HWK6

cristian razo

5/23/2022

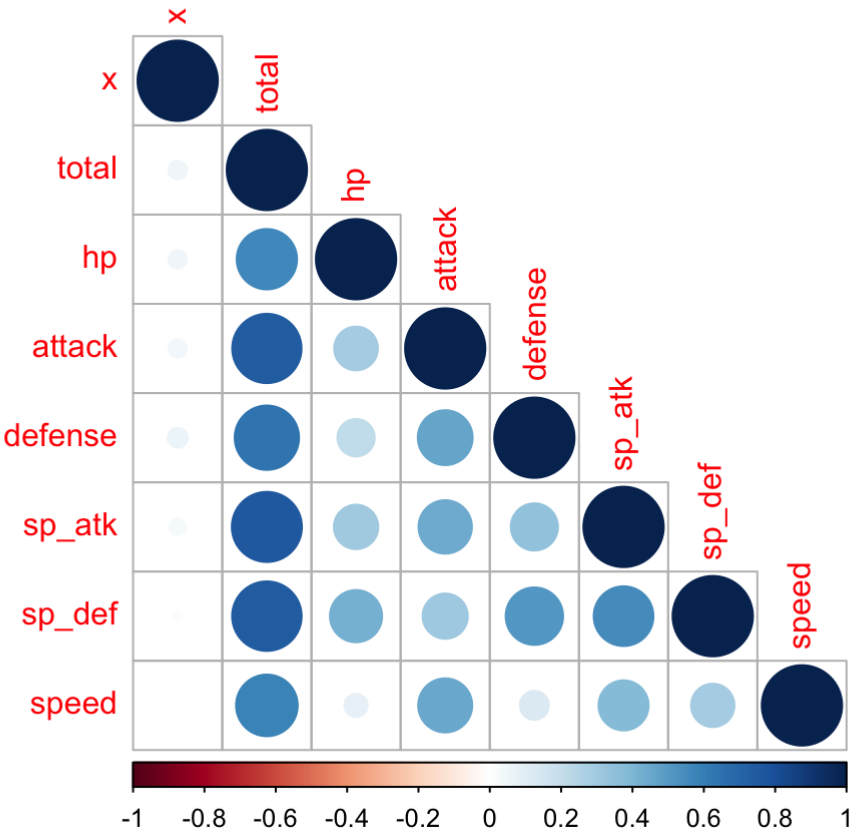
1

```
dt=read.csv("Downloads/homework-6/data/Pokemon.csv")

clean_dt=dt%>%
  clean_names()


clean_dt=clean_dt[clean_dt$type_1 %in% c('Bug', 'Fire', 'Grass', 'Normal', 'Water',  'Ps
ychic'), ]


clean_dt$type_1 <- as.factor(clean_dt$type_1)
clean_dt$legendary <- as.factor(clean_dt$legendary)
clean_dt$generation <- as.factor(clean_dt$generation)


split <- initial_split(clean_dt, prop = 0.80, strata = type_1)
train <- training(split)
test <- testing(split)



k_folds=vfold_cv(train,v=5)


pokemonster_recipe=recipe(type_1 ~ legendary+generation+sp_atk+attack+speed+defense+hp+s
p_def,data=clean_dt) %>%
  step_dummy(c(legendary,generation)) %>%
  step_normalize(all_predictors())
```

This is the preprocessing part of the model. Splitted it into a training and testing set and then did 5 k fold validation on the training set. Made a recipe that deals with my dummy variables and also centered and scaled all my predictors in the step normlize portion.

2

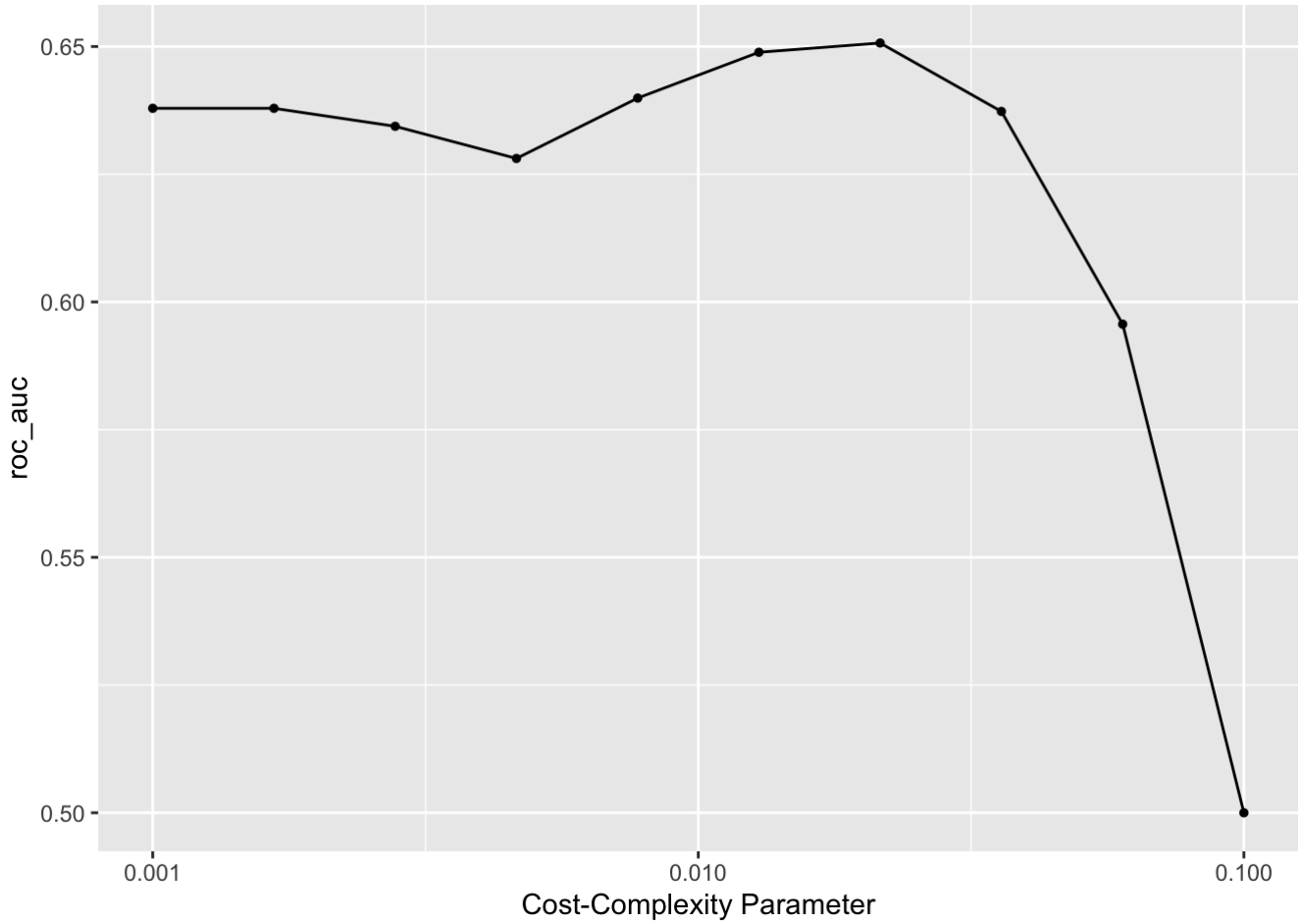
```
train_num=data.frame(select_if(train,is.numeric))
corrplot(cor(train_num),type = "lower")
```



This is my correlation plot we can see which featues are correlated with each other. Based on the plot all atributes have a significant positive correlation. This means each feature greatly situated with each other. Total is the column that is most influential to all other columns.Yes these relationships do make sense to me .

