] == 1])

paste("non sarcastic posts - mean of number of letters:",non_sarcastic_letters)
```

```
## [1] "non sarcastic posts - mean of number of letters: 47.47"
```

```
paste("sarcastic posts - mean of number of letters:",sarcastic_letters)
```

```
## [1] "sarcastic posts - mean of number of letters: 47.3814773980154"
```

Number of Characters Comparsion:

```
non_sarcastic_chars <- mean(sarcastic["charN"][sarcastic["label"] == 0])
sarcastic_chars <- mean(sarcastic["charN"][sarcastic["label"] == 1])

paste("non sarcastic posts - mean  of number of  characters:",non_sarcastic_chars)
```

```
## [1] "non sarcastic posts - mean  of number of  characters: 60.1966666666667"
```

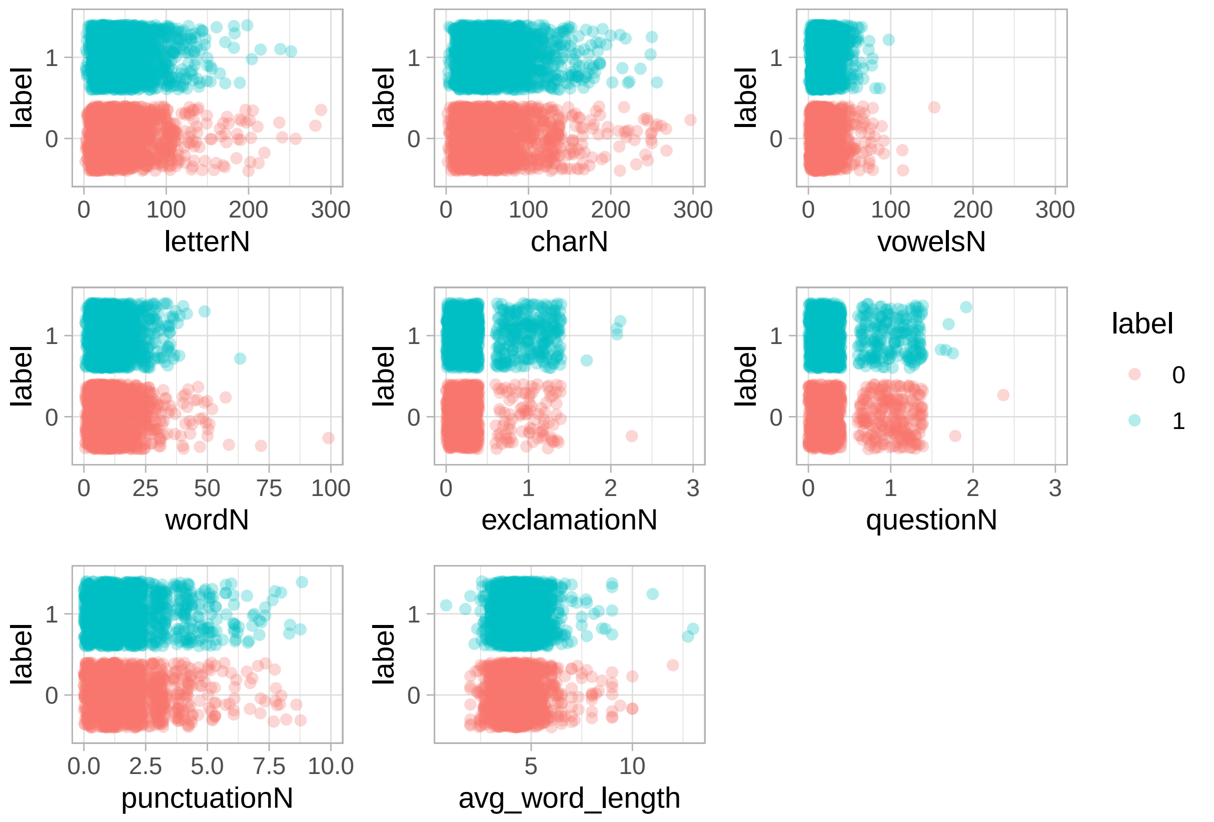
```r
paste("sarcastic posts - mean of number of characters:",sarcastic_chars)
```

