Predicting beneficiary households of the national health assistance scheme in villages across Indonesia

Alfian Bin Aman & Hamidah Alatas

December 23, 2021

1 Introduction

This project aims to predict the percentage of households in villages (*desa*) across Indonesia which receive health insurance under a national social assistance scheme. This scheme provides free health coverage to low-income households in Indonesia.

As our outcome variable is continuous (percentage), we use the following supervised learning techniques, which include a linear regression model, a generalized linear model, random forests, K-nearest neighbors, and Generalized Additive Models (GAMs) to predict our outcome variable. We then compare the results of these models to identify which model produces the lowest root mean-squared error (rmse) in the testing data, and thus is the most suitable predictor of our outcome variable.

2 Data and Variables

This project uses village-level data from the 2006 Village Potential Survey (PODES) conducted by Statistics Indonesia in conjunction with the National Economic Census. The data was sourced from Columbia's Research Data Services.

The data contains useful information on the characteristics of each village such as the total population size, the state of housing and infrastructure (e.g. presence of slums, number of educational, healthcare and social institutions), socioeconomic indicators (e.g. whether village has households living below the national poverty line, year-on-year trends in crime rate), and other economic indicators (e.g. number of medium and large industries, presence of informal micro finance institutions).

In summary, the PODES data has a total of 69,957 observations and 488 variables with no major issues pertaining to missing data. However, for the purpose of our analysis, we will only use 41 variables which comprise numeric and factor variables. Prior to analyzing the data, we first clean the data which includes dropping unused variables, renaming the remaining variables for an easier read of the predictor variables used in our analysis, and creating new variables derived from some of our existing variables.

```
# List vector of variables to keep
selected_var = c("r605", "r401c", "r401a","r401b", "r510ak2","r510bk2",
                 "r510ck2", "r510dk2", "r601ak2", "r601ak3", "r601bk2",
                 "r601bk3", "r601ck2", "r601ck3", "r601dk2", "r601dk3",
                 "r601ek2", "r601ek3", "r601fk2", "r601fk3", "r601gk2",
```

```
                "r601gk3", "r603ak2", "r603bk2", "r603ck2", "r603dk2",
                "r603ek2", "r603fk2", "r603gk2", "r603hk2", "r603ik2",
                "r603jk2", "r604a1", "r604a2",  "r606", "r607ak2", "r607bk2",
                "r607ck2", "r607dk2",  "r607ek2", "r704a1k2", "r704a2k2",
                "r704a3k2", "r704a4k2",  "r704a5k2", "r704a6k2", "r901b2",
                "r1107", "r1106", "r1204a1k3", "r1204a2k3", "r1204a3k3",
                "r1204a4k3", "r1204a5k3", "r1204a6k3", "r1204a7k3",
                "r1204a8k3", "r1204a9k3", "r1204b", "r1204a10k3")

# Load the raw data
podes_raw <- read_dta("podes06_merged.dta")

# Initially transform the selected variables into numeric data
podes_raw[,selected_var] <- lapply(podes_raw[,selected_var], as.numeric)

# Generate predictor variables from the data
podes_coded <- podes_raw %>%
  transmute(province = prop,
         urban = r105a,
         village_rep = r302,
         near_sea = r304a,
         near_forest = r305,
         population = r401a+r401b,
         source_income = r402,
         hh_electricity = r501a,
         street_lighting = r502a,
         waste_disposal = r504,
         sanitation = r505,
         river_bank = r506a,
         luxury_res = r509a,
         slums = r509b,
         pollution = ifelse((r510ak2 == 1 | r510bk2 == 1 | r510ck2 == 1 |
                             r510dk2 == 1), 1, 0),
         mining = r511,
         disaster_prone = r512,
         num_k12 = r601ak2 + r601ak3 + r601bk2 + r601bk3 +
           r601ck2 + r601ck3 + r601dk2 + r601dk3 + r601ek2 + r601ek3,
         num_higher_ed = r601fk2 + r601fk3,
         num_health_facil = r603ak2 + r603bk2 + r603ck2 + r603dk2 +
           r603ek2 + r603fk2 + r603gk2 + r603hk2 + r603ik2,
         num_doctors = r604a1 + r604a2,
         poor = ifelse(r606 > 0 , 1, 0),
         pandemic = ifelse((r607ak2 == 1 | r607bk2 == 1 | r607ck2 == 1 |
                            r607dk2 == 1 | r607ek2 == 1), 1, 0),
         water_source = r608a,
         social_institution = ifelse((r704a1k2 == 1 | r704a2k2 == 1 |
                                      r704a3k2 == 1 | r704a4k2 == 1 |
```

```
                                    r704a5k2 == 1 | r704a6k2 == 1), 1, 0),
         disabled = ifelse((r705ak2 == 1 | r705bk2 == 1 | r705ck2 == 1 |
                            r705dk2 == 1 | r705ek2 == 1), 1, 0),
         type_road = ifelse(is.na(r901b1), 0, r901b1),
         trafficability = ifelse(is.na(r901b2), 2, r901b2),
         dist_capital = r902ak21,
         num_hh_cable = r904,
         signal_strength = r911,
         land_area = r10011,
         industrial_area = r1103,
         num_med_large_industry = r1106 + r1107,
         num_commercial_bank = r1119,
         num_village_bank = r1120a,
         micro_fin = r1123,
         credit_facil = ifelse((r1124a == 1 | r1124b == 3 | r1124c == 5 |
                                r1124d == 7), 1, 0),
         most_common_crime = ifelse(is.na(r1204b), 0, r1204b),
         crime_rate_increase = ifelse((r1204a1k3 == 3 |
                                       r1204a2k3 == 3 |
                                       r1204a3k3 == 3 |
                                       r1204a4k3 == 3 |
                                       r1204a5k3 == 3 |
                                       r1204a6k3 == 3 |
                                       r1204a7k3 == 3 |
                                       r1204a8k3 == 3 |
                                       r1204a9k3 == 3 |
                                       r1204a10k3 == 3), 1, 0),
         police_station = r1207bk2,
         perc_hh_sa = r605/r401c*100) %>%
  mutate(crime_rate_increase = ifelse(is.na(crime_rate_increase), 0, 1))

# List vector of numeric variables
num_var <- c("perc_hh_sa", "population", "num_k12", "num_higher_ed",
             "num_health_facil", "num_doctors", "dist_capital",
             "num_hh_cable", "land_area", "num_med_large_industry",
             "num_commercial_bank", "num_village_bank")

# List vector of factor variables
factor_var <- c("province", "urban", "village_rep", "near_sea", "near_forest",
            "source_income", "hh_electricity", "street_lighting",
            "waste_disposal", "sanitation", "river_bank", "luxury_res",
            "slums", "pollution", "mining", "disaster_prone",
            "poor", "pandemic", "water_source", "social_institution",
            "disabled", "type_road", "trafficability", "signal_strength",
            "industrial_area", "micro_fin", "credit_facil",
            "most_common_crime","crime_rate_increase", "police_station")
```

