Project Report

STAT 380 Section 004

Gianna DeLorenzo & Melissa Kim

1 Title:

1.1 Introduction

Patient experiences with medications is crucial when it comes to the healthcare domain by providing insights on the side effects, drug effectiveness, and overall satisfaction.

Research Question

In this report, we will be exploring the following research question:

Can machine learning tasks predict the overall satisfaction of the patients with a particular class of drugs for a given medical condition?

This research question

1.2 Exploratory Data Analysis

1.2.1 Variable Description

The response variable of interest is the ratings of the drugs with the negative reviews from **0-6** and the positive reviews from **7-10**.

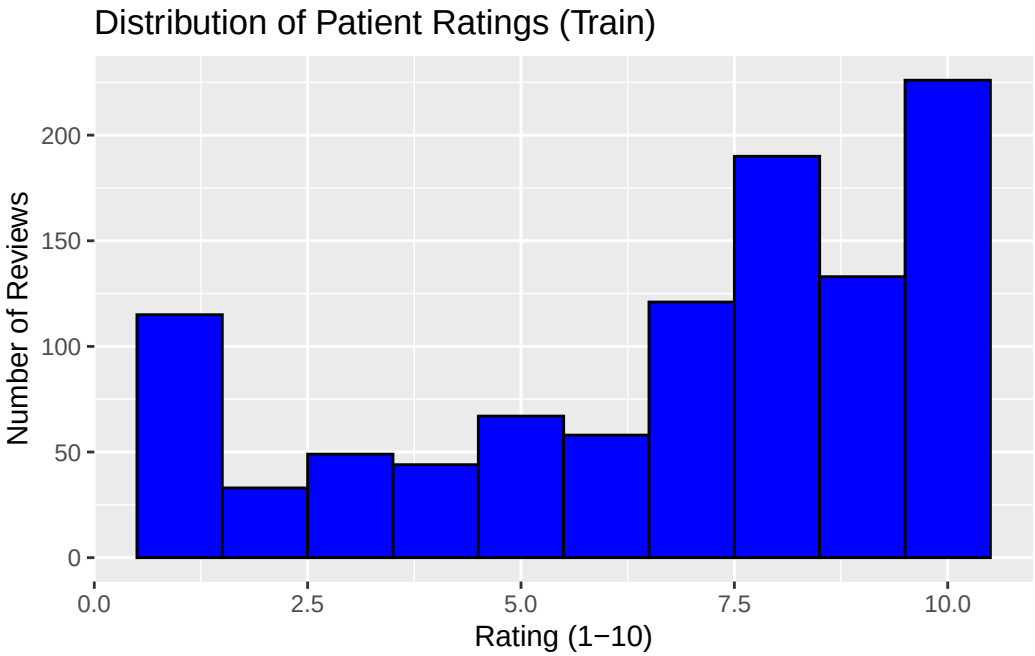
1.2.2 Data Visualization

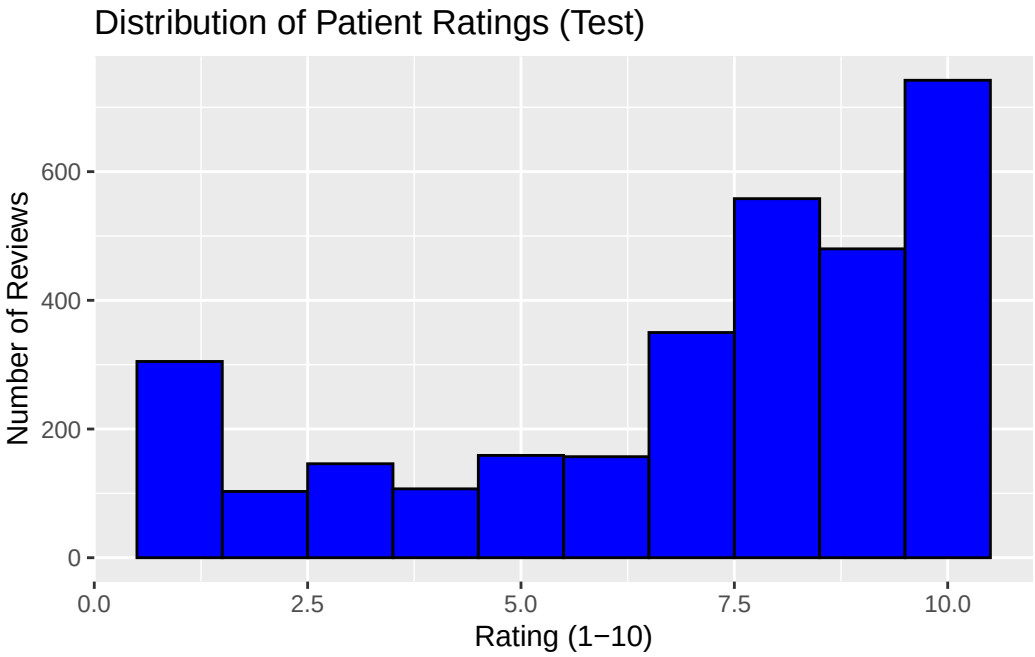
Table 1: Variables used in Analysis

Variable	Type	Explanation
reviewID	Integer	Review ID
urlDrugName	Categorial	Name of drug
rating	Integer	Name of condition
effectiveness	Categorical	Patient on benefits
sideEffects	Categorical	Patient on side effects
condition	Categorical	Overall patient comment
benefitsReview	Categorical	10 star patient rating

Variable	Type	Explanation
sideEffectsReview	Categorical	5 step side effect rating
commentsReview	Categorical	5 step effectiveness rating

Table 1





1.2.3 Data Cleaning

This glimpse of the Drug Reviews train and test data displays a dataset in need of being cleaned and tidied.

```
Rows: 1,036
Columns: 9