```
## [1] "sarcastic posts - mean of number of characters: 60.0242557883131"
```

Dispersion of Charaters Related Features

```r
letters_vs_label <- sarcastic %>%
  ggplot(.,aes(letterN,label,color=label)) +
  geom_jitter(alpha=0.3)  +
  theme_light()+
  xlim(0,300)


chars_vs_label <- sarcastic %>%
   ggplot(.,aes(charN,label,color=label)) +
  geom_jitter(alpha=0.3)  +
  theme_light() +
  xlim(0,300)

vowels_vs_label <- sarcastic %>%
   ggplot(.,aes(vowelsN,label,color=label)) +
  geom_jitter(alpha=0.3) +
 theme_light() +
  xlim(0,300)

words_vs_label <- sarcastic %>%
   ggplot(.,aes(wordN,label,color=label)) +
  geom_jitter(alpha=0.3) +
  theme_light() +
  xlim(0,100)

exclamation_vs_label <- sarcastic %>%
   ggplot(.,aes(exclamationN,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,3)+
  theme_light()


question_vs_label <- sarcastic %>%
   ggplot(.,aes(questionN,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,3)+
  theme_light()

punctuation_vs_label <- sarcastic %>%
   ggplot(.,aes(punctuationN,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,10)+
  theme_light()
```

```
avg_word_length_vs_label <- sarcastic %>%
  ggplot(.,aes(avg_word_length,label,color=label)) +
  geom_jitter(alpha=0.3) +
  theme_light()