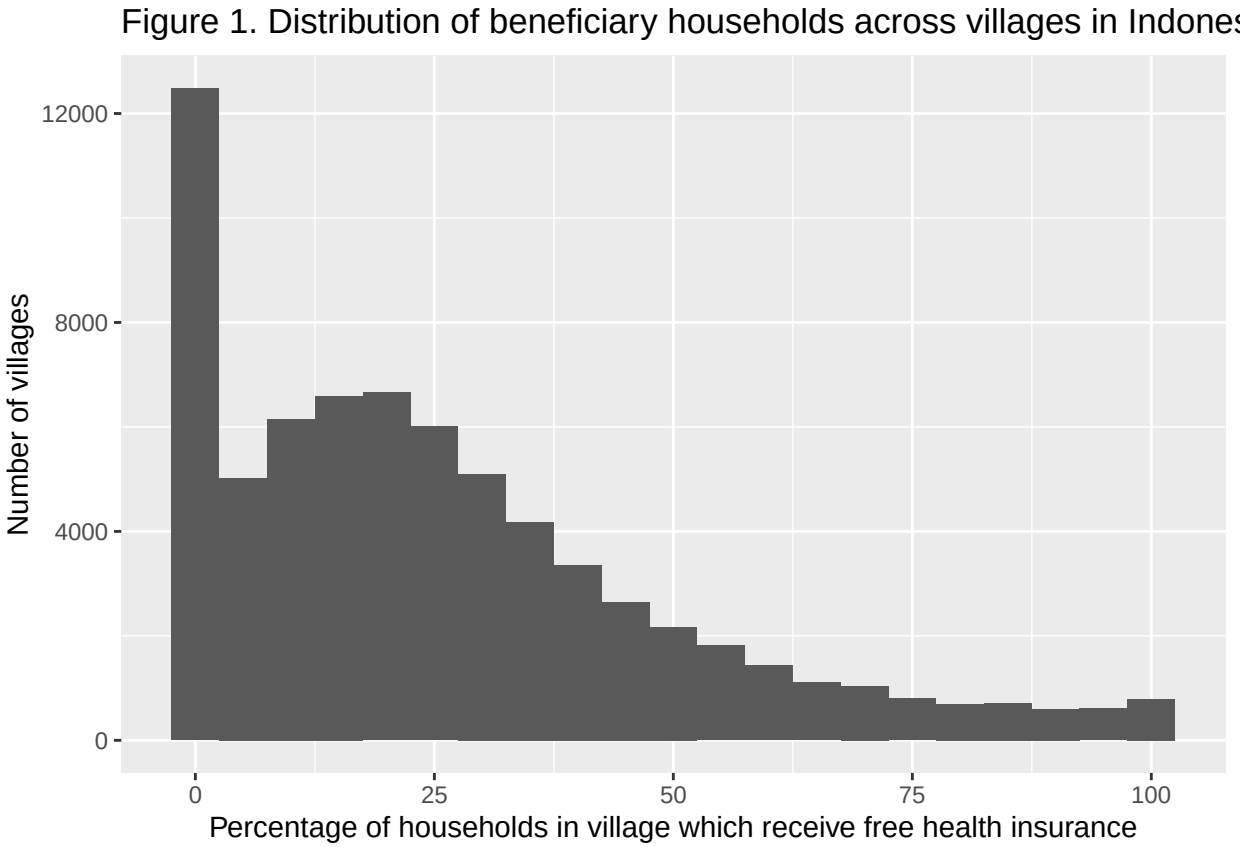
```
# Finally, transform the above variables into numeric and factor data accordingly.
podes_coded[,num_var] <- lapply(podes_coded[,num_var], as.numeric)
podes_coded[,factor_var] <- lapply(podes_coded[,factor_var], factor)

saveRDS(podes_coded, "podes_coded.RDS")
```