3

```
tree_spec <- decision_tree() %>%
  set_engine("rpart")

class_tree_spec <- tree_spec %>%
  set_mode("classification")


auto_wrkflw=workflow()%>%
  add_recipe((pokemonster_recipe)) %>%
  add_model(class_tree_spec %>% set_args(cost_complexity = tune()))

param_grid <- grid_regular(cost_complexity(range = c(-3, -1)), levels = 10)

tune_res <- tune_grid(
  auto_wrkflw,
  resamples = k_folds,
  grid = param_grid,
  metrics = metric_set(roc_auc)
  )

autoplot(tune_res)
```



This shows that we created a tuned classification decission tree with workflow that combines our model and recipe together. In our visual it shows theres a point where our model starts to overfit. Once our models meets a certrain threshold for the cost complexity parameter. It does better with a small complexity value.The ideal

complexity is the the point with the highest roc_auc score .

4

```
metrics_1=collect_metrics(tune_res)
arrange(metrics_1,mean)
```

```
## # A tibble: 10 × 7
##    cost_complexity .metric .estimator  mean     n std_err .config
##              <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1         0.1     roc_auc hand_till  0.5       5 0       Preprocessor1_Model10
##  2         0.0599  roc_auc hand_till  0.596     5 0.0262  Preprocessor1_Model09
##  3         0.00464 roc_auc hand_till  0.628     5 0.0201  Preprocessor1_Model04
##  4         0.00278 roc_auc hand_till  0.634     5 0.0193  Preprocessor1_Model03
##  5         0.0359  roc_auc hand_till  0.637     5 0.0170  Preprocessor1_Model08
##  6         0.001   roc_auc hand_till  0.638     5 0.0210  Preprocessor1_Model01
##  7         0.00167 roc_auc hand_till  0.638     5 0.0210  Preprocessor1_Model02
##  8         0.00774 roc_auc hand_till  0.640     5 0.0288  Preprocessor1_Model05
##  9         0.0129  roc_auc hand_till  0.649     5 0.0294  Preprocessor1_Model06
## 10         0.0215  roc_auc hand_till  0.651     5 0.00610 Preprocessor1_Model07
```

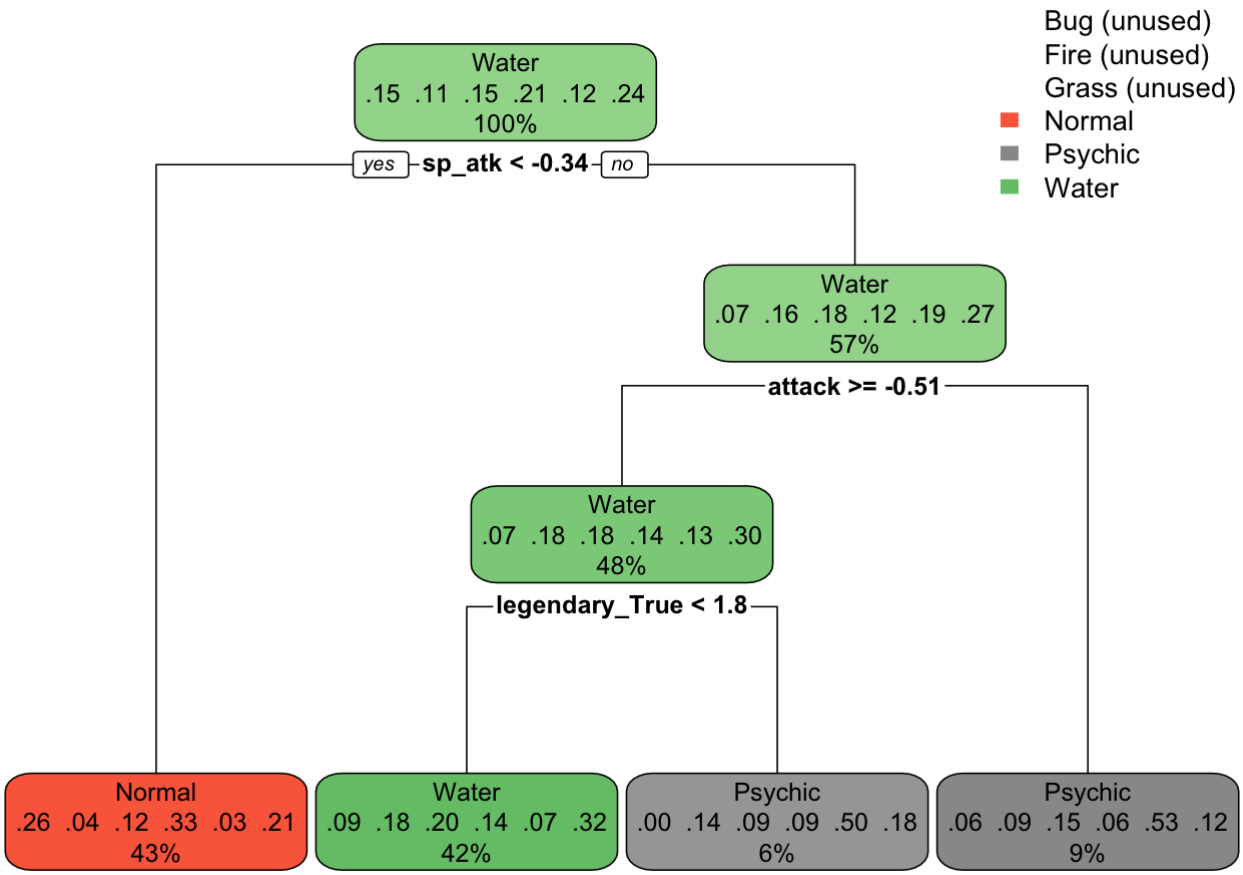
```
best=select_best(tune_res)
```

We see all the ROC_AUC score and choose the parameter with the best ROC_AUC score. The best score is .688

5

```
class_tree_final <- finalize_workflow(auto_wrkflw, best)
class_tree_final_fit <- fit(class_tree_final,train)

class_tree_final_fit %>%
  extract_fit_engine() %>%
  rpart.plot()
```

```
## Warning: Cannot retrieve the data used to build the model (so cannot determine roundi
nt and is.binary for the variables).
## To silence this warning:
##     Call rpart.plot with roundint=FALSE,
##     or rebuild the rpart model with model=TRUE.
```



The decission plot visual shows us the branch that consist of our decission tree model.

