ADM first challenge - Final Report

This is a kind, inclusive, brave and failure-tolerant class

Meghan Walsh, Maya Ballard, Sierra Bouchard, Mary Kreklow

9/24/2024

Auxiliary functions

Function Definitions

plot_digit: Converts a row to a vector, to a matrix, allows us to visualize the specific digit

plot_region: Converts a tbl into a matrix, and allows us to visualize the region

calc_prop: Calculates the proportion of dark pixels within the region

```r
plot_region <- function(tbl) {
  digit_mat <- as.matrix(tbl)*128 # Convert tbl into matrix and assign gray=128

  image(t(digit_mat)[,28:1]) #Plot the image making sure is rotated
}
```

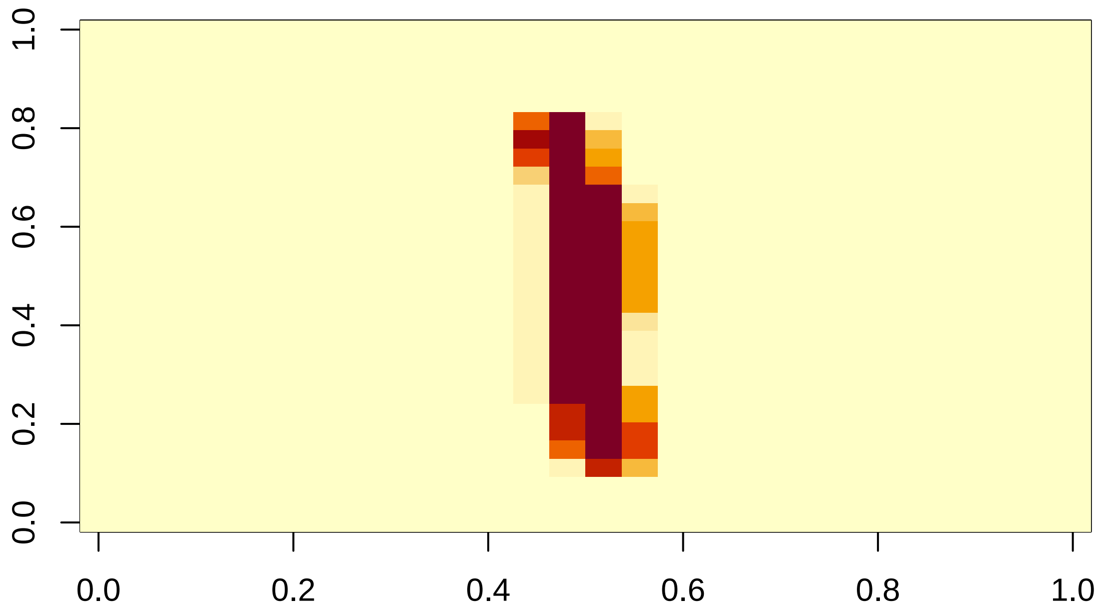
```r
calc_prop <- function (region, row) {
  # Take row from mnist and transform into a "digit" matrix
  digit_mat <-  row |>
    as.numeric()|>
    matrix(nrow = 28) |>
    t()
  # Find positions of pixels from "region"
  pos = (region==1)
  # Subset "digit" to the positions and count dark pixels (grey>20)
  dark = digit_mat[pos]>20
  # Return proportion of dark pixels of "image" in "region"
  return(sum(dark)/sum(pos))
}
```

Region description

For our first region, we drew two “less than” signs, trying to mimic the number three, hoping that the curves would help differentiate between the numbers. The second design for region two required some trial and error, we ended up making two vertical lines with space in the middle where a three normally go.

Image dataset

```r
mnist1_3_tbl<-read_csv("~/Mscs_341_F24/Class/Data/mnist.csv.gz") |>
  filter(digit==1|digit==3)|>
  mutate(digit=as.factor(digit)) |>
  slice_head(n=1000)

plot_digit(mnist1_3_tbl[7,])
```

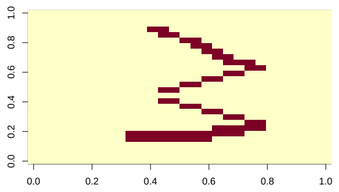


Code Description

The above chunk of code filters the digits our group was assigned (1 and 3). The mutate function converts it into a factor, which allows us to plot it. We plotted the 7th digit in the set using the plot digit function to create a visualization of our assigned digit of 1.

```r
region1 <- read_csv("~/Mscs_341_F24/Project/B_1/region1.csv", col_names = FALSE)
region2 <- read_csv("~/Mscs_341_F24/Project/B_1/region2.csv", col_names = FALSE)
```

```r
plot_region(region1)
```



```
calc_prop(region1, mnist1_3_tbl[1,2:785])
```

```
## [1] 0.3559322
```

