Project

Markos Meytarjyan

4/16/2022

Introduction

Parkinson's disease is a progressive nervous system disorder that affects movement. Symptoms start gradually, sometimes starting with a barely noticeable tremor in just one hand. Tremors are common, but the disorder also commonly causes stiffness or slowing of movement. One of the symptoms is speech change: you may speak softly, quickly, slur or hesitate before talking. Your speech may be more of a monotone rather than have the usual inflections. I was reading about a research that used significantly larger dataset then mine(more rows and 160 columns) and with certain algorithm the accuracy achieved was 99%. The dataset bellow contains different voice measurements of 31 patients, 23 of which have Parkinson's disease. The voice measurements include maximum, minimum, and average vocal fundamental frequency, several measurements of the variation of vocal fundamental frequency, several measures in variation of amplitude, and a couple of others. I will try to use some of algorithms that we used in the class to see what is the lowest missclassification rate I can achieve.

```
park <- tibble(read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/parkinsons/parkins
```

EDA

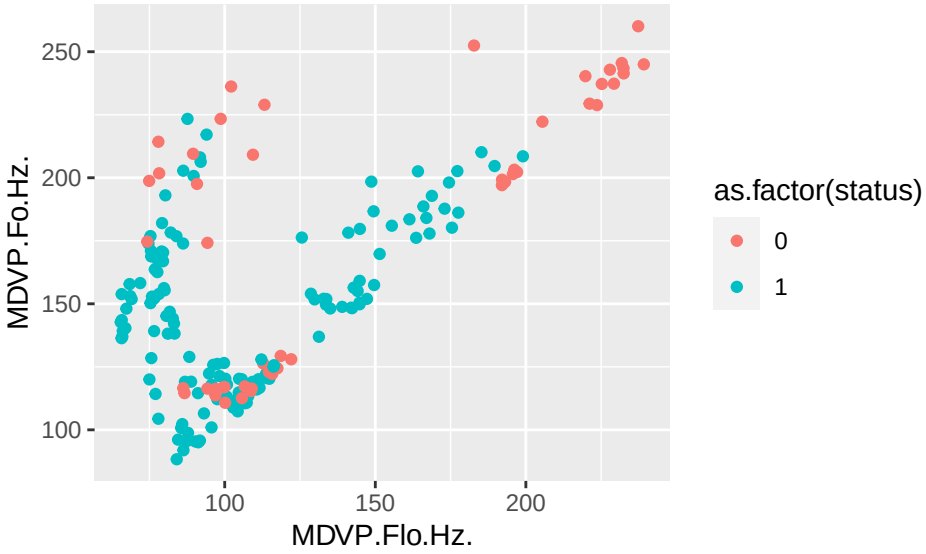
We will start off with some EDA, it would be a bit hard to go into details of different variables as the type of variables is not really known to me so I do not know from which variables what correlation I should expect.

```
park %>%
  count(status)
```

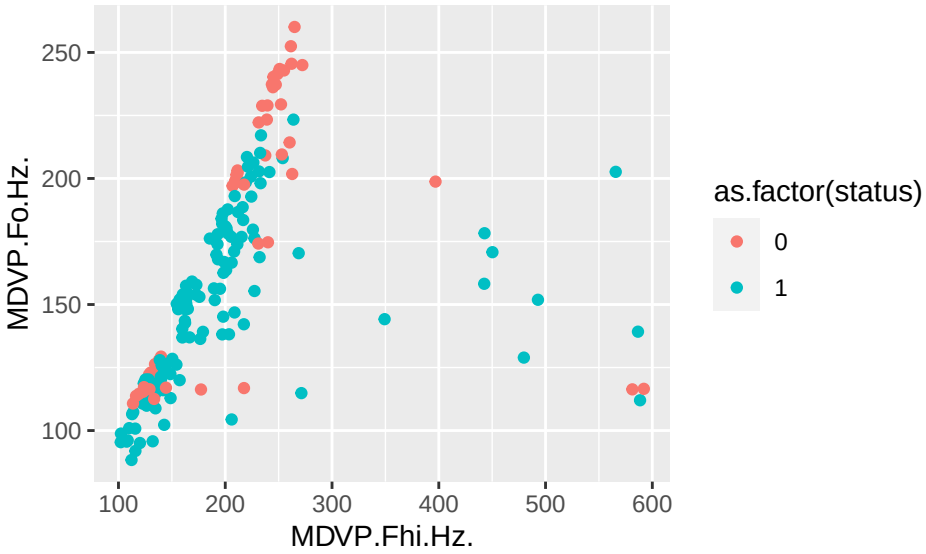
```
## # A tibble: 2 x 2
##   status     n
##    <int> <int>
## 1      0    48
## 2      1   147
```