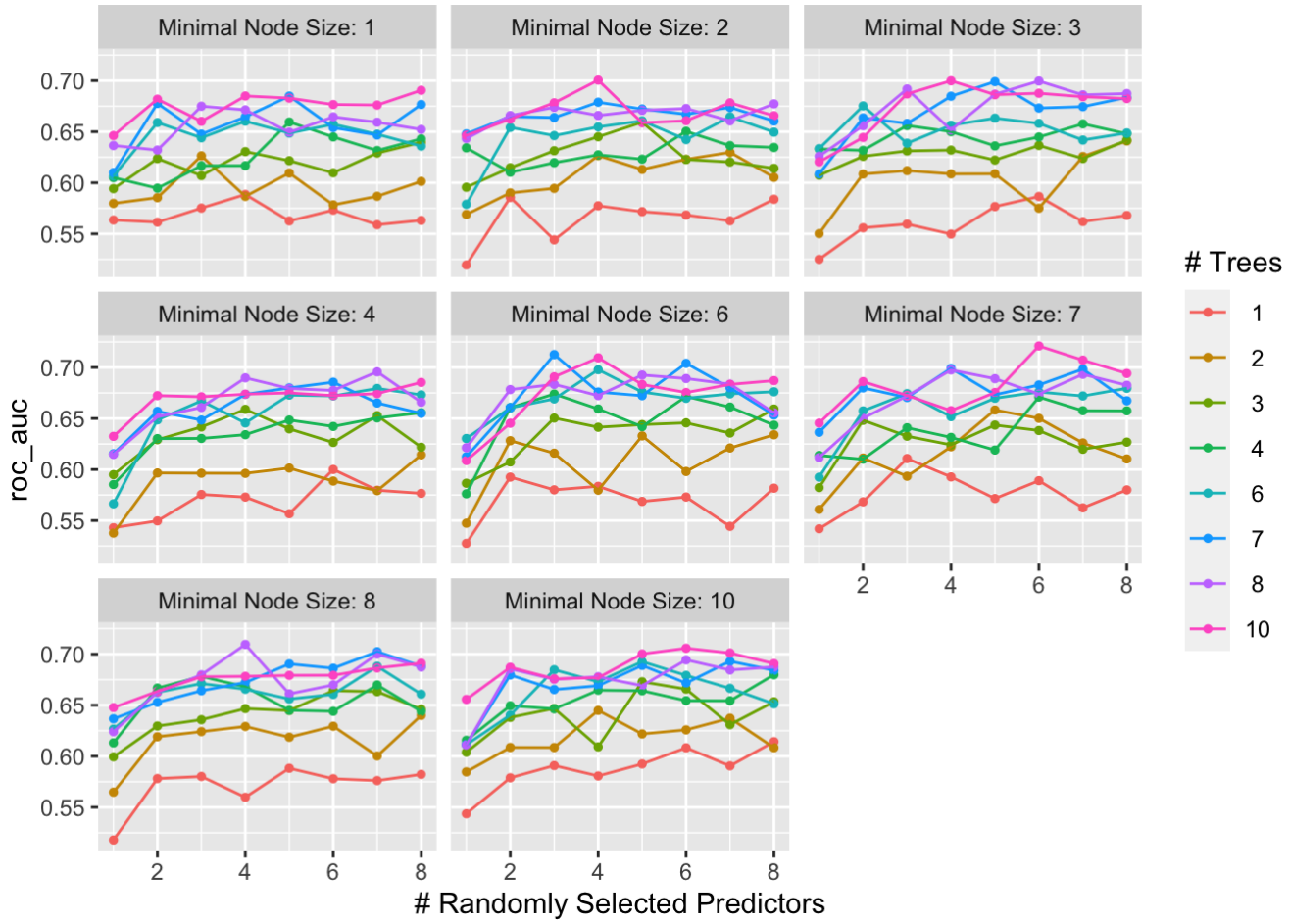
5 second part

```
rf_spec <- rand_forest(mtry = tune(),trees=tune(),min_n=tune())%>%
  set_engine("ranger", importance = 'impurity') %>%
  set_mode("classification")



auto_wrkflw2=workflow()%>%
  add_recipe((pokemonster_recipe)) %>%
  add_model(rf_spec)

grid_pen=grid_regular(mtry(range = c(1,8)),trees(range = c(1,10)),min_n(range = c(1,10
)),levels =8)

tune_res2=tune_grid(
  object = auto_wrkflw2,
  resamples = k_folds,
  grid = grid_pen,
  metrics = metric_set(roc_auc)
  )
```

It shound not be lower than 1 because we need to start somewhere and we should not go over 8 because we are only using 8 predictors to fit in our model.mtry= 8 means what we will do a split on each randomly sampled variable.

6 and 7

```
autoplot(tune_res2)
```




```
metrics_2=collect_metrics(tune_res2)
arrange(metrics_1,mean)
```

```
## # A tibble: 10 × 7
##    cost_complexity .metric .estimator  mean     n std_err .config
##              <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1         0.1     roc_auc hand_till  0.5       5 0       Preprocessor1_Model10
##  2         0.0599  roc_auc hand_till  0.596     5 0.0262  Preprocessor1_Model09
##  3         0.00464 roc_auc hand_till  0.628     5 0.0201  Preprocessor1_Model04
##  4         0.00278 roc_auc hand_till  0.634     5 0.0193  Preprocessor1_Model03
##  5         0.0359  roc_auc hand_till  0.637     5 0.0170  Preprocessor1_Model08
##  6         0.001   roc_auc hand_till  0.638     5 0.0210  Preprocessor1_Model01
##  7         0.00167 roc_auc hand_till  0.638     5 0.0210  Preprocessor1_Model02
##  8         0.00774 roc_auc hand_till  0.640     5 0.0288  Preprocessor1_Model05
##  9         0.0129  roc_auc hand_till  0.649     5 0.0294  Preprocessor1_Model06
## 10         0.0215  roc_auc hand_till  0.651     5 0.00610 Preprocessor1_Model07
```

```
best2=select_best(tune_res2)
```

We see all the ROC_AUC score and choose the parameter with the best ROC_AUC score. It seems that trees with high hyperprameter values yield the best results for our model. The best roc_auc score is .775

8

class_rand_final_fit%>% vip(geom='point')