2.1. Outcome Variable

In this analysis, we use `perc_hh_sa` as our outcome variable which indicates the percentage of households per village which receive free health insurance under the national social assistance scheme. This variable is derived by calculating the "number of households which receive free health insurance under the scheme in the past year" as a percentage of the "total number of households in the village". The range of values of this variable is 0 to 100, with 100 indicating that every household in the village has received free health insurance under the scheme in the past year, which is a possible occurrence in particularly small and poor villages. As shown in Figure 1, approximately 12,500 villages do not have any beneficiary households under the scheme. However, among villages which have beneficiary households, a parabolic trend can be observed with most villages having beneficiary households comprise approximately 20% of their population, and increasingly less villages as the percentage of beneficiary households per village changes in either direction.

```
ggplot() +
  geom_histogram(data = podes_coded, aes(perc_hh_sa), binwidth = 5) +
  ylab("Number of villages") +
  xlab("Percentage of households in village which receive free health insurance") +
  ggtitle("Figure 1. Distribution of beneficiary households across villages in Indonesia")
```



2.2. Predictor Variables

In this section, we will briefly describe the 41 variables we have selected as predictors to estimate our outcome variable as discussed above. In narrowing down our list of predictor variables, we had selected variables which we believed were relevant indicators of the state of infrastructure and poverty in a village, and which would provide a reasonable measure of the likelihood (and thus, percentage) of households in each village qualifying for, and receiving, free health insurance under the national social assistance scheme. The list of predictor variables used in our analysis is shown below.