We can see that 147 voice measurements from the dataset have the Parkinson disease, while 48 do not.

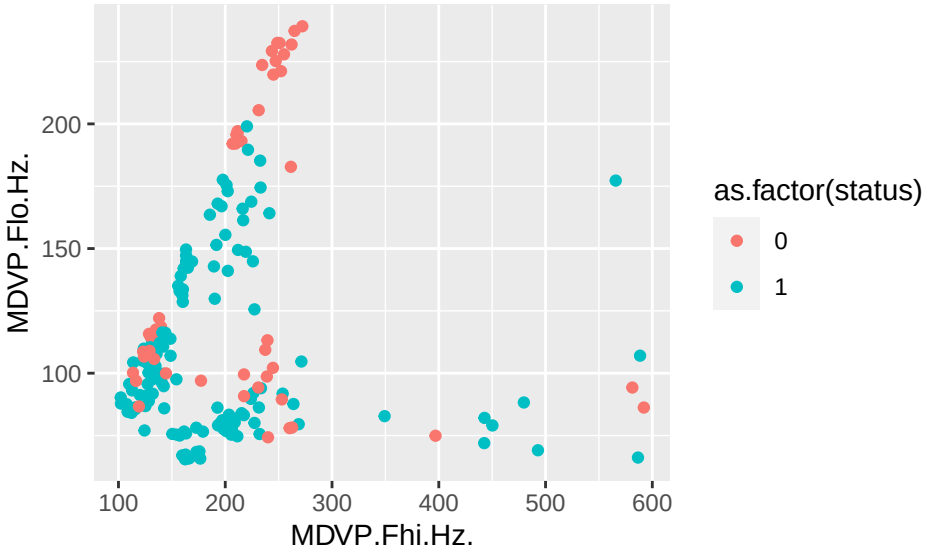
```
ggplot(park) +
  geom_point(aes(x = MDVP.Flo.Hz., y = MDVP.Fo.Hz., color = as.factor(status)))
```




```
ggplot(park) +
  geom_point(aes(x = MDVP.Fhi.Hz., y = MDVP.Fo.Hz., color = as.factor(status)))
```




```
ggplot(park) +
  geom_point(aes(x = MDVP.Fhi.Hz., y = MDVP.Flo.Hz., color = as.factor(status)))
```



We can see that for the minimum maximum and average of vocal fundamental frequency has some linear relationship however there are many outliers from the linear formation which actually do not tell much about weather a person has Parkinson's or not. The vocal fundamental frequency is the lowest frecuency is the lowest frequency in the audiorecording of the voice.

Now we can set a seed and divide the dataset with proportion of 80% to the training and 20% to the testing.

```
set.seed(170802)

park.split <- initial_split(park, prop=0.8)
park.train <- training(park.split) %>%
  select(-name)
park.test <- testing(park.split) %>%
  select(-name)
```

We saw in EDA a couple of linear relationships, so it would be a good idea to try out the regular linear regression and see how it performs.