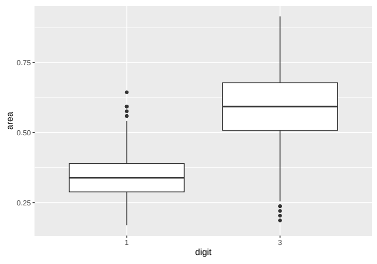
```
calc_prop(region1, mnist1_3_tbl[3,2:785])
```

```
## [1] 0.7457627
```

```
(tmp_tbl <- mnist1_3_tbl|>
  rowwise()|>
  mutate(area=calc_prop(region1,c_across(V1:V784))) |>
  ungroup()|>
  select(digit,area))
```

```
## # A tibble: 1,000 x 2
##    digit  area
##    <fct> <dbl>
##  1 1     0.356
##  2 1     0.407
##  3 3     0.746
##  4 1     0.254
##  5 3     0.593
##  6 3     0.712
##  7 1     0.254
##  8 1     0.356
##  9 1     0.322
## 10 3     0.831
## # i 990 more rows
```

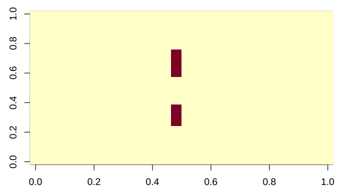
```
tmp_tbl|>
  ggplot(aes(x=digit, y=area))+
  geom_boxplot()
```



Code Description

This chunk of code accomplishes several items. First, we plotted our first region. The we calculated the proportion of pixels in our data set that overlapped with the region. We found that the 3 digits overlapped with this region about 75% of the time compared to the ones which overlapped 36% of the time. We then plotted the proportion of 1s and 3s for each regions using box plots.

```
plot_region(region2)
```



```r
calc_prop(region2, mnist1_3_tbl[1,2:785])
```

```
## [1] 0.4444444
```

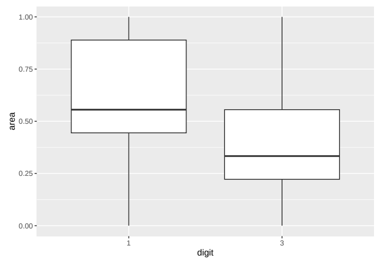
```r
calc_prop(region2, mnist1_3_tbl[3,2:785])
```

```
## [1] 0.2222222
```

```r
(tmp_tbl <- mnist1_3_tbl|>
  rowwise()|>
  mutate(area=calc_prop(region2,c_across(V1:V784))) |>
  ungroup()|>
  select(digit,area))
```

```
## # A tibble: 1,000 x 2
##    digit  area
##    <fct> <dbl>
##  1 1     0.444
##  2 1     0.889
##  3 3     0.222
##  4 1     0.778
##  5 3     0.667
##  6 3     0.333
##  7 1     1
##  8 1     0.333
##  9 1     0.778
## 10 3     0.556
## # i 990 more rows
```

```r
tmp_tbl|>
  ggplot(aes(x=digit, y=area))+
  geom_boxplot()
```



Code Description

This code is the same as above, just with our second region.