No.	Variable	Description	Type
1	province	Province code	Categorical
2	urban	Urbanicity of village	Categorical
3	village_rep	Village has a representative body	Categorical
4	near_sea	Proximity to the sea	Categorical
5	near_forest	Proximity to a forested area	Categorical
6	population	No. of villagers	Continuous
7	source_income	Primary source of income (e.g. agriculture, manufacturing)	Categorical
8	hh_electricity	Has houses with electricity	Categorical
9	street_lighting	Has street lighting	Categorical
10	waste_disposal	Primary waste disposal method	Categorical
11	sanitation	Primary sanitation method	Categorical

No.	Variable	Description	Type
12	river_bank	Near river bank	Categorical
13	luxury_res	Has luxury residences	Categorical
14	slums	Has slums	Categorical
15	pollution	Presence of environmental pollution in the past year	Categorical
16	mining	Presence of mining activities	Categorical
17	disaster_prone	Prone to natural disasters	Categorical
18	num_k12	No. of K-12 institutions	Continuous
19	num_hs	No. of higher education institutions	Continuous
20	num_health_facil	No. of hospitals and other healthcare facilities	Continuous
21	num_doctors	No. of doctors	Continuous
22	poor_letter	Has households below the national poverty line in the past year	Categorical
23	pandemic	Presence of a pandemic in the past year	Categorical
24	water_source	Primary water source	Categorical
25	social_institutions	Presence of social institutions	Categorical
26	disabled	Presence of people with disabilities	Categorical
27	type_road	Type of road surface	Categorical
28	trafficability	Vehicle trafficability	Categorical
29	dist_capital	Distance to capital district	Continuous
30	signal_Strength	Handphone signal strength	Categorical
31	num_hh_cable	No. of households with cable subscriptions	Continuous
32	land_area	Land area	Continuous
33	industrial_area	Presence of industrial area	Categorical
34	num_med_large_industry	No. of medium/large industries	Continuous
35	num_commercial_banks	No. of commercial banks	Continuous
36	num_village_banks	No. of village banks	Continuous
37	num_micro_finance	No. of informal micro finance institutions	Continuous
38	num_credit_facil	No. of credit facilities	Continuous
39	most_common_crime	Most common type of crime	Categorical
40	crime_rate_increase	Rise in crime rate from past year	Categorical
41	police_station	Presence of police station	Categorical

3 Data Mining

For our analysis, we first split our data into training data, which will be used to solve varying optimization problems depending on the regression model, and testing data, which are additional observations which will be used to evaluate the accuracy of the results produced by each model. Then, a series of supervised learning techniques from the `tidymodels` package are separately fit to the training data, and the trained model will then be used to predict our outcome variable using the attained solution to the respective optimization problem. Finally, we will evaluate the accuracy of the predictions in the testing data by comparing the rmse across all our models, with the model producing the lowest rmse being the most accurate.

```
set.seed(2112)
podes <- readRDS("podes_coded.RDS")
```

```
podes_split0 <- initial_split(podes, prob = 0.50, strata = province)
podes_sample <- training(podes_split0)

podes_split1 <- initial_split(podes_sample, prob = 0.70, strata = province)
podes_train <- training(podes_split1)
podes_test  <- testing(podes_split1)
```

```
podes_recipe <-
  recipe(perc_hh_sa ~ ., data = podes_train) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_center(all_numeric_predictors()) %>%
  step_scale(all_numeric_predictors()) %>%
  prep()
```

3.1. Linear Regression

Our first model uses an OLS (Ordinary Least Squares) linear regression model which minimizes the sum-of-squared residuals, and a base recipe in order to predict our outcome variable as shown below.

```
lm_model <-
  linear_reg() %>%
  set_engine("lm")

lm_workflow <-
  workflow() %>%
  add_model(lm_model) %>%
  add_recipe(podes_recipe)

lm_fit <- fit(lm_workflow, data = podes_train)

bind_cols(podes_test,
          predict(lm_fit, new_data = podes_test)) %>%
  rmse(truth = podes_test$perc_hh_sa, estimate = .pred)
```