letters_vs_label +
  chars_vs_label+
  vowels_vs_label+
  words_vs_label+
  exclamation_vs_label+
  question_vs_label+
  punctuation_vs_label +
  avg_word_length_vs_label+
  plot_layout(guides = "collect")
```

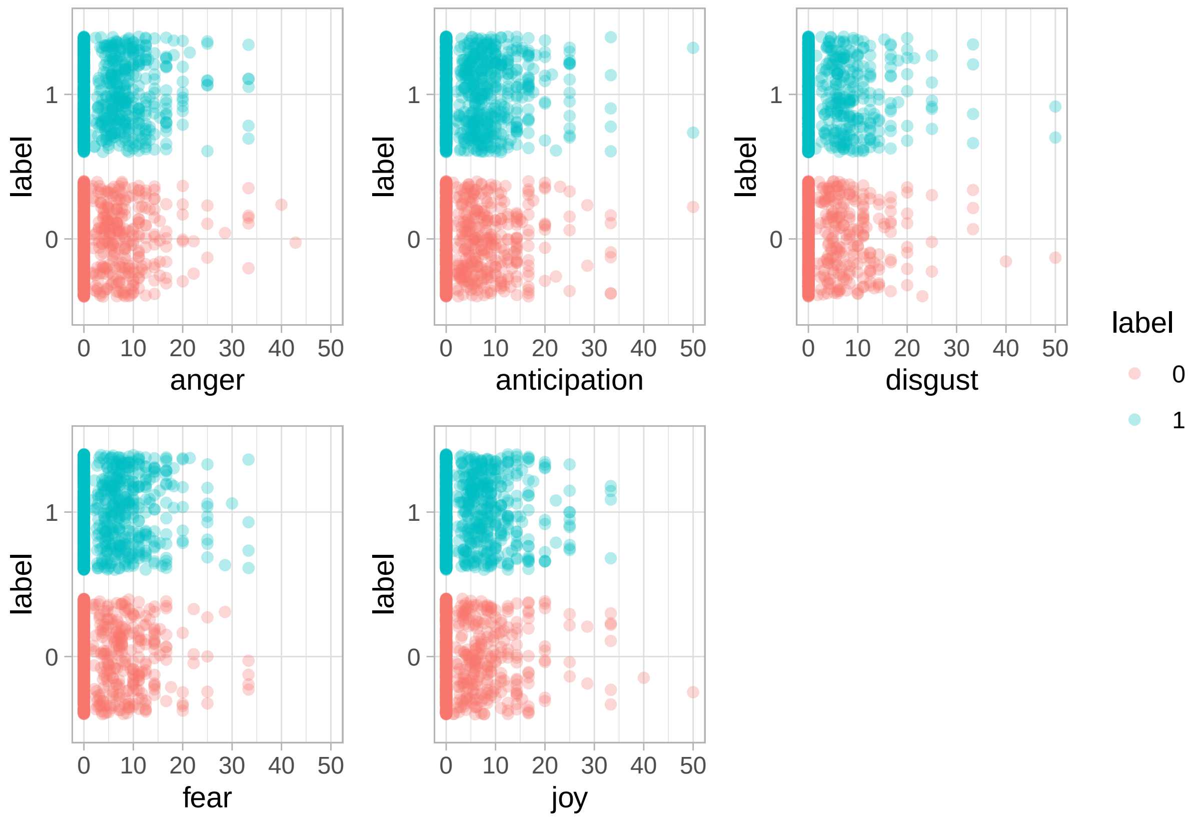


The dispersion graphs shows the there might be some outliers in the data. Therefore a data clean-up might be needed in order to train our model properly.

In some posts there's a huge difference between non sarcastic ones and sarcastic ones. However many of the posts (sarcastic or non sarcastic) are almost similar in their number of characters, number of words, number of letters etc.

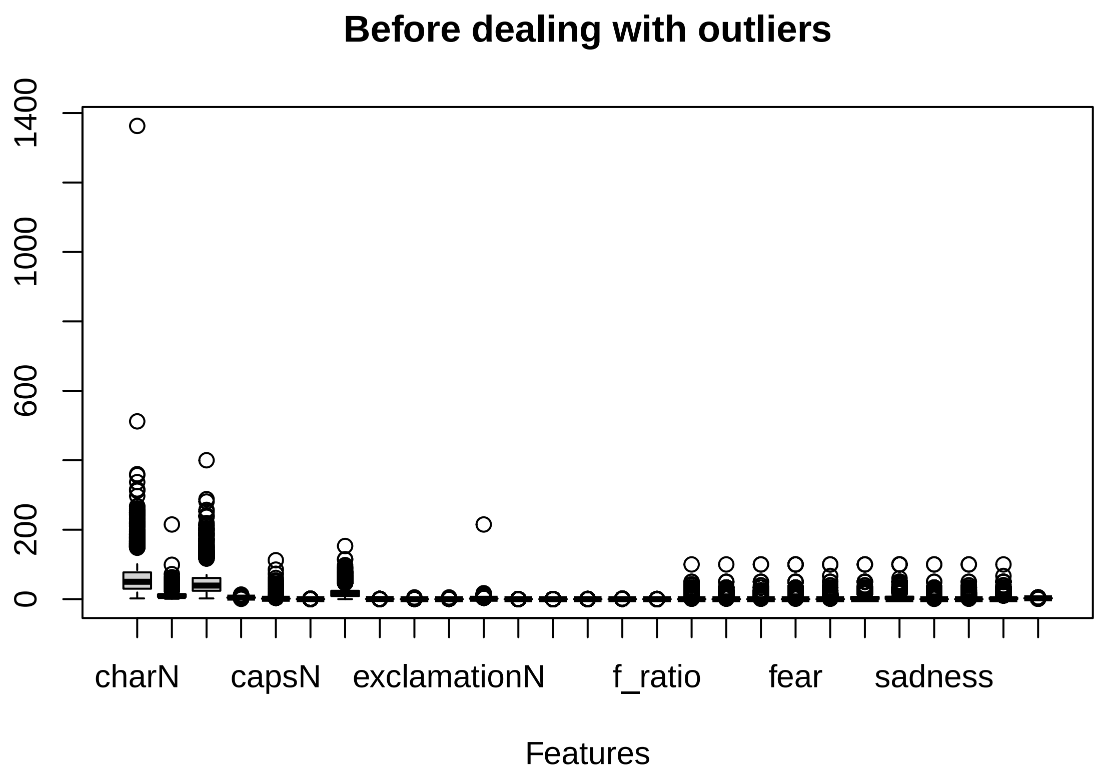
Dispersion of Sentiment Related Features