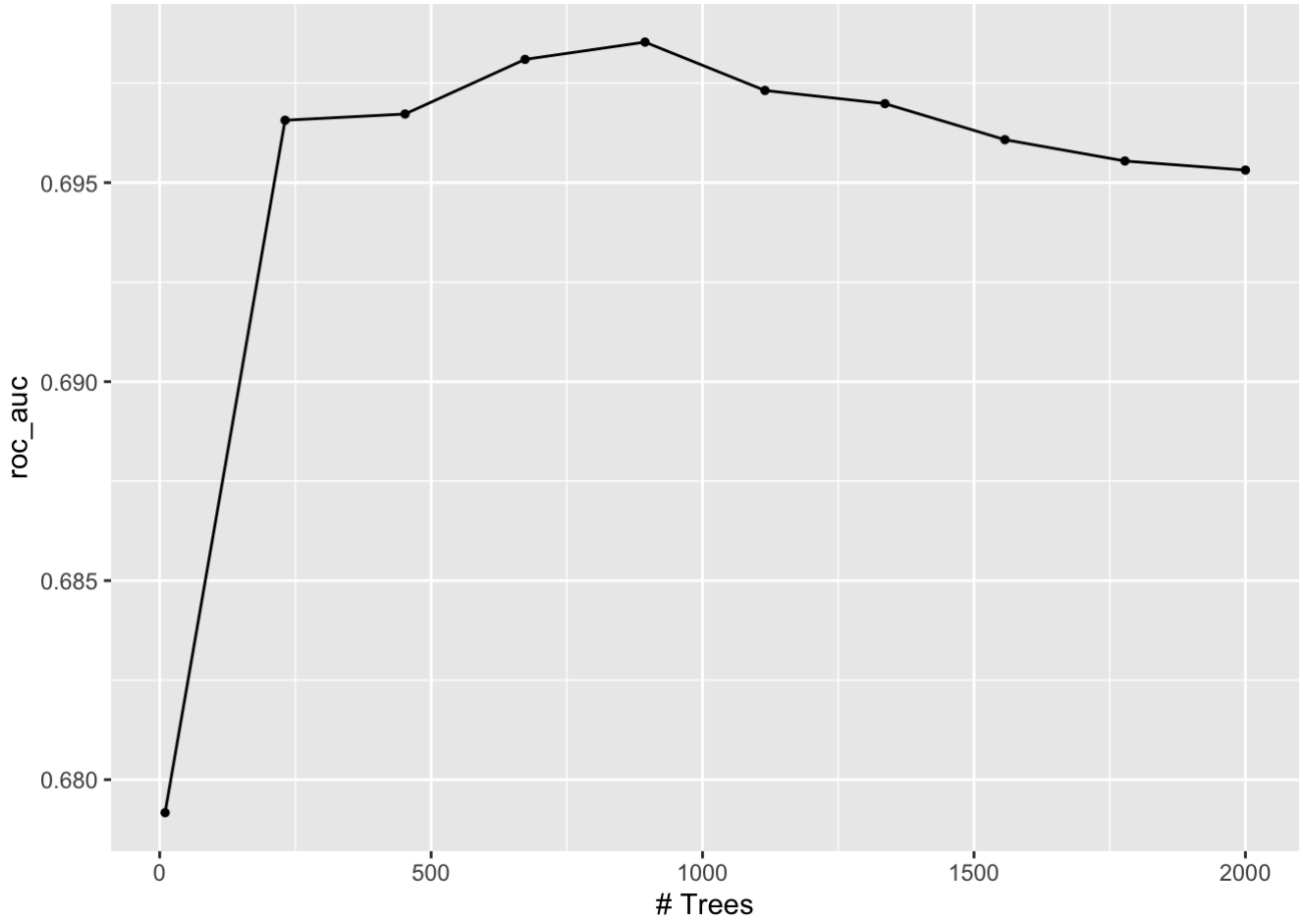
We get an error message I couldnt find the vip

9

```
boost_spec <- boost_tree(trees = tune() )%>%
  set_engine("xgboost") %>%
  set_mode("classification")



auto_wrkflw3=workflow()%>%
  add_recipe((pokemonster_recipe)) %>%
  add_model(boost_spec)

grid_trees=grid_regular(trees(range = c(10,2000)),levels =10)


tune_res3=tune_grid(
  object = auto_wrkflw3,
  resamples = k_folds,
  grid = grid_trees,
  metrics = metric_set(roc_auc)
)
```

```
## Warning: package 'xgboost' was built under R version 4.1.2
```

```
autoplot(tune_res3)
```




```
metrics_3=collect_metrics(tune_res3)
arrange(metrics_3,mean)
```

```
## # A tibble: 10 × 7
##    trees .metric .estimator  mean     n std_err .config
##    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1    10 roc_auc hand_till  0.679     5  0.0237 Preprocessor1_Model01
##  2  2000 roc_auc hand_till  0.695     5  0.0233 Preprocessor1_Model10
##  3  1778 roc_auc hand_till  0.696     5  0.0235 Preprocessor1_Model09
##  4  1557 roc_auc hand_till  0.696     5  0.0237 Preprocessor1_Model08
##  5   231 roc_auc hand_till  0.697     5  0.0247 Preprocessor1_Model02
##  6   452 roc_auc hand_till  0.697     5  0.0247 Preprocessor1_Model03
##  7  1336 roc_auc hand_till  0.697     5  0.0237 Preprocessor1_Model07
##  8  1115 roc_auc hand_till  0.697     5  0.0240 Preprocessor1_Model06
##  9   673 roc_auc hand_till  0.698     5  0.0247 Preprocessor1_Model04
## 10   894 roc_auc hand_till  0.699     5  0.0240 Preprocessor1_Model05
```

```
best3=select_best(tune_res3)
```

We see all the ROC_AUC score and choose the parameter with the best ROC_AUC score.We see that when we a bagging tree the highest score is when it has the least amount of trees . The best score for this model is .691.

10

```
arrange(metrics_1,mean)
```

```
## # A tibble: 10 × 7
##    cost_complexity .metric .estimator  mean     n std_err .config
##              <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1         0.1     roc_auc hand_till  0.5       5 0       Preprocessor1_Model10
##  2         0.0599  roc_auc hand_till  0.596     5 0.0262  Preprocessor1_Model09
##  3         0.00464 roc_auc hand_till  0.628     5 0.0201  Preprocessor1_Model04
##  4         0.00278 roc_auc hand_till  0.634     5 0.0193  Preprocessor1_Model03
##  5         0.0359  roc_auc hand_till  0.637     5 0.0170  Preprocessor1_Model08
##  6         0.001   roc_auc hand_till  0.638     5 0.0210  Preprocessor1_Model01
##  7         0.00167 roc_auc hand_till  0.638     5 0.0210  Preprocessor1_Model02
##  8         0.00774 roc_auc hand_till  0.640     5 0.0288  Preprocessor1_Model05
##  9         0.0129  roc_auc hand_till  0.649     5 0.0294  Preprocessor1_Model06
## 10         0.0215  roc_auc hand_till  0.651     5 0.00610 Preprocessor1_Model07
```

```
arrange(metrics_2,mean)
```