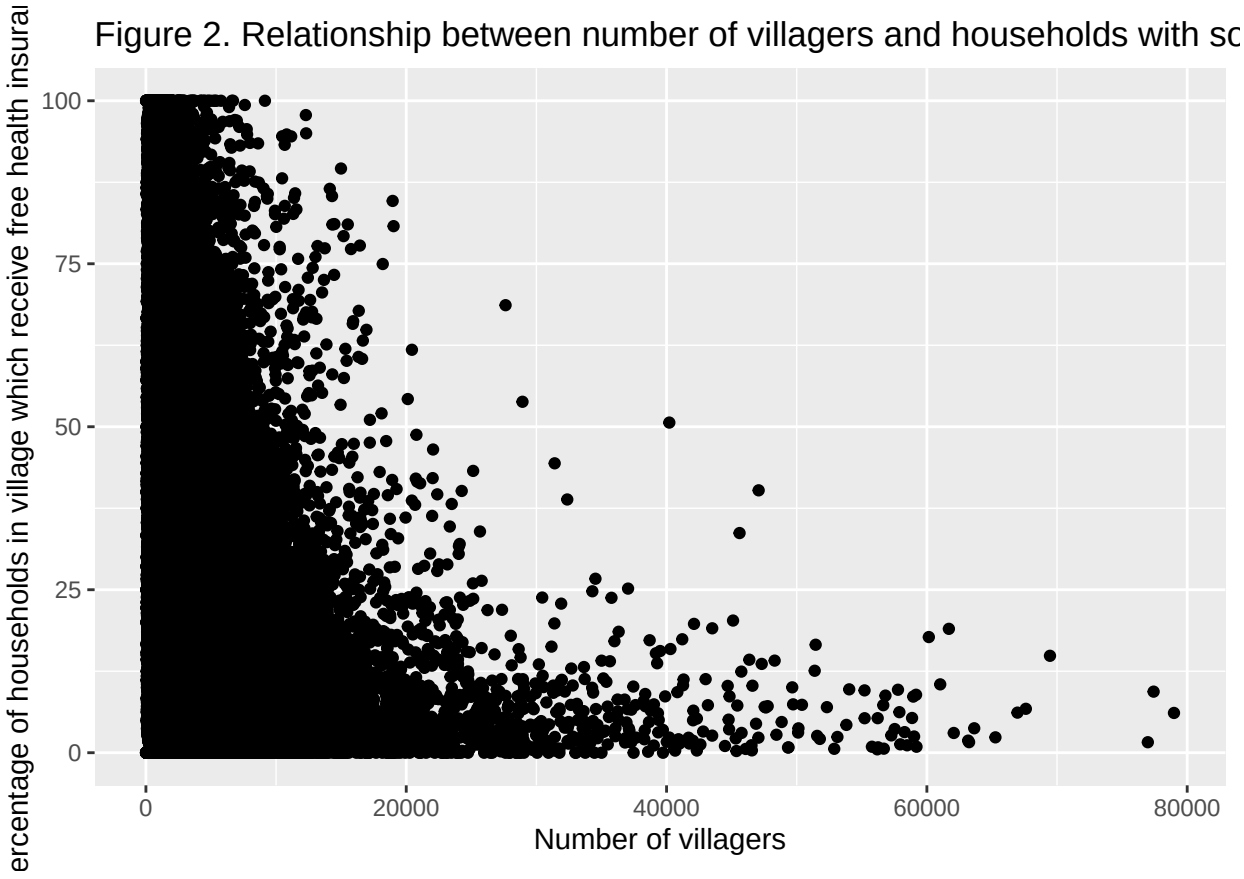
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        21.8
```

3.2. Regression with polynomial

However, the above model assumes a strictly linear relationship between our outcome and predictor variables. Our next model takes into account that there is a non-linear relationship between our outcome variable and the population size of the village, as shown in Figure 2 below. As such, `step_poly` is added to the base recipe to tune and estimate the approximate degree of polynomial for this non-linear relationship using bootstrapping.

```
ggplot() +
  geom_point(data = podes_coded, aes(y = perc_hh_sa,
                                     x = population), binwidth = 5) +
  ylab("Percentage of households in village which receive free health insurance") +
  xlab("Number of villagers") +
  ggtitle("Figure 2. Relationship between number of villagers and households with social assista
```

```
## Warning: Ignoring unknown parameters: binwidth
```




```
lm_model <-
  linear_reg() %>%
  set_engine("lm")

poly_recipe <-
  podes_recipe %>%
  step_poly(population, degree = tune())

poly_bs <- bootstraps(podes_train, times = 10)
poly_grid <- tibble(degree = 1:5)

poly_wf <-
```

```
  workflow() %>%
  add_model(lm_model) %>%
  add_recipe(poly_recipe)

results <- tune_grid(poly_wf, resamples = poly_bs, grid = poly_grid)
```

```
(best <- select_best(results, "rmse"))
```

```
## # A tibble: 1 x 2
##   degree .config
##    <int> <chr>
## 1      5 Preprocessor5_Model1
```