Feature dataset

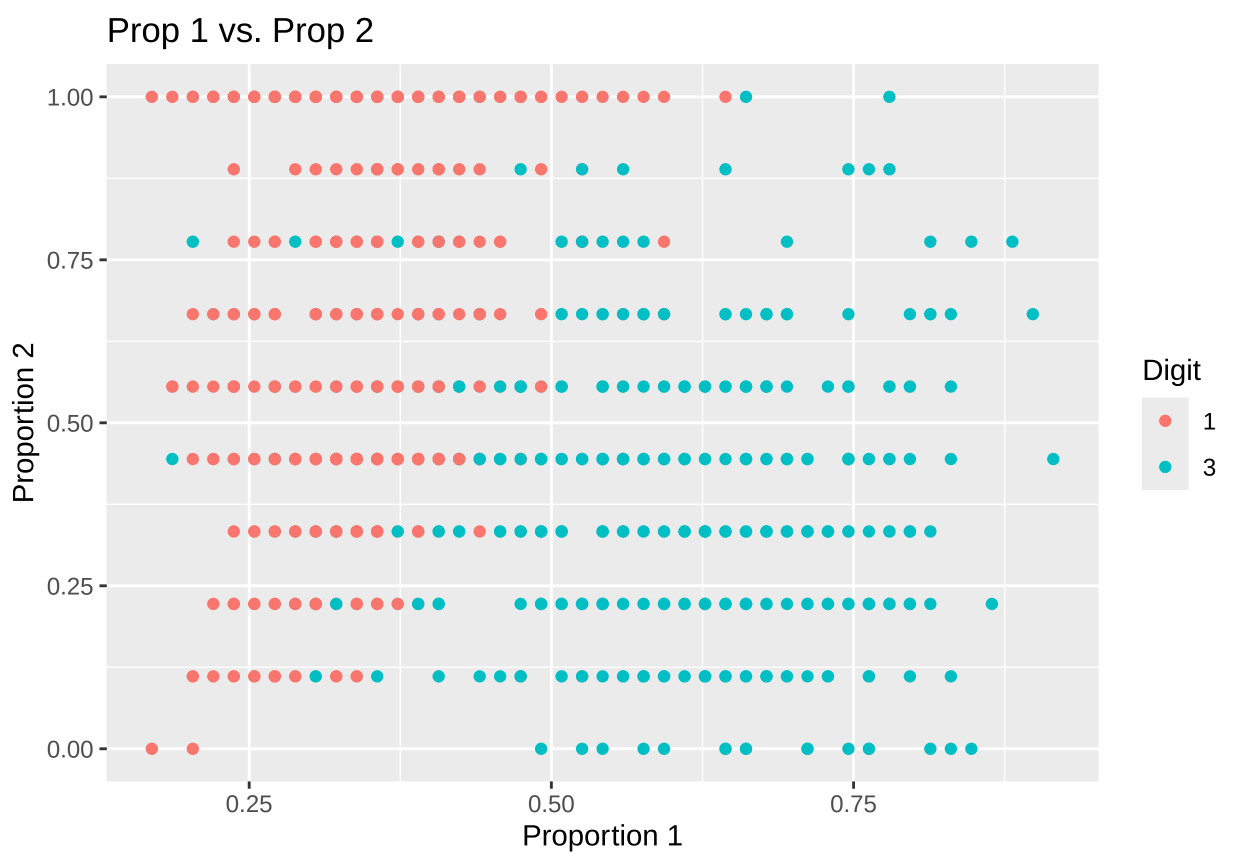
```
features_tbl <- mnist1_3_tbl|>
  rowwise()|>
  mutate(
         prop1 = calc_prop(region1,c_across(V1:V784)),
         prop2 =  calc_prop(region2,c_across(V1:V784))) |>
  ungroup()|>
  mutate(id = row_number())%>%
  select(id, digit,prop1, prop2)

write_csv(features_tbl, "~/Mscs_341_F24/Project/B_1/mnist1_3.features.csv")
```

Code Description

In this chunk we created a tibble with 4 variables: the id, digit, prop 1, and prop 2. Prop1 calculated the proportion of each digit overlap with region one. The same process took place with prop 2, this time with our second region. We then saved the data set.

Initial plotting

```
ggplot(features_tbl, aes(x=prop1, y=prop2, group = digit, color = as.factor(digit)))+
  geom_point()+
  labs(title = "Prop 1 vs. Prop 2",
       x = "Proportion 1",
       y = "Proportion 2",
       color = "Digit")
```



It looks like the ones are very distinguishable and the threes are also pretty good, although there are a few threes that are in with the ones more and have high prop 2/low prop 1 values.