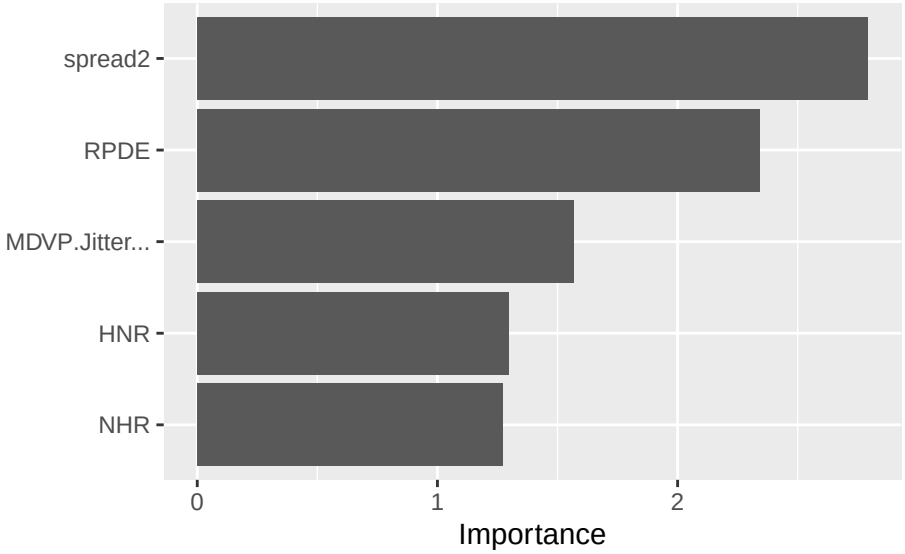
```
lm.model <-
  linear_reg() %>%
  set_mode("regression") %>%
  set_engine("lm")

lm.recipe <- recipe(formula = status ~ ., data = park.train)

lm.wf <- workflow() %>%
  add_recipe(lm.recipe) %>%
  add_model(lm.model)

lm.fit <- fit(lm.wf, park.train)

augment(lm.fit, new_data = park.test) %>%
  mutate(pred = .pred >= 0.5,
         pred = ifelse(pred == TRUE, 1, 0),
         pred = as.factor(pred),
         status = as.factor(status)) %>%
  accuracy(truth = status, estimate = pred)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.897
```

```
augment(lm.fit, new_data = park.test) %>%
  mutate(pred = .pred >= 0.5,
         pred = ifelse(pred == TRUE, "Parkinson", "No Parkinson"),
         pred = as.factor(pred),
         status = ifelse(status == 1, "Parkinson", "No Parkinson"),
         status = as.factor(status)) %>%
  conf_mat(truth = status, estimate = pred)
```

```
##               Truth
## Prediction     No Parkinson Parkinson
##   No Parkinson            9         1
##   Parkinson               3        26
```

```
extract_fit_parsnip(lm.fit) %>%
  vip(num_features = 5)
```



As we can see we achieved 89.7% accuracy which is quite good, there were only 4 missclassifications from 39 observations. We can also see the most important variables for the linear regression. It is very interesting that nonlinear measures of fundamental frequency variation is the most contributing variable to the linear regression. Now we can take a look at how decision tree would perform.

conclusion: accuracy: 89.7% important features: variation of fundamental frequency, ratio of noise to tonal components in the voice, dynamical complexity measure.

```
park.train <- park.train %>%
  mutate(status = ifelse(status == 1, "Parkinson", "No Parkinson"),
         status = as.factor(status))

park.test <- park.test %>%
  mutate(status = ifelse(status == 1, "Parkinson", "No Parkinson"),
         status = as.factor(status))

park.model <-decision_tree(tree_depth=tune(), cost_complexity=tune()) %>%
```

```
  set_mode("classification") %>%
  set_engine("rpart")

park.recipe <- recipe(status ~ ., data=park.train)

park.wf <- workflow() %>%
  add_recipe(park.recipe) %>%
  add_model(park.model)

park.folds <- vfold_cv(park.train, v = 10)

park.grid <-
  grid_regular(cost_complexity(), tree_depth(), levels = 4)

park.res <- tune_grid(
    park.wf,
    resamples = park.folds,
    grid = park.grid,
    metrics = metric_set(accuracy, roc_auc, sensitivity, specificity))

best.penalty <- select_by_one_std_err(x = park.res, metric = "accuracy", (cost_complexity))
park.final.wf <- finalize_workflow(park.wf, best.penalty)
park.final.fit <- fit(park.final.wf, data = park.train)

augment(park.final.fit, park.test) %>%
  accuracy(truth = status, estimate = .pred_class)
```