$ ...1              <dbl> 1366, 3724, 3824, 969, 696, 1380, 45, 1939, 2576, 10~
$ urlDrugName       <chr> "biaxin", "lamictal", "depakene", "sarafem", "accuta~
$ rating            <dbl> 9, 9, 4, 10, 10, 2, 8, 10, 10, 1, 3, 9, 6, 10, 5, 5,~
$ effectiveness     <chr> "Considerably Effective", "Highly Effective", "Moder~
$ sideEffects       <chr> "Mild Side Effects", "Mild Side Effects", "Severe Si~
$ condition         <chr> "sinus infection", "bipolar disorder", "bipolar diso~
$ benefitsReview    <chr> "The antibiotic may have destroyed bacteria causing ~
$ sideEffectsReview <chr> "Some back pain, some nauseau.", "Drowsiness, a bit ~
$ commentsReview    <chr> "Took the antibiotics for 14 days. Sinus infection w~

Rows: 3,107
Columns: 9
$ ...1              <dbl> 2202, 3117, 1146, 3947, 1951, 2372, 1043, 2715, 1591~
$ urlDrugName       <chr> "enalapril", "ortho-tri-cyclen", "ponstel", "prilose~
$ rating            <dbl> 4, 1, 10, 3, 2, 1, 9, 10, 10, 1, 7, 8, 8, 9, 4, 8, 6~
$ effectiveness     <chr> "Highly Effective", "Highly Effective", "Highly Effe~
$ sideEffects       <chr> "Mild Side Effects", "Severe Side Effects", "No Side~
$ condition         <chr> "management of congestive heart failure", "birth pre~
$ benefitsReview    <chr> "slowed the progression of left ventricular dysfunct~
$ sideEffectsReview <chr> "cough, hypotension , proteinuria, impotence , renal~
$ commentsReview    <chr> "monitor blood pressure , weight and asses for resol~
```

We tidied the Drug Reviews Train and Test datasets by cleaning up the names with the Janitor library.