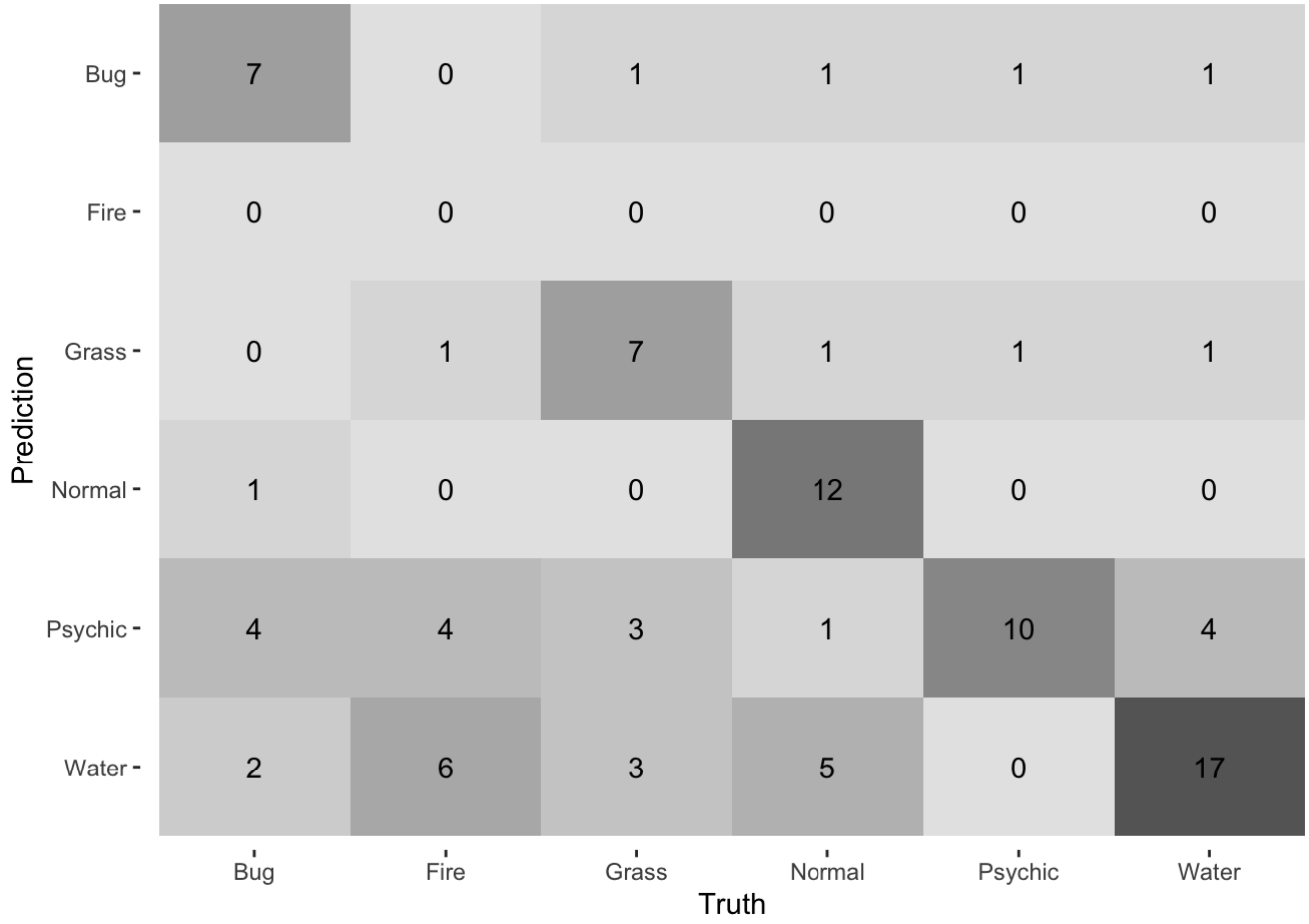
```
## # A tibble: 512 × 9
##     mtry trees min_n .metric .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     1     1     8 roc_auc hand_till  0.518     5 0.0183  Preprocessor1_Model…
##  2     1     1     2 roc_auc hand_till  0.520     5 0.0103  Preprocessor1_Model…
##  3     1     1     3 roc_auc hand_till  0.525     5 0.0206  Preprocessor1_Model…
##  4     1     1     6 roc_auc hand_till  0.528     5 0.0240  Preprocessor1_Model…
##  5     1     2     4 roc_auc hand_till  0.538     5 0.0377  Preprocessor1_Model…
##  6     1     1     7 roc_auc hand_till  0.542     5 0.0125  Preprocessor1_Model…
##  7     1     1     4 roc_auc hand_till  0.543     5 0.0148  Preprocessor1_Model…
##  8     1     1    10 roc_auc hand_till  0.544     5 0.0194  Preprocessor1_Model…
##  9     3     1     2 roc_auc hand_till  0.544     5 0.00970 Preprocessor1_Model…
## 10     7     1     6 roc_auc hand_till  0.544     5 0.0261  Preprocessor1_Model…
## # … with 502 more rows
```

```
arrange(metrics_3,mean)
```