Training/Testing Split

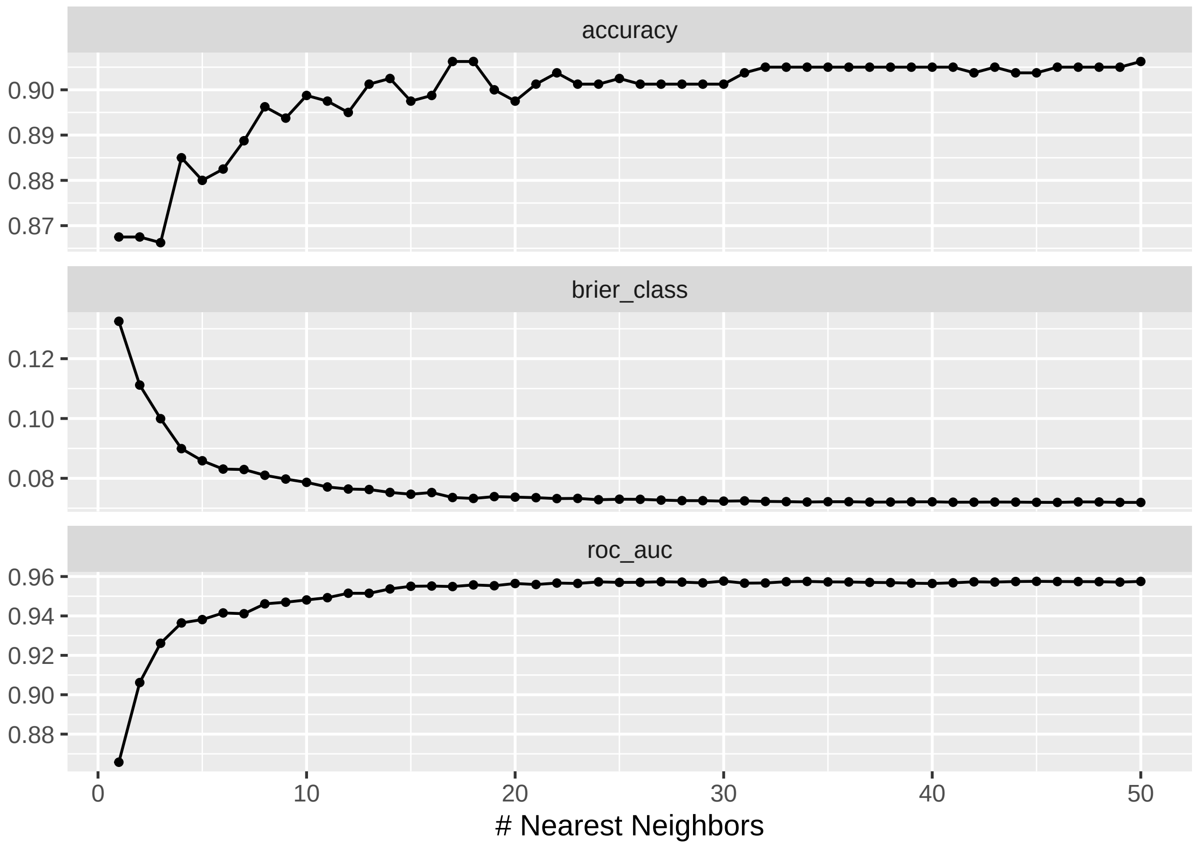
```
set.seed(12345)
feature_split<- initial_split(features_tbl, prop = 0.8)
feature_train <- training(feature_split)
feature_test <- testing(feature_split)
```

Split our data into testing and training dataset, with a proportion of 80% in our training and 20% in our testing.

Model creation, optimization and selection - Model 1

```
knn_model<- nearest_neighbor(neighbors = tune())%>%
  set_mode("classification")%>%
  set_engine("kknn")
recipe<- recipe(digit ~ prop1 + prop2, data = feature_train)
workflow<- workflow()%>%
  add_recipe(recipe)%>%
  add_model(knn_model)
```

```r
library(dials)
set.seed(12345)
feature_fold <- vfold_cv(feature_train, v = 5)
neighbors_tbl<- grid_regular(neighbors(range = c(1,50)), levels = 50)
tune_neighbor<- tune_grid(object = workflow, resamples = feature_fold, grid = neighbors_tbl)
autoplot(tune_neighbor)
```



```r
show_best(tune_neighbor, metric = "accuracy")
```

```
## # A tibble: 5 x 7
##   neighbors .metric  .estimator  mean     n std_err .config
##       <int> <chr>    <chr>      <dbl> <int>   <dbl> <chr>
## 1        17 accuracy binary     0.906     5  0.0110 Preprocessor1_Model17
## 2        18 accuracy binary     0.906     5  0.0133 Preprocessor1_Model18
## 3        50 accuracy binary     0.906     5  0.0123 Preprocessor1_Model50
## 4        32 accuracy binary     0.905     5  0.0132 Preprocessor1_Model32
## 5        33 accuracy binary     0.905     5  0.0132 Preprocessor1_Model33
```

```r
(best_neighbor<- select_best(tune_neighbor, metric = "accuracy"))
```

```
## # A tibble: 1 x 2
##   neighbors .config
##       <int> <chr>
## 1        17 Preprocessor1_Model17
```

```r
final_workflow <- finalize_workflow(workflow, best_neighbor)
final_fit <- fit(final_workflow, feature_train)
```

```
augment(final_fit, feature_test)%>%
  accuracy(truth = digit, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.92
```