1.3 Modeling

1.3.1 Logistic Regression: Predicting Positive vs. Negative Reviews

Can we predict if a patient's review is positive or negative based on their feedback?

```
Confusion Matrix and Statistics

          Reference
Prediction    0    1
         0  548  121
         1  305 1716

               Accuracy : 0.8416
```

```
               95% CI : (0.8273, 0.8552)
    No Information Rate : 0.6829
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6119

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6424
            Specificity : 0.9341
         Pos Pred Value : 0.8191
         Neg Pred Value : 0.8491
             Prevalence : 0.3171
         Detection Rate : 0.2037
   Detection Prevalence : 0.2487
      Balanced Accuracy : 0.7883

       'Positive' Class : 0
```

Confusion Matrix 0 = negative review, 1 = positive review

True Negatives (TN) = 212 212 reviews were predicted negative and actually negative

False Positives (FP) = 107 107 reviews were predicted positive but actually negative

False Negatives (FN) = 48 48 reviews were predicted negative but actually positive

True Positives (TP) = 537 537 reviews were predicted positive and actually positive

Metric	Value	Meaning
Accuracy	0.8285	82.85% of predictions were correct overall
Sensitivity (Class 0)	0.6646	66.46% of actual negative reviews were correctly identified
Specificity (Class 1)	0.9179	91.79% of actual positive reviews were correctly identified
Kappa	0.6081	Moderate to strong agreement beyond chance, follows within the “good” range of 0.6-0.8
Precision (class 0)	0.8154	Out of all predicted negative reviews, 81.54% were correct
Negative Predictive Value	0.8339	Out of all predicted positive reviews, 83.39% were correct
Balanced Accuracy	0.7913	Average of sensitivity and specificity; useful for imbalanced classes
No Information Rate (NIR)	0.6471	64.71% of reviews in the test set are in the majority class (positive); always guessing the majority class (positive) would be correct 64.71% of the time

Metric	Value	Meaning
McNemar's Test	p = 3.183e-06	Suggests a significant difference in types of errors; model is not just guessing randomly

Model Strengths: - High accuracy (82.9%) - Excellent at identifying positive reviews (specificity = 91.8%) - High precision for negative class (precision = 81.5%), so false alarms are limited - Performs much better than guessing the majority class (accuracy > NIR)

Model Weaknesses: - Struggles more with detecting negative reviews (sensitivity = 66.5%), misses about 1 in 3 negative reviews

Conclusion The logistic regression model achieved an overall accuracy of 82.85%, which significantly outperforms the no-information rate of 64.71% (p < 2.2e-16). It performs well in identifying positive reviews (specificity = 91.8%) but is less sensitive to negative reviews (sensitivity = 66.5%). The balanced accuracy of 79.1% confirms that the model maintains reasonable performance across both classes. The confusion matrix and McNemar's test (p = 3.18e-06) further indicate that the model is not biased toward one class and captures meaningful patterns in patient review sentiment.

Answer to "Can we predict if a patient's review is positive or negative based on their feedback?" Yes, we can predict whether a patient's review is positive or negative using machine learning. Our logistic regression model, trained on features such as the drug name, medical condition, and reported effectiveness, achieved an overall accuracy of 82.85%, which was much higher than the no-information rate of 64.71%. The model showed strong performance in identifying positive reviews (specificity = 91.8%) and reasonably good performance for negative reviews (sensitivity = 66.5%). These results indicate that the model can reliably classify overall patient sentiment based on the structured data.