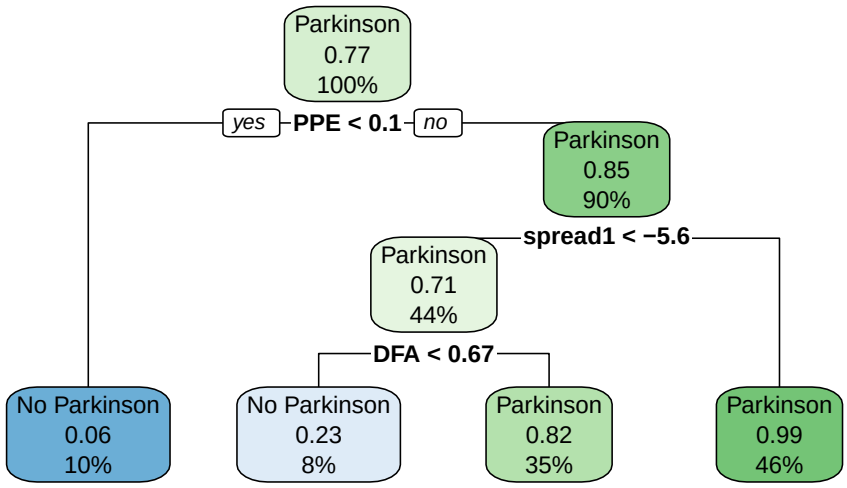
```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.821
```

```
augment(park.final.fit, park.test) %>%
  conf_mat(truth = status, estimate = .pred_class)
```

```
##               Truth
## Prediction     No Parkinson Parkinson
##   No Parkinson            8         3
##   Parkinson               4        24
```

```
park.final.fit %>%
  extract_fit_engine() %>%
  rpart.plot()
```

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



As we can see decision tree performs worse then the linear regression.

conclusion: accuracy: 82.1% important variables: fundamental frequency variation, Signal fractal scaling exponent best parameters: cost complexity = 0.0000000001, tree depth = 5

KNN sometimes performs really well so I decided to try it out on this dataset as well. Setting up a tune it turned out that the optimal number neighbors is 6 and the accuracy associated with it is 100% which is a perfect result. There is no way with the KNN algorithm to determine the important variables so for the final conclusion we will have to look at the important variables of other algorithms.

```
park.train <- park.train %>%
  mutate(status = as.factor(status))

park.test <- park.test %>%
  mutate(status = as.factor(status))

knn.model <- nearest_neighbor(neighbors = tune()) %>%
    set_engine("kknn") %>%
    set_mode("classification")

knn.recipe <- recipe(status ~ ., data = park.train)

knn.wf <- workflow() %>%
    add_recipe(knn.recipe) %>%
    add_model(knn.model)

park.train.fold <- vfold_cv(park.train, v = 10)

neighbors.grid.tbl <- tibble(neighbors = seq(1,30, by=1))

tune.results <- tune_grid(object = knn.wf,
                          resamples = park.train.fold,
                          grid = neighbors.grid.tbl)

best.penalty <- select_best(tune.results, metric = "roc_auc")
knn.final.wf <- finalize_workflow(knn.wf, best.penalty)
knn.final.fit <- fit(knn.final.wf, data = park.train)
```

```
augment(knn.final.fit, park.test) %>%
  accuracy(truth = status, estimate = .pred_class)
```