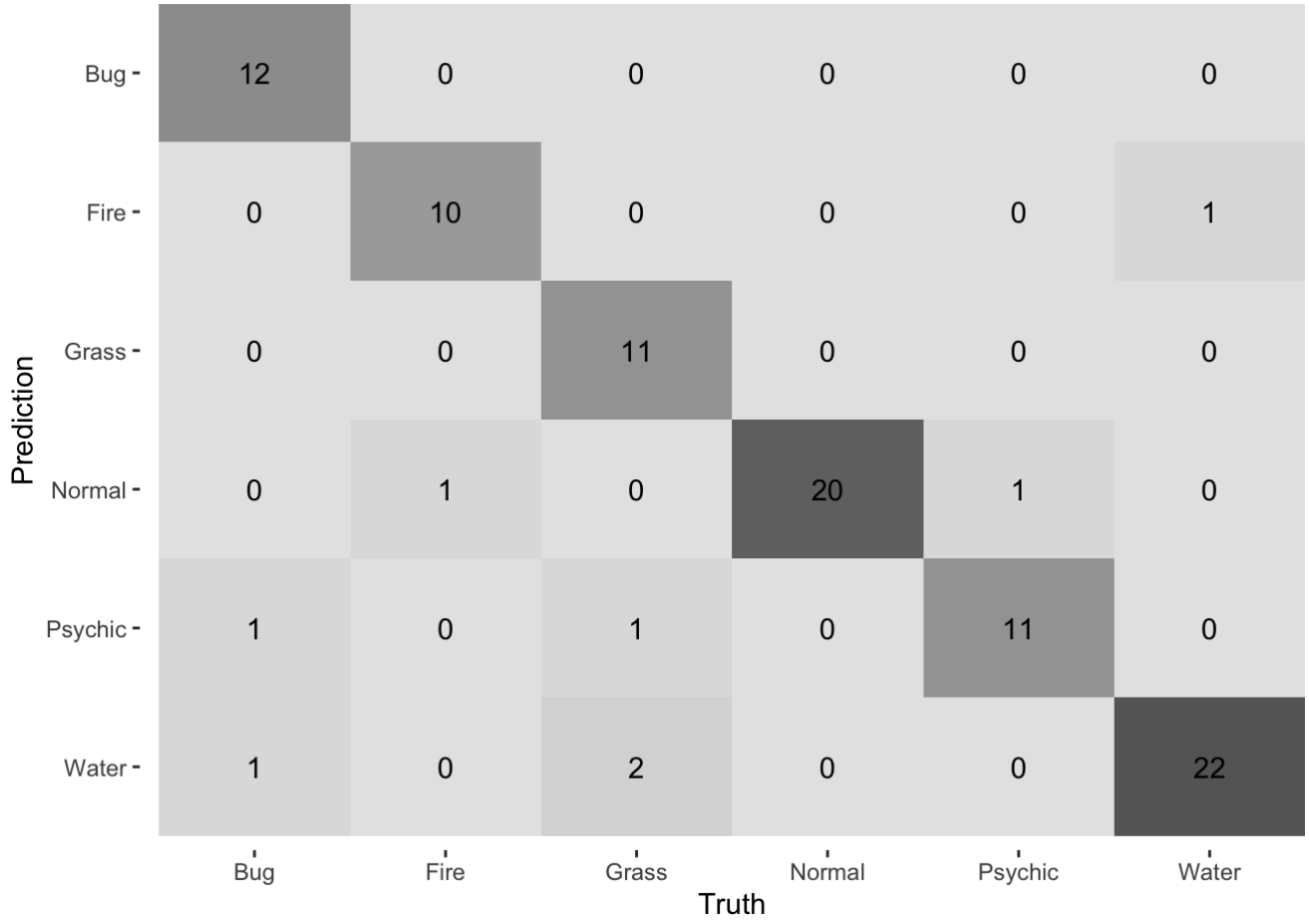
```
## # A tibble: 10 × 7
##    trees .metric .estimator  mean     n std_err .config
##    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1    10 roc_auc hand_till  0.679     5  0.0237 Preprocessor1_Model01
##  2  2000 roc_auc hand_till  0.695     5  0.0233 Preprocessor1_Model10
##  3  1778 roc_auc hand_till  0.696     5  0.0235 Preprocessor1_Model09
##  4  1557 roc_auc hand_till  0.696     5  0.0237 Preprocessor1_Model08
##  5   231 roc_auc hand_till  0.697     5  0.0247 Preprocessor1_Model02
##  6   452 roc_auc hand_till  0.697     5  0.0247 Preprocessor1_Model03
##  7  1336 roc_auc hand_till  0.697     5  0.0237 Preprocessor1_Model07
##  8  1115 roc_auc hand_till  0.697     5  0.0240 Preprocessor1_Model06
##  9   673 roc_auc hand_till  0.698     5  0.0247 Preprocessor1_Model04
## 10   894 roc_auc hand_till  0.699     5  0.0240 Preprocessor1_Model05
```

```
class_tree_final <- finalize_workflow(auto_wrkflw, best)
class_tree_final_fit <- fit(class_tree_final,test)

class_rand_final <- finalize_workflow(auto_wrkflw2, best2)
class_rand_final_fit <- fit(class_rand_final,test)

class_boost_final <- finalize_workflow(auto_wrkflw3, best3)
class_boost_final_fit <- fit(class_boost_final,test)
```

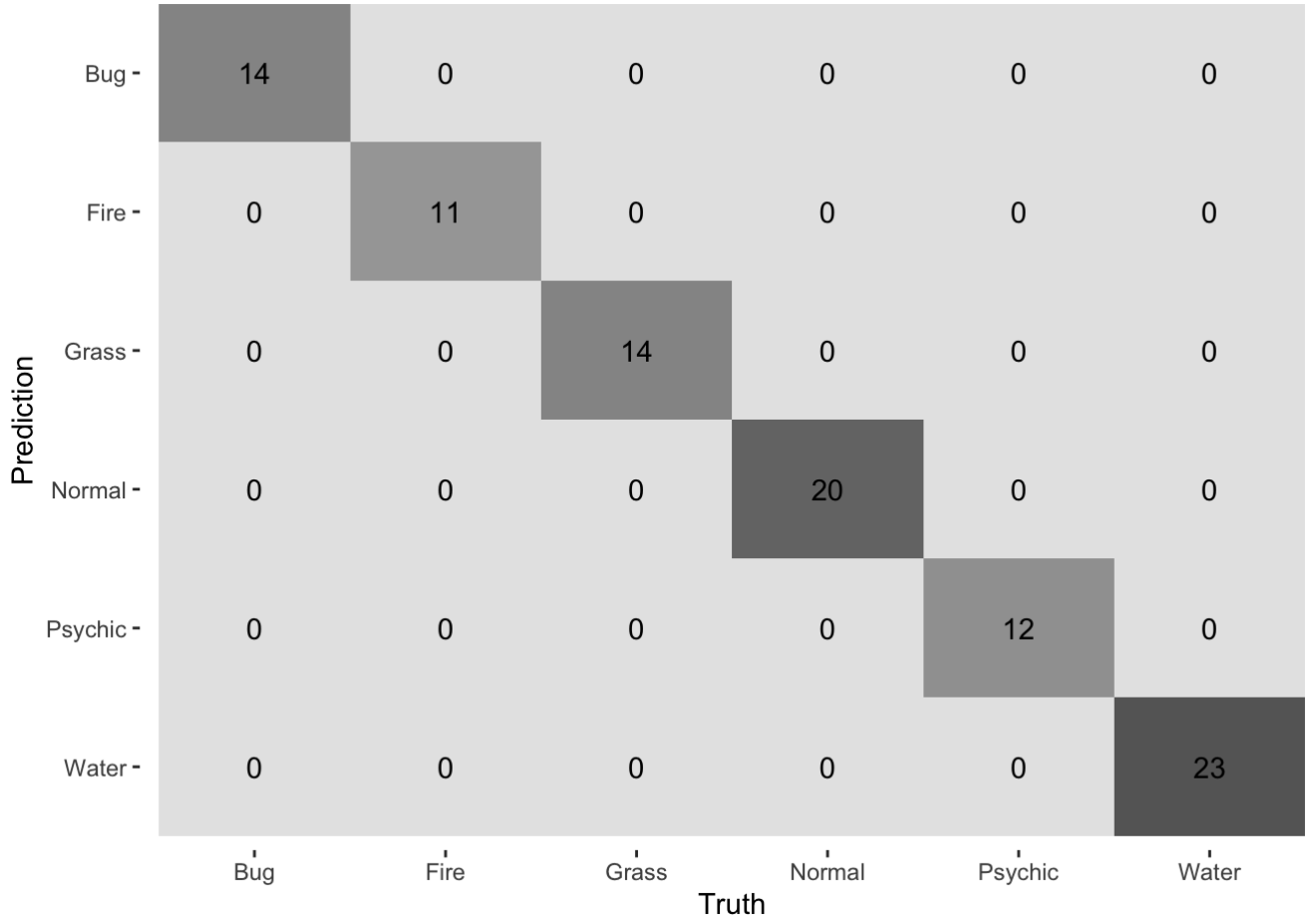
```
augment(class_tree_final_fit, new_data = test) %>%
  conf_mat(truth = type_1, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