```r
anger_vs_label <- sarcastic %>%
  ggplot(.,aes(anger,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,50)+
  theme_light()

anticipation_vs_label <- sarcastic %>%
  ggplot(.,aes(anticipation,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,50)+
  theme_light()

disgust_vs_label <- sarcastic %>%
  ggplot(.,aes(disgust,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,50)+
  theme_light()

fear_vs_label <- sarcastic %>%
  ggplot(.,aes(fear,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,50)+
  theme_light()

joy_vs_label <- sarcastic %>%
  ggplot(.,aes(joy,label,color=label)) +
  geom_jitter(alpha=0.3) +
  xlim(0,50)+
  theme_light()


anger_vs_label+
  anticipation_vs_label+
  disgust_vs_label+
  fear_vs_label+
  joy_vs_label+
  plot_layout(guides="collect")
```

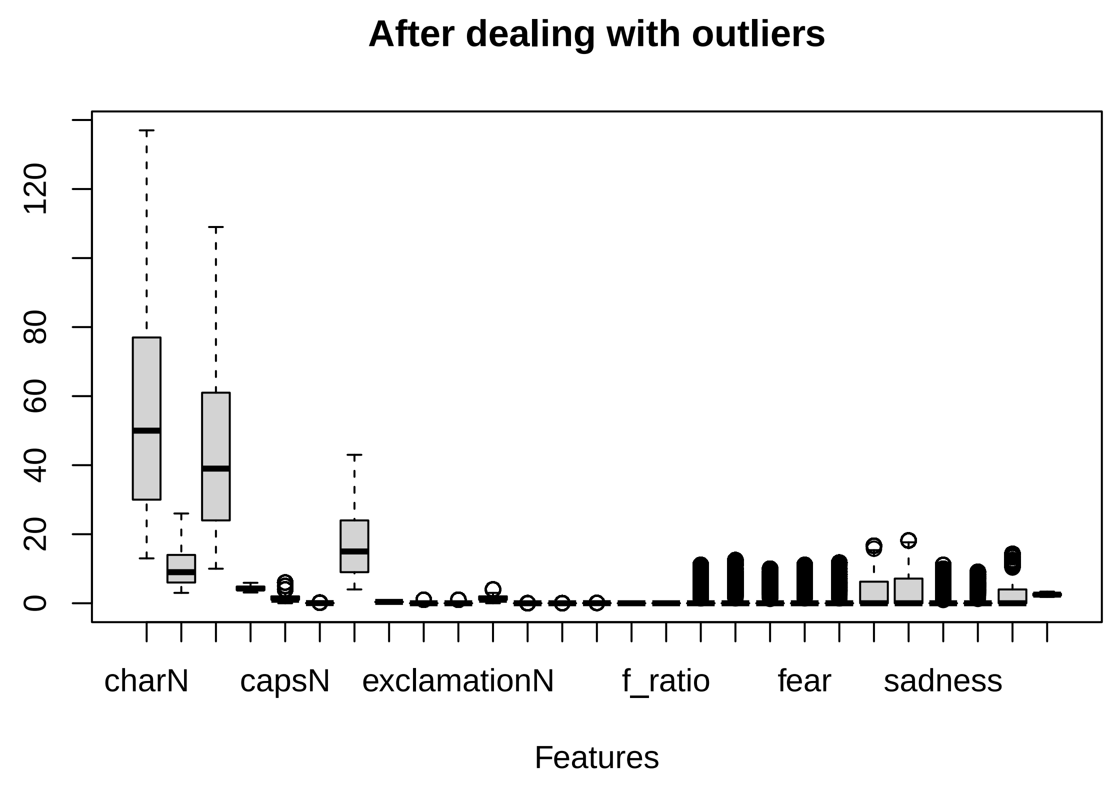


Data Manipulations

```
boxplot(sarcastic[3:ncol(sarcastic)],main="Before dealing with outliers", xlab="Features")
```