1.3.2 Neutral Network: Predicting Ratings with Nuance

Can we predict a more nuanced outcome of the review based on patient attributes, dosage, and drug features?

```
Confusion Matrix and Statistics

          Reference
Prediction  1  2  3  4  5  6  7  8  9 10
        1  12  3  4  2  1  2  1  0  0  0
        2   0  0  0  0  0  0  0  0  0  0
        3   0  0  0  0  0  0  0  0  0  0
        4   1  0  2  1  0  0  0  0  0  0
        5   0  0  0  0  0  0  0  0  2  0
        6   1  1  0  0  0  1  5  1  1  0
        7   3  2  1  1  5  4  5 10  4  1
        8   3  0  1  1  3  3  7 18  8  3
        9   0  0  0  0  0  0  0  1  2  0
```

```
        10  3  0  1  3  4  1  6  8  9 41

Overall Statistics

               Accuracy : 0.3941
                 95% CI : (0.3264, 0.4649)
    No Information Rate : 0.2217
    P-Value [Acc > NIR] : 2.428e-08

                  Kappa : 0.2724

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
Sensitivity           0.52174  0.00000  0.00000 0.125000 0.000000 0.090909
Specificity           0.92778  1.00000  1.00000 0.984615 0.989474 0.953125
Pos Pred Value        0.48000      NaN      NaN 0.250000 0.000000 0.100000
Neg Pred Value        0.93820  0.97044  0.95567 0.964824 0.935323 0.948187
Prevalence            0.11330  0.02956  0.04433 0.039409 0.064039 0.054187
Detection Rate        0.05911  0.00000  0.00000 0.004926 0.000000 0.004926
Detection Prevalence  0.12315  0.00000  0.00000 0.019704 0.009852 0.049261
Balanced Accuracy     0.72476  0.50000  0.50000 0.554808 0.494737 0.522017
                     Class: 7 Class: 8 Class: 9 Class: 10
Sensitivity           0.20833  0.47368 0.076923    0.9111
Specificity           0.82682  0.82424 0.994350    0.7785
Pos Pred Value        0.13889  0.38298 0.666667    0.5395
Neg Pred Value        0.88623  0.87179 0.880000    0.9685
Prevalence            0.11823  0.18719 0.128079    0.2217
Detection Rate        0.02463  0.08867 0.009852    0.2020
Detection Prevalence  0.17734  0.23153 0.014778    0.3744
Balanced Accuracy     0.51757  0.64896 0.535637    0.8448
```

1.3.2.1 Interpretation

1.3.2.1.1 Class-by-Class Breakdown

Class	Sensitivity	Precision	Balanced Accuracy
1	50.8%	64.6%	73.9%
2	0%	NaN	50.0%
3	13.8%	12.9%	54.6%
4	9.5%	50.0%	54.6%
5	0%	0%	49.8%
6	3.2%	7.7%	50.6%
7	27.1%	24.4%	58.2%

Class	Sensitivity	Precision	Balanced Accuracy
8	48.6%	35.3%	64.6%
9	2.1%	7.4%	48.6%
10	89.9%	50.8%	81.2%

1.3.2.1.2 Overall Model Performance

Accuracy: 0.3981

- The model correctly predicted the exact rating about 40% of the time.

No Information Rate (NIR): 0.2395

- Always guessing the most common rating (10) would be correct about 24% of the time.

P-value (Accuracy > NIR): < 2.2e-16

- The model significantly outperforms random guessing or defaulting to the majority class.

Kappa: 0.2672

- This indicates fair agreement beyond chance; falls within 0.2-0.4 range.

1.3.2.1.3 Model Strengths

- Performs best on class 10, the most common rating (recall = 89.9%)
- Moderate sensitivity for Classes 1 and 8, meaning the model is able to detect some negative (1) and mid-range (8) ratings well