```
augment(class_rand_final_fit, new_data = test) %>%
  conf_mat(truth = type_1, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




```
augment(class_boost_final_fit, new_data = test) %>%
  conf_mat(truth = type_1, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```




```
augment(class_tree_final_fit, new_data = test) %>%
  roc_auc(type_1,.pred_Bug:.pred_Water)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.823
```

```
augment(class_rand_final_fit, new_data = test) %>%
  roc_auc(type_1,.pred_Bug:.pred_Water)
```

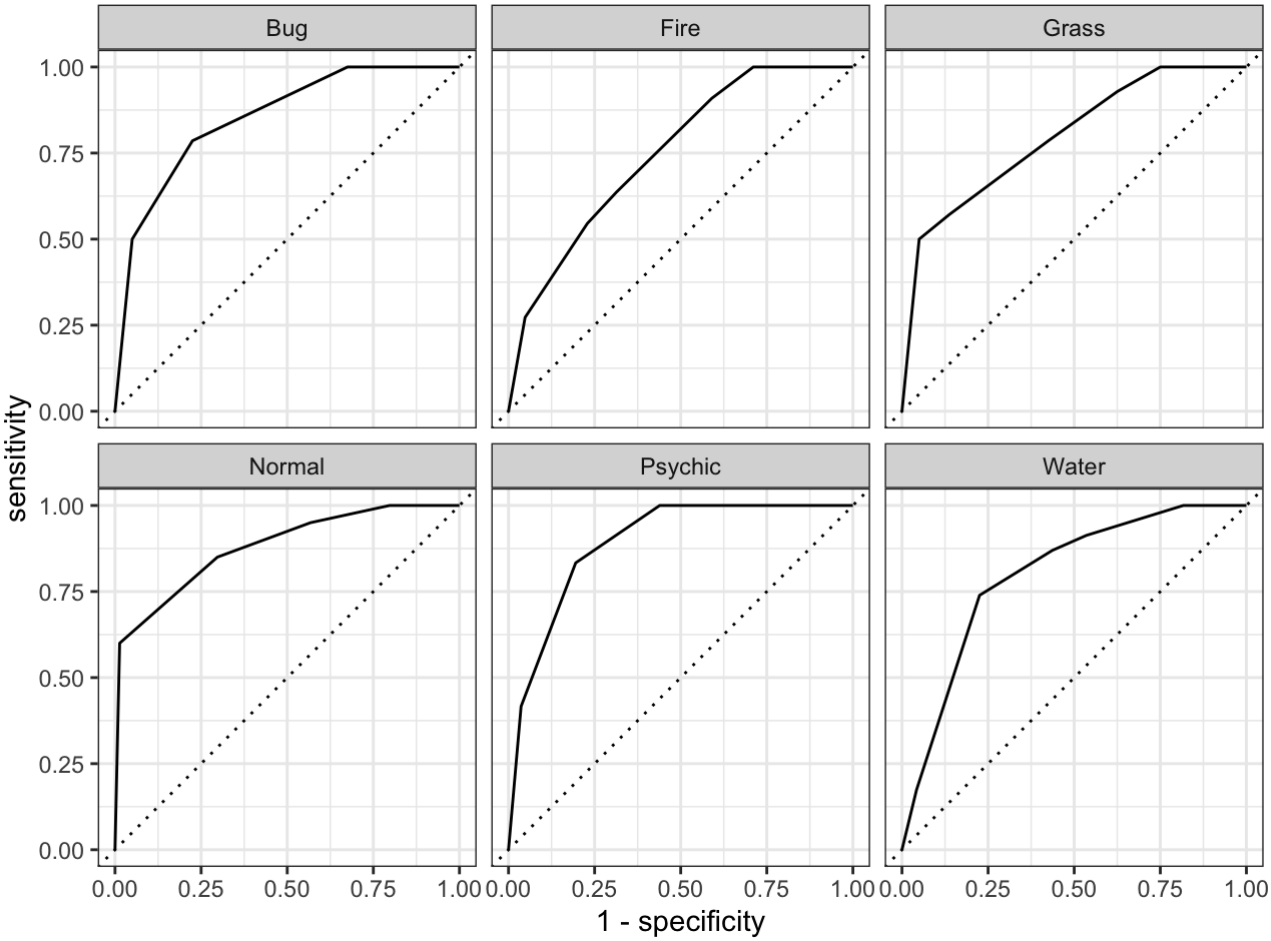
```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.992
```