Model 2

```
logistical_spec<- logistic_reg()%>%
  set_engine("glm")%>%
  set_mode("classification")
workflow_log<- workflow()%>%
  add_recipe(recipe)%>%
  add_model(logistical_spec)
logistical_fit <- fit(workflow_log, feature_train)
augment(logistical_fit, feature_test)%>%
  accuracy(digit, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary          0.93
```

```
augment(logistical_fit, feature_test)%>%
  conf_mat(digit, .pred_class)
```

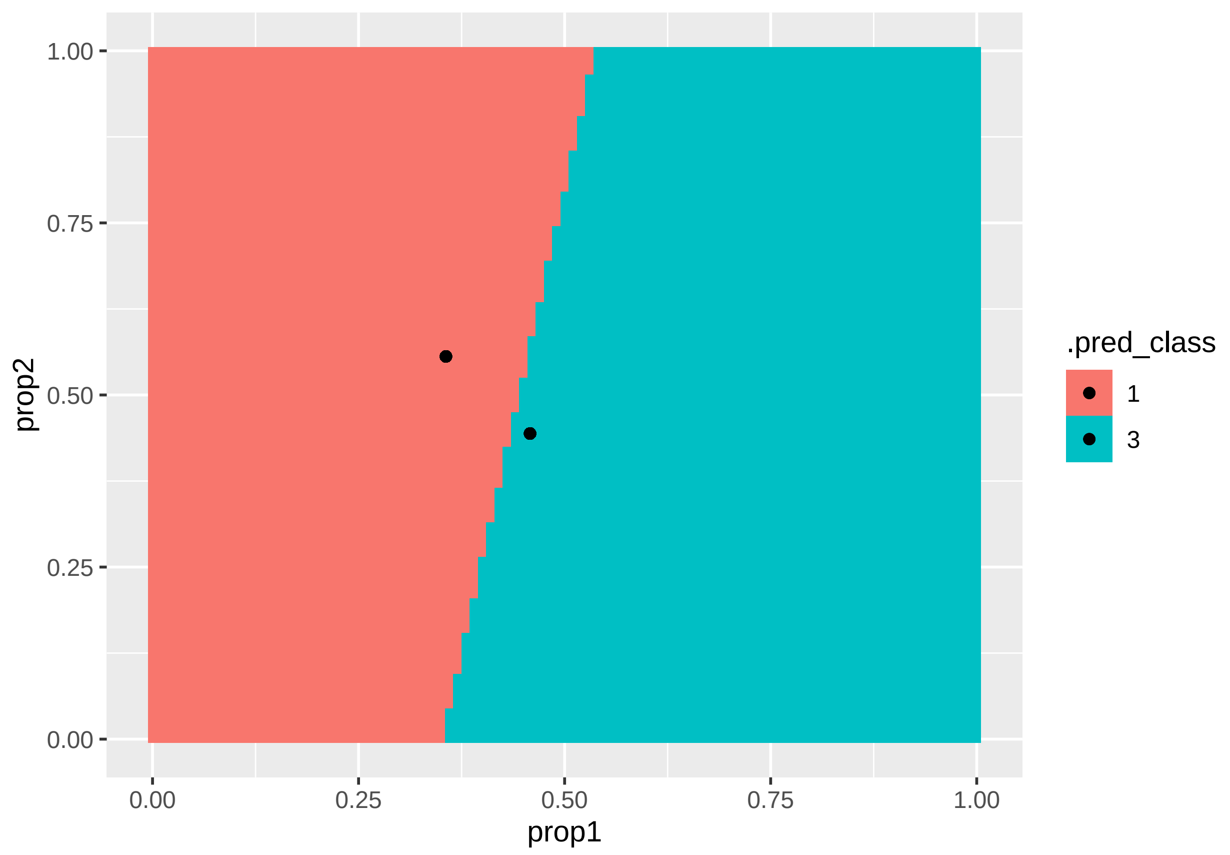
```
##           Truth
## Prediction   1   3
##          1 107   8
##          3   6  79
```

Code description

In the past two chunks we created our knn model and our multinomial logistical regression model. In the knn model we used cross validation to optimize our k value or our number of neighbors we used. We found that this model had a 92% accuracy. For the multinomial logistic regression model, we got a 93% accuracy, which was slightly better than our knn model. Thus we created a confusion matrix for this model.