```r
for (i in 3:ncol(sarcastic)){

  quant_high <- quantile(sarcastic[,i], prob=0.95) # high quantile value
  quant_low <- quantile(sarcastic[,i], prob=0.05) # low quantile value

  sarcastic[,i] <- ifelse(sarcastic[,i] > quant_high, quant_high, sarcastic[,i])
  sarcastic[,i] <- ifelse(sarcastic[,i] < quant_low, quant_low, sarcastic[,i])
}

boxplot(sarcastic[3:ncol(sarcastic)],main="After dealing with outliers", xlab="Features")
```



Feature Selection using Boruta

```
bor_test <- Boruta(label~., sarcastic)

bor_test
```

```
## Boruta performed 99 iterations in 1.146078 mins.
##  14 attributes confirmed important: anticipation, avg_word_length,
## charN, concrete, exclamationN and 9 more;
##  13 attributes confirmed unimportant: anger, capsN, disgust, f_ratio,
## fBombN and 8 more;
##  1 tentative attributes left: caps_ratio;
```

```
bor_test_fixed <- TentativeRoughFix(bor_test) # few more features are left to check.
getConfirmedFormula(bor_test_fixed) # gives the predicting formula.
```

```
## label ~ subreddit + charN + wordN + letterN + avg_word_length +
##     caps_ratio + vowelsN + questionN + exclamationN + punctuationN +
##     questions_ratio + excs_ratio + puncs_ratio + anticipation +
##     concrete
## <environment: 0x11f715098>
```

Test - Train Split & Cross Validation Method

```
set.seed(101)
sample <- sample.int(n = nrow(sarcastic), size = floor(.75*nrow(sarcastic)), replace = F)
train <- sarcastic[sample, ]
test  <- sarcastic[-sample, ]

cv <- trainControl(method="cv",number=10)
```

Supervised Machine Learning Models

Support Vector Machine

```
set.seed(101)
svm <- train(label ~ subreddit + charN + wordN + letterN + avg_word_length +
    vowelsN + questionN + exclamationN + punctuationN + questions_ratio +
    excs_ratio + puncs_ratio + concrete,
    data = train,
    method="svmLinear2",
    trControl = cv)
svm
```

```
## Support Vector Machines with Linear Kernel
##
## 2485 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2235, 2236, 2237, 2237, 2237, 2237, ...
## Resampling results across tuning parameters:
##
##   cost  Accuracy   Kappa
##   0.25  0.5681939  0.09100224
##   0.50  0.5661810  0.08882498
##   1.00  0.5669810  0.09033089
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cost = 0.25.
```

Random Forest

```
set.seed(101)
rf <- train(label ~ subreddit + charN + wordN + letterN + avg_word_length +
    vowelsN + questionN + exclamationN + punctuationN + questions_ratio +
    excs_ratio + puncs_ratio + concrete,
```

```r
    data = train,
    method = "rf",
    ntree = 500,
      importance = T,
    tuneGrid = expand.grid(mtry = 2),
    trControl = cv)

rf
```

```
## Random Forest
##
## 2485 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2235, 2236, 2237, 2237, 2237, 2237, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.5943487  0.1513206
##
## Tuning parameter 'mtry' was held constant at a value of 2
```

KNN

```r
set.seed(101)
knn <- train(label ~ subreddit + charN + wordN + letterN + avg_word_length +
    vowelsN + questionN + exclamationN + punctuationN + questions_ratio +
    excs_ratio + puncs_ratio + concrete,
    data = train,
    method = "knn",
    tuneGrid = expand.grid(k = c(1:10)),
    trControl = cv)
knn
```