```
augment(class_boost_final_fit, new_data = test) %>%
  roc_auc(type_1,.pred_Bug:.pred_Water)
```

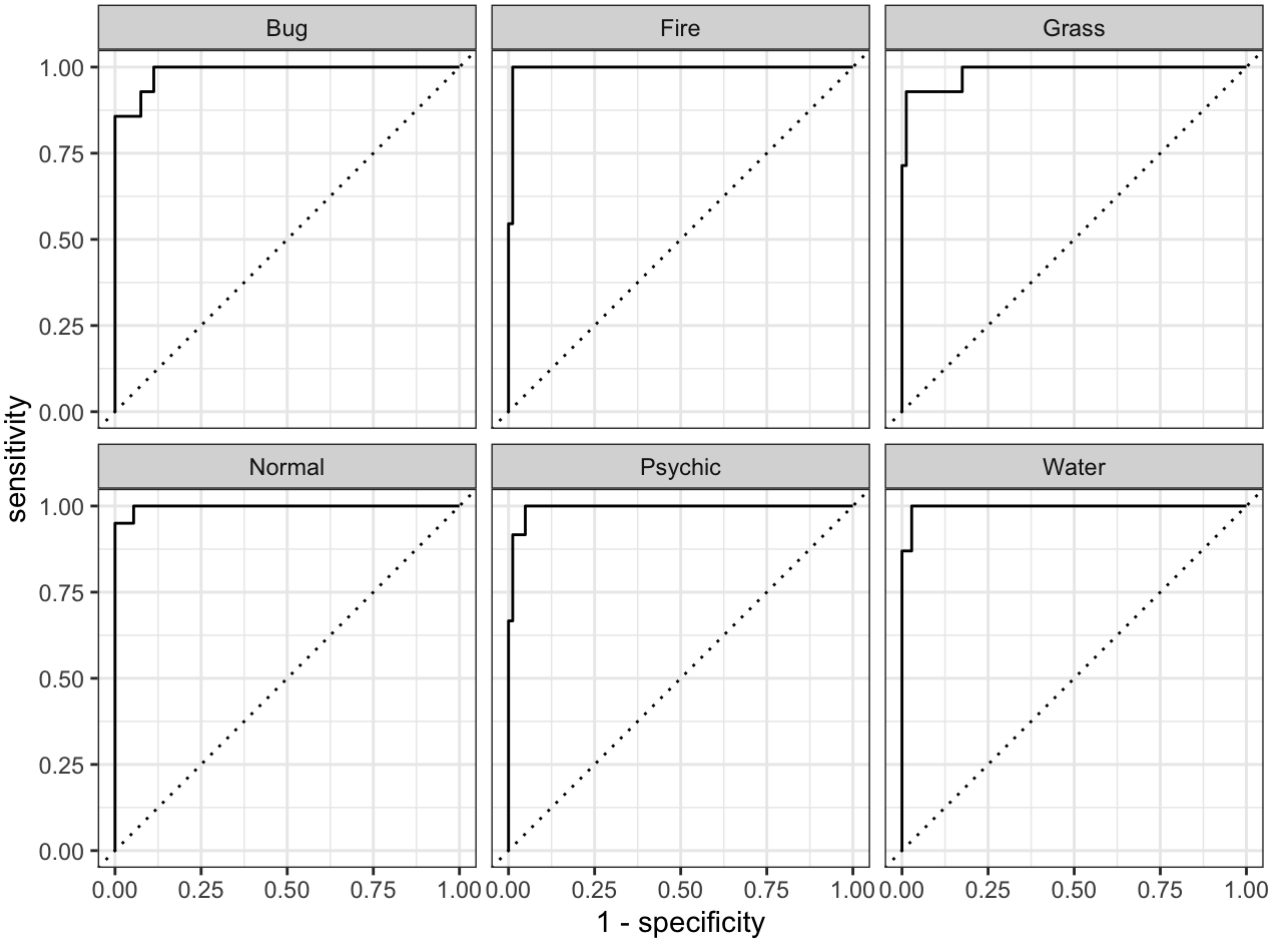
```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till          1
```

These are AUC value and we can see that the boosting model has a 1 meaning that boosting model was able to fix our model really well.

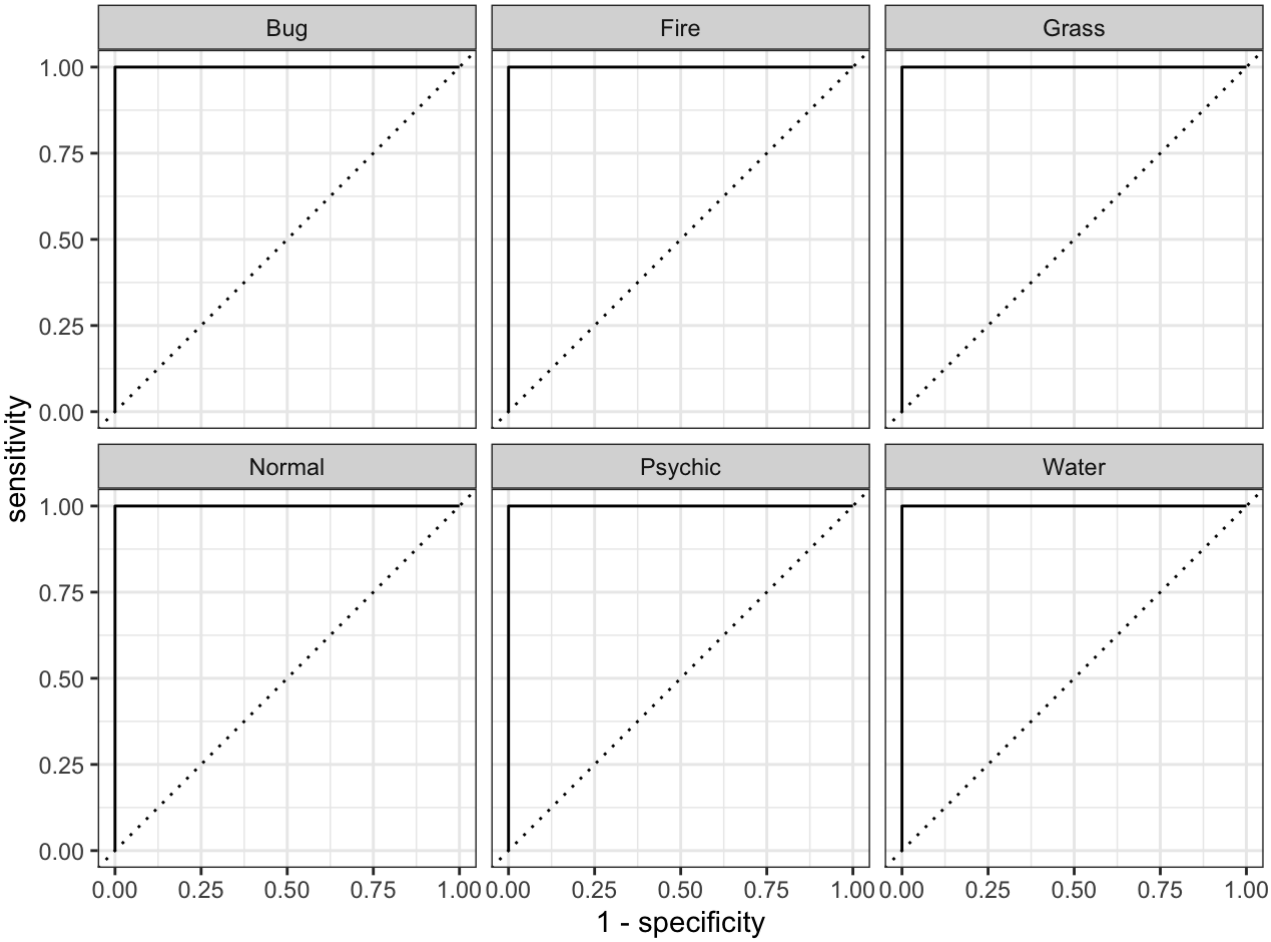
```
augment(class_tree_final_fit, new_data = test) %>%
  roc_curve(type_1,.pred_Bug:.pred_Water) %>%
  autoplot()
```




```
augment(class_rand_final_fit, new_data = test) %>%
  roc_curve(type_1,.pred_Bug:.pred_Water)  %>%
  autoplot()
```




```
augment(class_boost_final_fit, new_data = test) %>%
  roc_curve(type_1,.pred_Bug:.pred_Water) %>%
  autoplot()
```



These are our ROC curve based on the three model best hyperparameters

The model was good at predicting fire, normal, and grass type pokemon and it struggled to predict the classes of bug , psychic , and water types of pokemon based on the confusion matrix values