Decision boundary

```
prop_grid<- expand_grid(prop1 = seq(0,1,.01), prop2 = seq(0,1,.01))
augment(logistical_fit, prop_grid)%>%
  ggplot(aes(x = prop1, y = prop2, fill = .pred_class))+
  geom_raster()+
  geom_point(aes(x = .458, y = .444))+
  geom_point(aes(x = .356, y = .556))
```



```r
#we plotted the two misclassifications we were looking at so we could see them better.
```

Code description

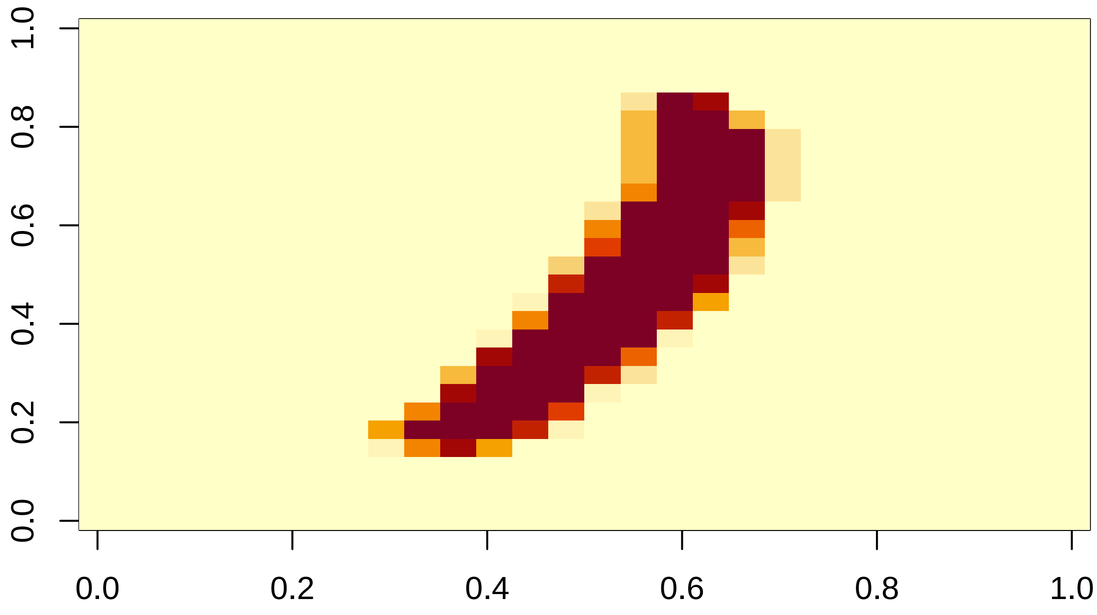
In this chunk we plotted our logistical fit, our more accurate model, predictions across a grid in `prop1` and `prop2` using `geom_raster()`.