```
## k-Nearest Neighbors
##
## 2485 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2235, 2236, 2237, 2237, 2237, 2237, ...
## Resampling results across tuning parameters:
##
##   k   Accuracy   Kappa
##    1  0.5307554  0.05442467
##    2  0.5303635  0.05380449
##    3  0.5468619  0.07969860
##    4  0.5412119  0.06587402
##    5  0.5480440  0.07862981
##    6  0.5504650  0.08552911
##    7  0.5572940  0.09654912
##    8  0.5633327  0.10903572
##    9  0.5537086  0.08501548
##   10  0.5500682  0.07685311
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 8.
```

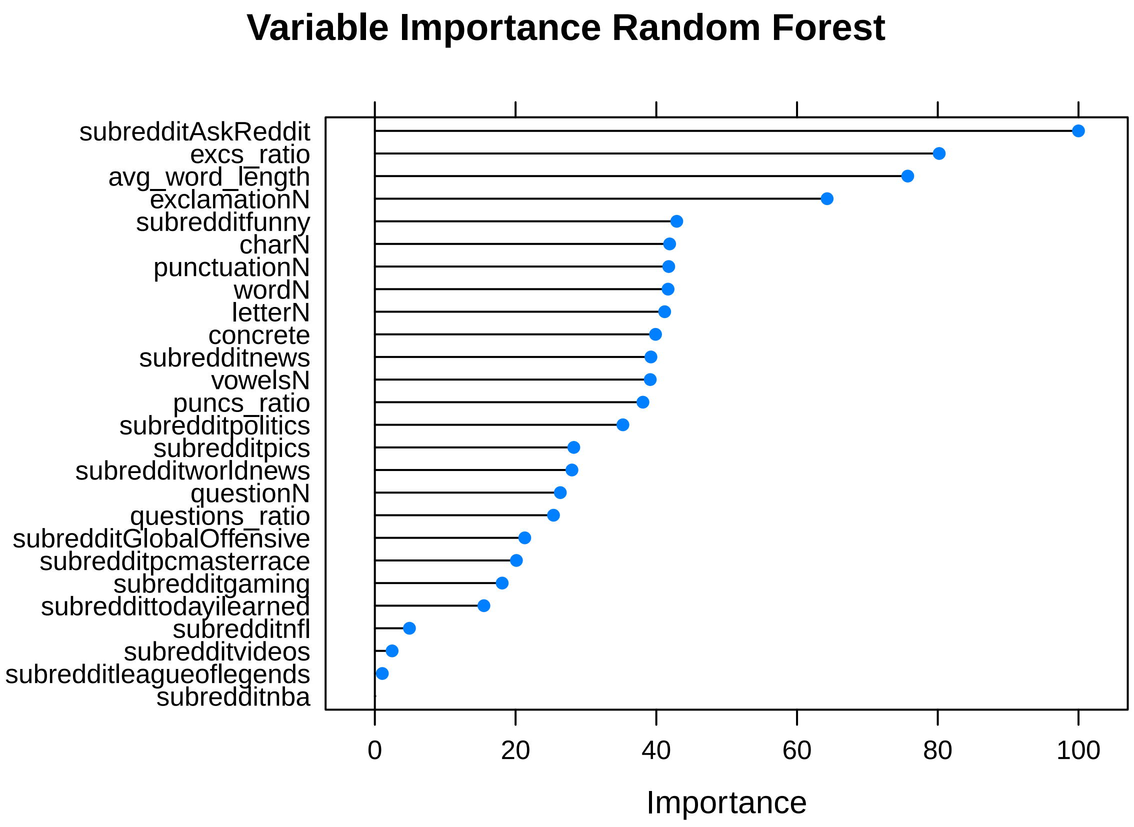
Neural Network