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

```
augment(knn.final.fit, park.test) %>%
  conf_mat(truth = status, estimate = .pred_class)
```

```
##               Truth
## Prediction     No Parkinson Parkinson
##   No Parkinson           12         0
##   Parkinson               0        27
```

conclusion: accuracy: 100% best parameter: k = 6

Another thing that I decided to do is a random forest, which generally performs well with several variables, for that I tuned the mtry variable. The most important variables were fundamental frequency variation, average vocal fundamental frequency and highest vocal fundamental frequency. The optimal value of mtry is 11.

```
park.folds <- vfold_cv(park.train, v = 10)

model.park <- rand_forest(trees = 500, mtry = tune(), min_n = 1) %>%
    set_mode("classification") %>%
    set_engine("ranger",importance = "impurity")

recipe.park <- recipe(status ~ ., park.train)

wf.park <- workflow() %>%
    add_recipe(recipe.park) %>%
    add_model(model.park)


grid.park <- expand.grid(mtry=c(1:22))

res.park <- tune_grid(wf.park,
                      resamples = park.folds,
                      grid = grid.park)

show_best(res.park, metric = "accuracy")
```

```
## # A tibble: 5 x 7
##    mtry .metric  .estimator  mean     n std_err .config
##   <int> <chr>    <chr>      <dbl> <int>   <dbl> <fct>
## 1     4 accuracy binary     0.922    10  0.0214 Preprocessor1_Model04
## 2     2 accuracy binary     0.917    10  0.0272 Preprocessor1_Model02
## 3     7 accuracy binary     0.916    10  0.0237 Preprocessor1_Model07
## 4    10 accuracy binary     0.916    10  0.0237 Preprocessor1_Model10
## 5     3 accuracy binary     0.910    10  0.0279 Preprocessor1_Model03
```

```
best.penalty <- select_best(res.park,
                            metric = "accuracy")
```

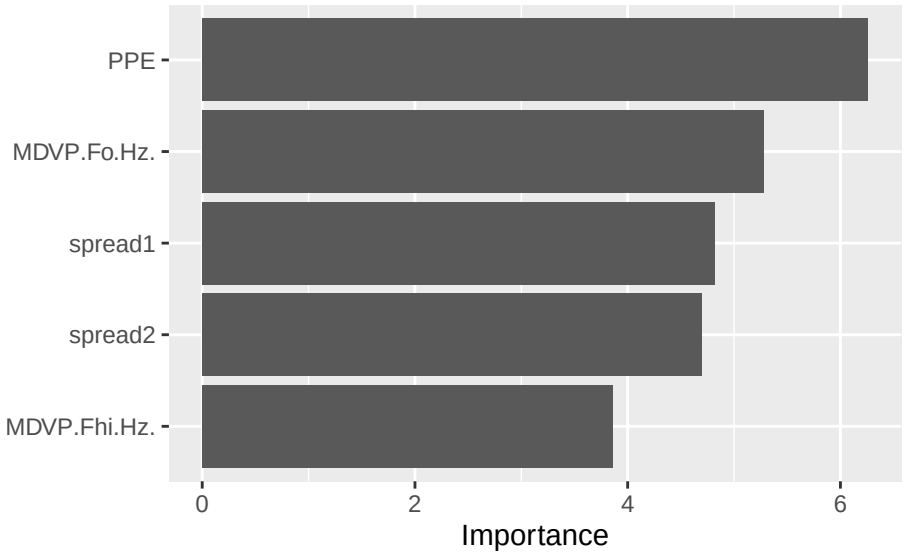
```
park.final.wflow <- finalize_workflow(wf.park,
                                      best.penalty)

park.final.fit <- fit(park.final.wflow,  park.train)

augment(park.final.fit, park.test) %>%
  accuracy(truth = status, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.923
```

```
augment(park.final.fit, park.test) %>%
  conf_mat(truth = status, estimate = .pred_class)
```

```
##                Truth
## Prediction      No Parkinson Parkinson
##   No Parkinson           10         1
##   Parkinson               2        26
```

```
extract_fit_parsnip(park.final.fit) %>%
  vip(num_features = 5)
```



conclusion: accuracy: 89.7% best parameter: mtry = 11 important variables: fundamental frequency variation, average vocal fundamental frequency, maximum vocal fundamental frequency