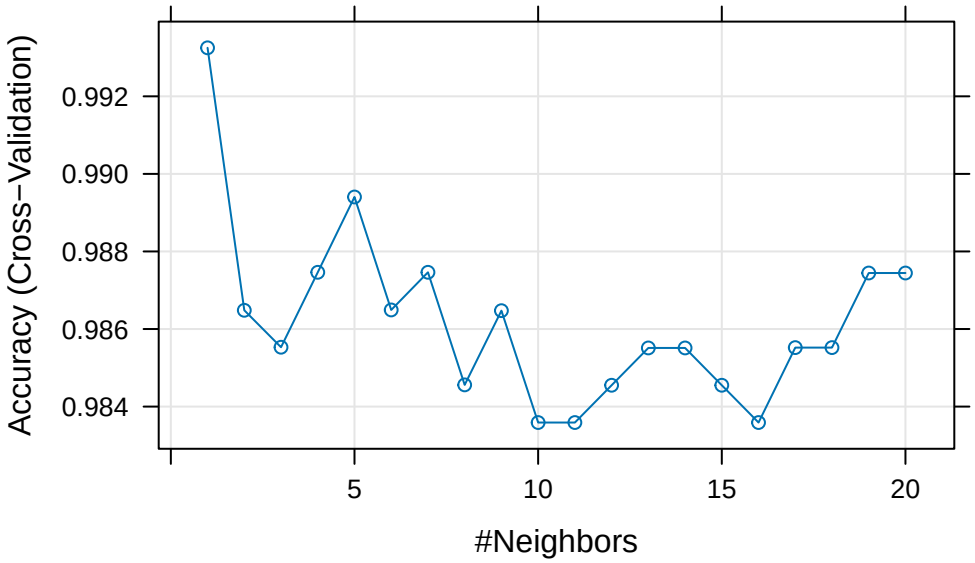
1.3.2.1.4 Model Weaknesses

- Struggles with middle ratings (2-6), very low recall and precision
- Leans heavily towards the most frequent classes, possibly due to class imbalance within the dataset

The neutral network model achieves moderate overall accuracy (39.8%) and significantly outperforms baseline guessing. However, it shows a strong bias towards frequent ratings like 10, while struggling to accurately detect rare or mid-range ratings (2-6). Future work or revisions should address the class imbalance to improve performance across all review scores.

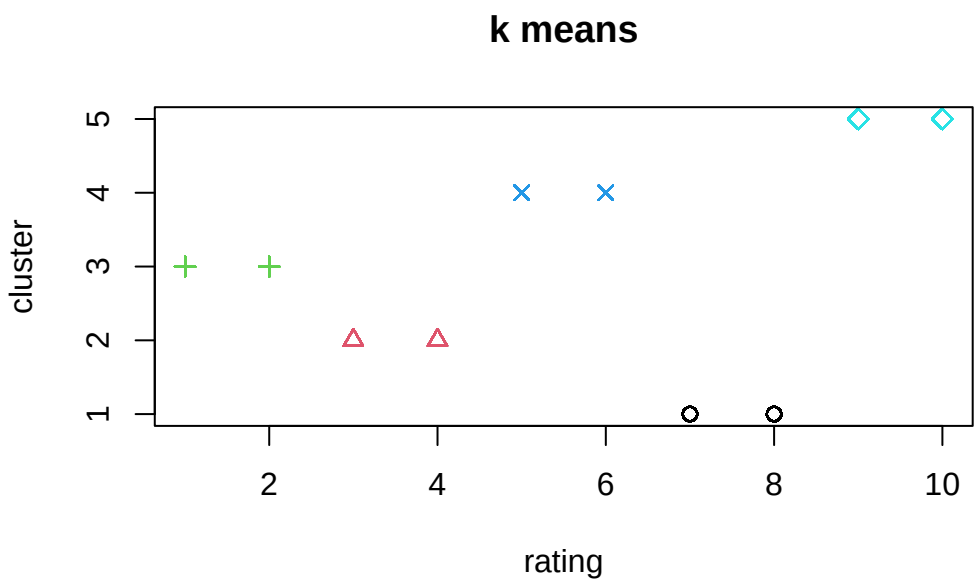
1.3.3 KNN (K-Nearest Neighbors)

Can we classify the sentiment of a new review based on sentiments of similar past reviews?




```
Confusion Matrix and Statistics

          Reference
Prediction Negative Positive
  Negative      973        0
  Positive        4     2130

               Accuracy : 0.9987
                 95% CI : (0.9967, 0.9996)
    No Information Rate : 0.6855
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.997

 Mcnemar's Test P-Value : 0.1336

            Sensitivity : 0.9959
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.9981
             Prevalence : 0.3145
         Detection Rate : 0.3132
   Detection Prevalence : 0.3132
      Balanced Accuracy : 0.9980

       'Positive' Class : Negative
```