Misclassifications

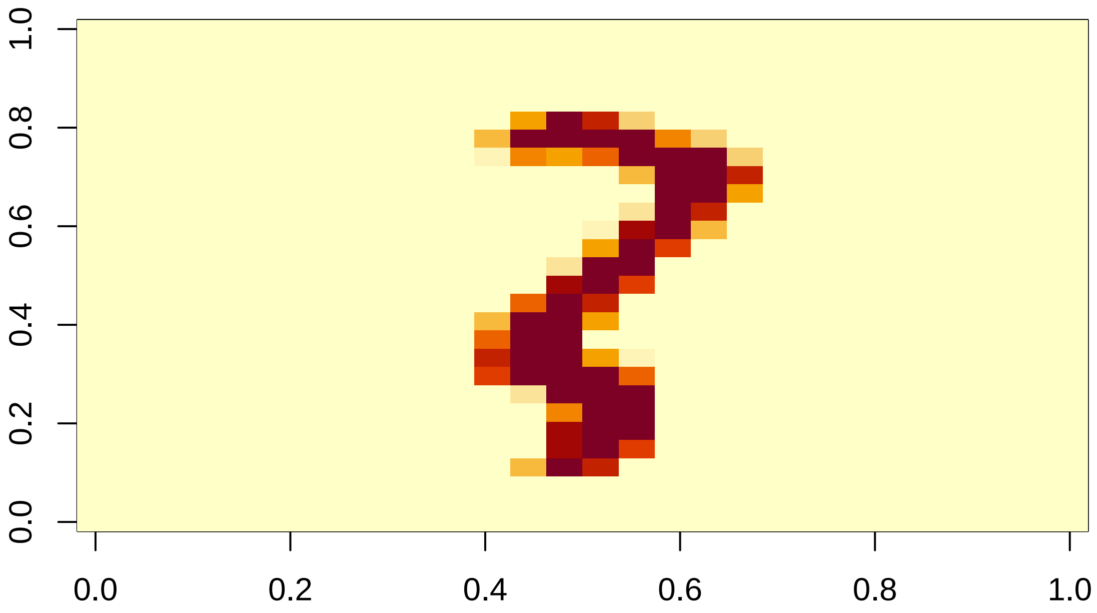
```r
augment(logistical_fit, feature_test)%>%
  filter(digit != .pred_class)
```

```
## # A tibble: 14 x 7
##    .pred_class .pred_1 .pred_3    id digit prop1 prop2
##    <fct>         <dbl>   <dbl> <int> <fct> <dbl> <dbl>
##  1 3             0.383  0.617     22 1     0.458 0.444
##  2 1             0.726  0.274    221 3     0.407 0.556
##  3 1             0.553  0.447    236 3     0.424 0.444
##  4 1             0.969  0.0306   237 3     0.305 0.667
##  5 3             0.154  0.846    288 1     0.559 0.667
##  6 1             0.789  0.211    402 3     0.390 0.556
##  7 1             0.830  0.170    501 3     0.356 0.444
##  8 3             0.306  0.694    693 1     0.475 0.444
##  9 3             0.401  0.599    762 1     0.475 0.556
## 10 1              0.553  0.447    808 3     0.424 0.444
## 11 3              0.122  0.878    850 1     0.593 0.778
## 12 1              0.808  0.192    870 3     0.322 0.222
## 13 3              0.467  0.533    940 1     0.441 0.444
## 14 1              0.881  0.119    975 3     0.356 0.556
```

```
plot_digit(mnist1_3_tbl[22,])
```



```
plot_digit(mnist1_3_tbl[975,])
```



Code Description

Here we filtered the misclassified digits and selected two of them. We picked id 22 and id 975 as our two misclassified digits. For 22 the value of prop1 was .458 and the value of prop2 was .444. For 975 the value of prop1 was .356 and the value of prop2 was .556. The digits were misclassfied because the proportions were in the decision boundary of the opposite digit. They also were close to the decision boundary. The model gives the probability of our first digit being a 3 a value of 62%. The second digit has a probability of being a 3 of 88%. Looking at the digits, we observed that they were very oddly shaped and did not look like a typical number.

Changing things up

```
mnist_6tbl<-read_csv("~/Mscs_341_F24/Class/Data/mnist.csv.gz") |>
  filter(digit==6)|>
  mutate(digit=as.factor(digit)) |>
  slice_head(n=500)
mnist1_3_6_tbl <-mnist_6tbl%>%
  full_join(mnist1_3_tbl)

write_csv(mnist1_3_6_tbl,  "~/Mscs_341_F24/Project/B_1/mnist1_3_6.csv")
```

```
new_features_tbl <- mnist1_3_6_tbl|>
  rowwise()|>
  mutate(
         prop1 = calc_prop(region1,c_across(V1:V784)),
         prop2 =  calc_prop(region2,c_across(V1:V784))) |>
  ungroup()|>
  mutate(id = row_number())%>%
  select(id, digit,prop1, prop2)

write_csv(new_features_tbl, "~/Mscs_341_F24/Project/B_1/mnist1_3_6.features.csv")

set.seed(12345)
feature_6_split <- initial_split(new_features_tbl, prop = .8)
feature_6_train <- training(feature_6_split)
feature_6_test <- testing(feature_6_split)
```