In the end I applied boosting to the Parkinson's disease dataset to see the performance of it. The most important variables were . The optimal value of learning rate was

```
park.grid <- grid_regular(trees(range=c(100,1000)), learn_rate(range=c(-3,0)), levels = 5)

park.folds <- vfold_cv(park.train, v = 10)

xgboost_recipe <-
  recipe(formula = status ~ ., data = park.train) %>%
  step_zv(all_predictors())
```

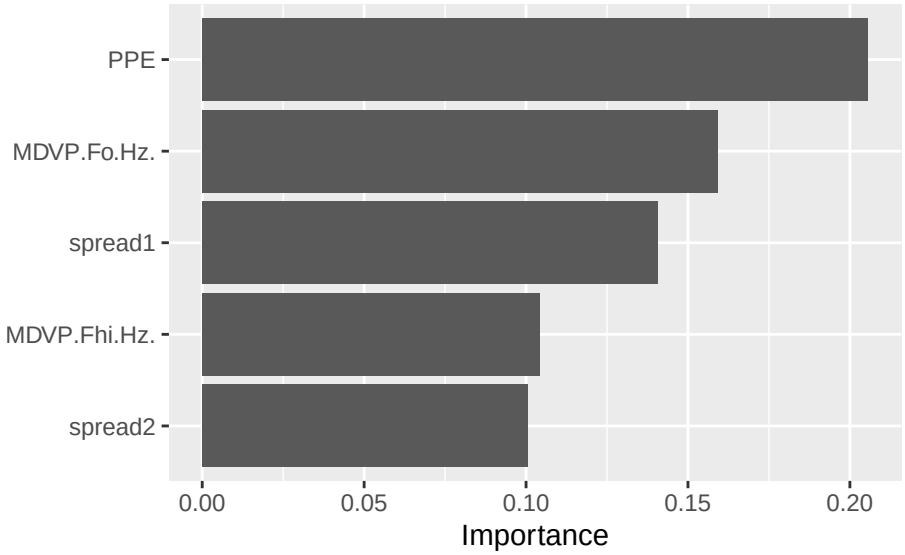
```
xgboost_spec <-
  boost_tree(trees = tune(), learn_rate = tune()) %>%
  set_mode("classification") %>%
  set_engine("xgboost")

xgboost_workflow <-
  workflow() %>%
  add_recipe(xgboost_recipe) %>%
  add_model(xgboost_spec)

xgboost_tune <-
  tune_grid(xgboost_workflow,
            resamples = park.folds,
            grid = park.grid)

best.param <- select_best(xgboost_tune, "accuracy")
park.boost.wf <- finalize_workflow(xgboost_workflow, best.param)
park.boost.model <- fit(park.boost.wf, park.train)

augment(park.boost.model, park.test) %>%
  accuracy(truth = status, estimate = .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy binary         0.897
```

```
augment(park.boost.model, park.test) %>%
  conf_mat(truth = status, estimate = .pred_class)
```

```
##                Truth
## Prediction      No Parkinson Parkinson
##   No Parkinson             9         1
##   Parkinson                3        26
```

```
extract_fit_parsnip(park.boost.model) %>%
  vip(num_features = 5)
```



conclusion: accuracy: 92.3% best parameter: trees = 325, learn rate = 0.178 important variables: fundamental frequency variation, average vocal fundamental frequency, maximum vocal fundamental frequency

Conclusion

Overall I think most of the models performed really well. The best one turned out to be KNN which had accuracy of 100%. I do understand that for a larger dataset the accuracy would drop, the dataset that I used had only 39 testing observations. We can also notice that for all of the models the most important variable was the variation of fundamental frequency. Other variables that were important were the average and highest fundamental frequency, meaning that fundamental frequency is the most important measure in determining weather person has Parkinson's disease. As a reminder fundamental frequency is the lowest frequency in the audio recording of the pearson speaking.

Model	**Accuracy**	**Important variables**
Linear regression	89.7%	fundamental frequency variation
Decision tree	82.1%	fundamental frequency variation
KNN	100%	n/a
Random forest	89.7%	fundamental frequency variation
Boosting	92.3%	fundamental frequency variation