```
set.seed(101)
NNK <- train(label ~ subreddit + charN + wordN + letterN + avg_word_length +
    vowelsN + questionN + exclamationN + punctuationN + questions_ratio +
    excs_ratio + puncs_ratio + concrete,
    data = train,
    method = "nnet",
    trace =F,
    tuneGrid = expand.grid(size=c(1,3,5,7,9),decay=0.01),
      importance = T,
    trControl = cv)
NNK
```

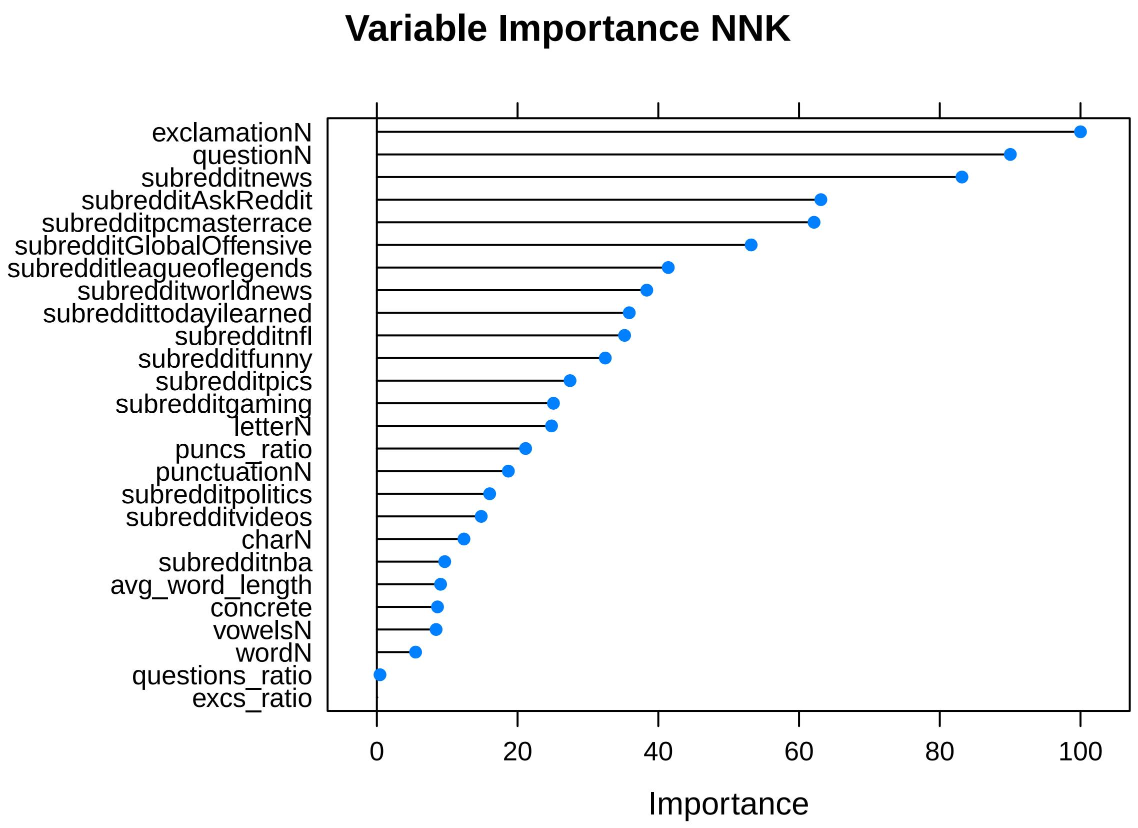
```
## Neural Network
##
## 2485 samples
##   13 predictor
##    2 classes: '0', '1'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 2235, 2236, 2237, 2237, 2237, 2237, ...
## Resampling results across tuning parameters:
##
##   size  Accuracy   Kappa
##   1     0.5706246  0.1352087
##   3     0.5798729  0.1492928
##   5     0.5625697  0.1146722
##   7     0.5693810  0.1254108
##   9     0.5786164  0.1423640
##
## Tuning parameter 'decay' was held constant at a value of 0.01
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were size = 3 and decay = 0.01.
```

Variable Importance

```r
plot(varImp(rf), main="Variable Importance Random Forest")
```



```r
plot(varImp(NNK),main="Variable Importance NNK")
```



As the graphs shows, there's difference between random forest's rating of features and neural network's rating of features.

Model Comparsion

```
svm_pred <- predict(svm,test)
rf_pred <- predict(rf,test)
knn_pred <- predict(knn,test)
nnk_pred <- predict(NNK,test)


test_results <- tibble(
  name=c('SVM','Random Forest','K-Nearest Neighbors','Neural Network'),
  R2=c(R2(as.numeric(svm_pred),as.numeric(test$label)),
       R2(as.numeric(rf_pred),as.numeric(test$label)),
       R2(as.numeric(knn_pred),as.numeric(test$label)),
       R2(as.numeric(nnk_pred),as.numeric(test$label))),
  RMSE=c(RMSE(as.numeric(svm_pred),as.numeric(test$label)),
         RMSE(as.numeric(rf_pred),as.numeric(test$label)),
         RMSE(as.numeric(knn_pred),as.numeric(test$label)),
         RMSE(as.numeric(nnk_pred),as.numeric(test$label))))

knitr::kable(test_results, align = 'cl', caption = "Comparing Results")
```

Table 1: Comparing Results

name	R2	RMSE
SVM	0.0221571	0.6479065
Random Forest	0.0497887	0.6203233
K-Nearest Neighbors	0.0084805	0.6716740
Neural Network	0.0321618	0.6460420

By calculating R squared and Root Mean Squared, I can tell which model did better at detecting sarcasm. From the calculations above, we can clearly see that Random Forest's RMSE is the lowest therefore random forest is probably the better choice for this kinda of data. # Confusion Matrix