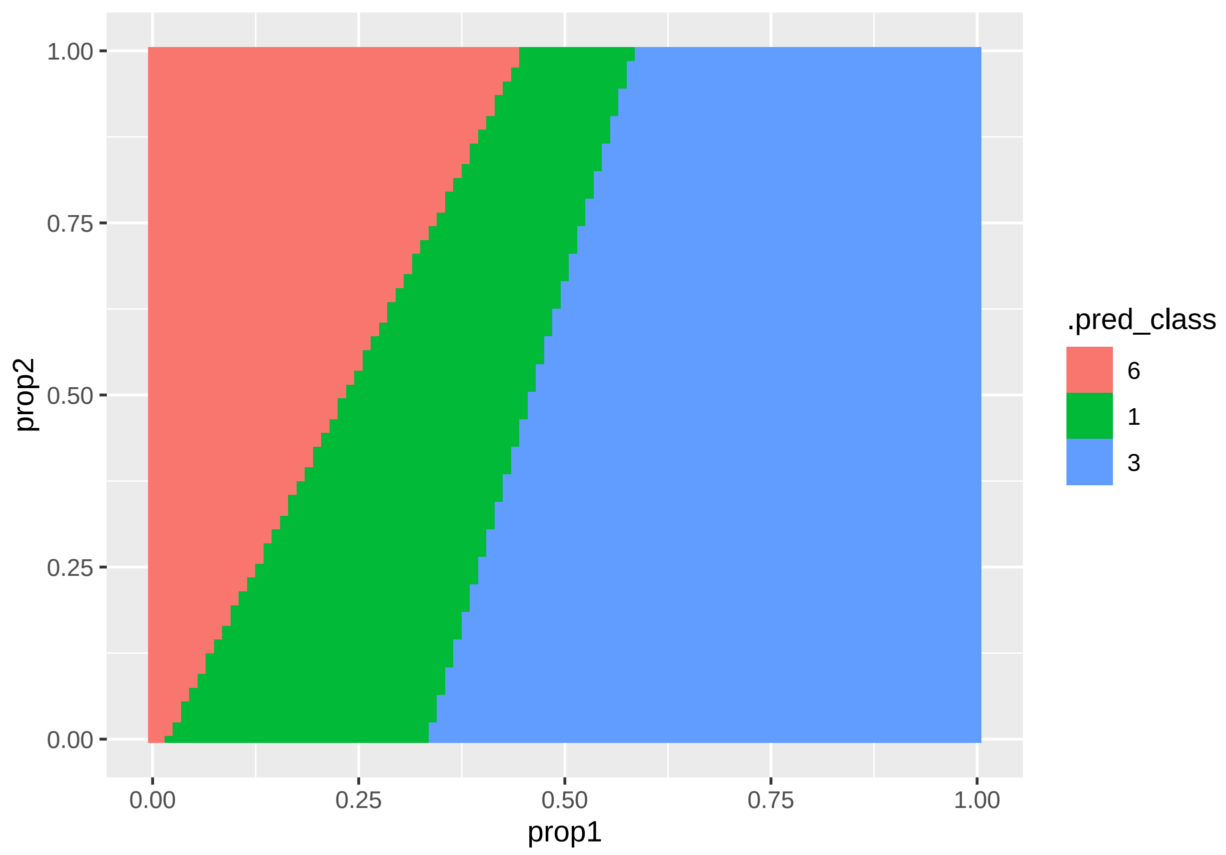
```
new_logistical_spec<- multinom_reg()%>%
  set_engine("nnet")%>%
  set_mode("classification")
new_recipe <- recipe(digit~prop1+prop2, feature_6_train)
workflow_log<- workflow()%>%
  add_recipe(new_recipe)%>%
  add_model(new_logistical_spec)
new_logistical_fit <- fit(workflow_log, feature_6_train)
augment(new_logistical_fit, feature_6_test)%>%
  accuracy(digit, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass      0.74
```

```
augment(new_logistical_fit, feature_6_test)%>%
  conf_mat(digit, .pred_class)
```

```
##           Truth
## Prediction  6  1  3
##          6 68 27  1
##          1 28 63 11
##          3  7  4 91
```

```
augment(new_logistical_fit, prop_grid)%>%
  ggplot(aes(x = prop1, y = prop2, fill = .pred_class))+
  geom_raster()
```



Code description

In this chunk we created our new model using the dataset with 1,3,6 digits. We then created the model and plotted the new decision boundary and calculated our new accuracy and confusion matrix. We found that the 1s were misclassified as 6s much more than 3s which makes sense because our region we created to catch the 3s was very distinct.