1.3.4 K-means Clustering



1.4 Discussion

1.5 References

2 Code Appendix

```
#Load necessary libraries
library(dplyr) #data manipulation
library(knitr) #table formats
library(readr)
library(kableExtra)
library(caret)
library(forcats)
library(nnet)
library(kknn)
library(tidyr)
library(broom)
library(ggcorrplot)

#Import UC Irvine test csv file. (Initial Definition for All Code)
drug_train_tsv <- read_tsv("https://raw.githubusercontent.com/gkd5216/STAT380Project/refs/heads
drug_test_tsv <- read_tsv("https://raw.githubusercontent.com/gkd5216/STAT380Project/refs/heads/

# Summary Statistics grouping by Geographic Area.
variable_analysis <- data.frame(
  Variable = c("reviewID", "urlDrugName", "rating", "effectiveness", "sideEffects", "condition
```

```
              "sideEffectsReview", "commentsReview"),
  Type = c("Integer", "Categorial", "Integer", "Categorical", "Categorical", "Categorical",
           "Categorical", "Categorical", "Categorical"),
  Explanation = c(
    "Review ID",
    "Name of drug",
    "Name of condition",
    "Patient on benefits",
    "Patient on side effects",
    "Overall patient comment",
    "10 star patient rating",
    "5 step side effect rating",
    "5 step effectiveness rating"
  )
)

# Outputs Formatted Summary Table with Kable Styling tools
kable(
  variable_analysis,
  caption = "Variables used in Analysis"
  )

ggplot(drug_train_tsv, aes(x = rating)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Distribution of Patient Ratings (Train)", x = "Rating (1-10)", y = "Number of P

ggplot(drug_test_tsv, aes(x = rating)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +
  labs(title = "Distribution of Patient Ratings (Test)", x = "Rating (1-10)", y = "Number of Re
library(tidyverse)
library(janitor)
glimpse(drug_train_tsv)
glimpse(drug_test_tsv)


#Tidied drug train
#drug_train <- drug_train %>%
  #clean_names()

#Tidied drug test
#drug_test <- drug_test %>%
  #clean_names()

# Create binary sentiment variable: 1 (positive w/ rating 7-10), 0 (negative w/ rating 1-6)

drug_train <- drug_train_tsv %>%
  mutate(sentiment = ifelse(rating >= 7, 1, 0))
```

```r
drug_test <- drug_test_tsv %>%
  mutate(sentiment = ifelse(rating >= 7, 1, 0))

drug_train <- drug_train %>%
  mutate(across(c(urlDrugName, condition, effectiveness), as.factor))

drug_test <- drug_test %>%
  mutate(across(c(urlDrugName, condition, effectiveness), as.factor))

drug_train <- drug_train %>%
  mutate(urlDrugName = fct_lump(urlDrugName, n = 20),
         condition = fct_lump(condition, n = 20),
         effectiveness = fct_lump(effectiveness, n = 5))

# Match levels in test data to training data
drug_test <- drug_test %>%
  mutate(urlDrugName = fct_lump(urlDrugName, n = 20),
         condition = fct_lump(condition, n = 20),
         effectiveness = fct_lump(effectiveness, n = 5))

# Align factor levels
drug_test$urlDrugName <- factor(drug_test$urlDrugName, levels = levels(drug_train$urlDrugName))
drug_test$condition <- factor(drug_test$condition, levels = levels(drug_train$condition))
drug_test$effectiveness <- factor(drug_test$effectiveness, levels = levels(drug_train$effective

# Remove rows with NAs
test_data_clean <- drug_test %>%
  filter(!is.na(urlDrugName) & !is.na(condition) & !is.na(effectiveness))

# Logistic regression model
logit_model <- glm(sentiment ~ urlDrugName + condition + effectiveness,
                   data = drug_train, family = "binomial")

# Predict on clean test data
test_pred <- predict(logit_model, newdata = test_data_clean, type = "response")
test_pred_class <- ifelse(test_pred > 0.5, 1, 0)

# Confusion matrix
confusionMatrix(factor(test_pred_class), factor(test_data_clean$sentiment))

# Drop NAs
drug_train <- drug_train_tsv %>%
  select(rating, urlDrugName, condition, effectiveness) %>%