```
svm_conf_mat <- confusionMatrix(svm_pred,test$label)
rf_conf_mat <- confusionMatrix(rf_pred,test$label)
knn_conf_mat <- confusionMatrix(knn_pred,test$label)
nnk_conf_mat <- confusionMatrix(nnk_pred,test$label)
```

SVM Confusion Matrix

```
svm_conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 122  82
##          1 266 359
##
##                Accuracy : 0.5802
##                  95% CI : (0.5458, 0.6141)
##     No Information Rate : 0.532
##     P-Value [Acc > NIR] : 0.002924
##
##                   Kappa : 0.1323
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.3144
##             Specificity : 0.8141
##          Pos Pred Value : 0.5980
##          Neg Pred Value : 0.5744
##              Prevalence : 0.4680
##          Detection Rate : 0.1472
##    Detection Prevalence : 0.2461
##       Balanced Accuracy : 0.5642
##
##        'Positive' Class : 0
##
```

Random Forest Confusion Matrix

```
rf_conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 164  95
##          1 224 346
##
##                Accuracy : 0.6152
##                  95% CI : (0.5811, 0.6485)
##     No Information Rate : 0.532
##     P-Value [Acc > NIR] : 8.134e-07
##
##                   Kappa : 0.2115
##
##  Mcnemar's Test P-Value : 7.687e-13
##
##             Sensitivity : 0.4227
##             Specificity : 0.7846
##          Pos Pred Value : 0.6332
##          Neg Pred Value : 0.6070
##              Prevalence : 0.4680
##          Detection Rate : 0.1978
##    Detection Prevalence : 0.3124
##       Balanced Accuracy : 0.6036
##
##        'Positive' Class : 0
##
```

K-Nearest Neighbor Confusion Matrix

```
knn_conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 194 180
##          1 194 261
##
##                Accuracy : 0.5489
##                  95% CI : (0.5143, 0.5831)
##     No Information Rate : 0.532
##     P-Value [Acc > NIR] : 0.1737
##
##                   Kappa : 0.092
##
```

```
##  Mcnemar's Test P-Value : 0.5014
##
##             Sensitivity : 0.5000
##             Specificity : 0.5918
##          Pos Pred Value : 0.5187
##          Neg Pred Value : 0.5736
##              Prevalence : 0.4680
##          Detection Rate : 0.2340
##    Detection Prevalence : 0.4511
##       Balanced Accuracy : 0.5459
##
##        'Positive' Class : 0
##
```

Neural Network Confusion Matrix

```
nnk_conf_mat
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 264 222
##          1 124 219
##
##                Accuracy : 0.5826
##                  95% CI : (0.5482, 0.6165)
##     No Information Rate : 0.532
##     P-Value [Acc > NIR] : 0.001888
##
##                   Kappa : 0.1744
##
##  Mcnemar's Test P-Value : 1.841e-07
##
##             Sensitivity : 0.6804
##             Specificity : 0.4966
##          Pos Pred Value : 0.5432
##          Neg Pred Value : 0.6385
##              Prevalence : 0.4680
##          Detection Rate : 0.3185
##    Detection Prevalence : 0.5862
##       Balanced Accuracy : 0.5885
##
##        'Positive' Class : 0
##
```

After checking how every model did on the sarcasm detection task I came up with few conclusions: first,it seems that Random Forest is probably the best choice from the models I have tried, to predict sarcasm in Reddit posts. Even though there's a linear relation between few features the SVM linear model wasn't able to get over 60% accuracy. Neural Network is an astonishing model, however it still falls bellow Random Forest at this specific task.