  drop_na()

# Treat rating as factor
drug_train$rating <- as.factor(drug_train$rating)
```

```r
drug_train <- drug_train %>%
  mutate(
    urlDrugName = fct_lump(urlDrugName, n = 20),
    condition = fct_lump(condition, n = 20),
    effectiveness = fct_lump(effectiveness, n = 5)
  )

# Split into training and validation sets
set.seed(42)
split_index <- createDataPartition(drug_train$rating, p = 0.8, list = FALSE)
train_set <- drug_train[split_index, ]
valid_set <- drug_train[-split_index, ]

# Train the neural network model
nn_model <- nnet(rating ~ urlDrugName + condition + effectiveness,
                 data = train_set,
                 size = 5,
                 maxit = 200,
                 trace = FALSE)

# Predict on validation set
nn_pred <- predict(nn_model, newdata = valid_set, type = "class")

nn_pred <- factor(nn_pred, levels = levels(valid_set$rating))
valid_rating <- factor(valid_set$rating, levels = levels(valid_set$rating))

# Evaluate model
confusionMatrix(nn_pred, valid_rating)

library(FNN)
set.seed(123)

#Turns categorial variables into numeric variables for computation
drug_train <- drug_train_tsv %>%
  mutate(
    rating = as.numeric(rating),
    sentiment = ifelse(rating >= 7, "Positive", "Negative")
    ) %>%
  drop_na(rating)

drug_test <- drug_test_tsv %>%
  mutate(
    rating = as.numeric(rating),
    sentiment = ifelse(rating >= 7, "Positive", "Negative")
    ) %>%
  drop_na(rating)
```

```r
#Top 5 drug names and conditions
top_drugs <- drug_train %>%
  count(urlDrugName, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(urlDrugName)

top_conditions <- drug_train %>%
  count(condition, sort = TRUE) %>%
  slice_head(n = 5) %>%
  pull(condition)

#Create dummy variables
drug_train <- drug_train %>%
  mutate(
    drug_top = ifelse(urlDrugName %in% top_drugs, urlDrugName, "Other"),
    condition_top = ifelse(condition %in% top_conditions, condition, "Other")
  ) %>%
  mutate(
    across(c(drug_top, condition_top), ~ as.factor(.))
  )

drug_test <- drug_test %>%
  mutate(
    drug_top = ifelse(urlDrugName %in% top_drugs, urlDrugName, "Other"),
    condition_top = ifelse(condition %in% top_conditions, condition, "Other")
  ) %>%
  mutate(
    across(c(drug_top, condition_top), ~ as.factor(.))
  )

#One-hot encode for drug_top and condition_top
drug_train_encode <- model.matrix(~ drug_top + condition_top - 1, data = drug_train)
drug_test_encode <- model.matrix(~ drug_top + condition_top - 1, data = drug_test)

#Scaling
drug_train$rating <- scale(drug_train$rating)
drug_test$rating <- scale(drug_test$rating)

#Final predictors matrix
train_predictors <- cbind(drug_train$rating, drug_train_encode)
test_predictors <- cbind(drug_test$rating, drug_test_encode)

train_labels <- factor(drug_train$sentiment)
test_labels <- factor(drug_test$sentiment)

#KNN Cross-Validation
knn_control <- trainControl(method = "cv", number = 10)
```

```r
knn_cv <- train(
  x = as.data.frame(train_predictors),
  y = train_labels,
  method = "knn",
  trControl = knn_control,
  tuneGrid = expand.grid(k = 1:20)
)

plot(knn_cv)
knn_predictions <- knn(
  train = train_predictors,
  test = test_predictors,
  cl = train_labels,
  k = knn_cv$bestTune$k
)

#Confusion matrix with KNN Predictions
confusionMatrix(knn_predictions, test_labels)

library(ggplot2)
library(factoextra)
set.seed(123)

drug_kmeans <- drug_train_tsv %>%
  mutate(rating = as.numeric(rating)) %>%
  drop_na(rating) %>%
  select(rating)

# perform kmeans clustering (k = 5 clusters)
kmeans_res <- kmeans(drug_kmeans, centers = 5, nstart = 20)
drug_KMeanClusters <- kmeans_res$cluster
drug_kmeans$cluster <- as.factor(drug_KMeanClusters)

#Plot k-means clustering
set.seed(123)
plot(drug_kmeans, pch = kmeans_res$cluster, col = kmeans_res$cluster, main = 'k means')
```