```
poly_recipe <-
  podes_recipe %>%
  step_poly(population, degree = 5)

linear_wf <-
  workflow() %>%
  add_model(lm_model) %>%
  add_recipe(poly_recipe)

linear_fit <- fit(linear_wf, podes_train)

bind_cols(podes_test,
          predict(linear_fit, new_data = podes_test)) %>%
  rmse(truth = perc_hh_sa, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        21.6
```

3.3. Elastic Net

The following models will utilize non-linear mappings to predict our outcome variable and assess if a non-linear or biased estimator will yield better predictions in our testing data. Our next model in particular fits a generalized linear model to the training data via a penalized maximum likelihood approach. This approach involves generalizing a Ridge regression and Lasso regression to an objective function called 'elastic net' which minimizes the sum-of-squared residuals subject to a penalty function λ.

```
bake_train <- bake(podes_recipe, new_data = NULL)
bake_test <- bake(podes_recipe, new_data = podes_test)

podes_rs <- bootstraps(bake_train, times = 10)
```

```
glmnet_model <-
  linear_reg(penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet")

glmnet_wf <-
  workflow() %>%
  add_model(glmnet_model) %>%
  add_recipe(recipe(perc_hh_sa ~ ., data = bake_train))

glmnet_grid <- grid_regular(parameters(glmnet_model), levels = 10)

glmnet_results <- tune_grid(glmnet_wf, resamples = podes_rs, grid = glmnet_grid)
```

```
best <-
  glmnet_results %>%
  select_best("rmse")

final_wf <- finalize_workflow(glmnet_wf, best)

tuned_fit <- fit(final_wf, data = bake_train)

bind_cols(bake_test,
          predict(tuned_fit, new_data = bake_test)) %>%
  rmse(truth = bake_test$perc_hh_sa, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        21.8
```

3.4. Generalized Additive Models

This model allows us to automatically model non-linear relationships between the predictor variables and the outcome variable in our data by replacing each linear component with a smooth non-linear function, while maintaining additivity. Due to the nature of GAMs, it provides a useful compromise between linear and non-parametric models.

```
library(additive)

GAM <-
  additive() %>%
  set_engine("mgcv") %>%
  set_mode("regression")

GAM_wf <-
```

```
  workflow() %>%
  add_model(GAM, formula = perc_hh_sa ~ s(population, land_area)) %>%
  add_recipe(podes_recipe)

GAM_fit <- fit(GAM_wf, data = podes_train)
```

```
bind_cols(podes_test,
          predict(GAM_fit, new_data = podes_test)) %>%
  rmse(truth = perc_hh_sa, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        23.5
```

3.5. Random Forest

The final model introduces a more flexible approach to predicting our outcome variable. The random forest model is derived from a decision tree approach to regression which first involves stratifying the predictor space into a set of regions, then utilizing the mean response value of the training observations in each region to assign, and thus predict, a value for any test observation that also falls within that region. Random forests drastically improves the predictive performance of a decision tree model by choosing a random sample of m predictors as split candidates from the full set of predictor variables at each branch in order to de-correlate the trees, and subsequently aggregating multiple decision trees.

```
rf_model <- rand_forest() %>%
  set_engine("randomForest",
             num.threads = parallel::detectCores(),
             importance = TRUE,
             verbose = TRUE) %>%
  set_mode("regression") %>%
  set_args(trees = 50)

rf_wf <- workflow() %>%
  add_model(rf_model) %>%
  add_recipe(podes_recipe)

rf_fit <- fit(rf_wf, podes_train)
```

```
predict(rf_fit, new_data = podes_test) %>%
  bind_cols(podes_test) %>%
  rmse(truth = perc_hh_sa, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        20.1
```

4 Conclusion

Based on our analysis, the random forest model appears to be the most accurate in predicting the percentage of households in a village which receive free health insurance under the national health assistance program. Despite this, the model does not actually predict the outcome variable very well due to the high rmse. Future research should try using other models and embed new variables that may help to better predict the test data. Moreover, using supervised learning models to predict social science outcomes may lead to issues of external validity (both temporal and spatial) therefore, caution must be taken in interpreting the model predictions.

```
result <-
  bind_rows(
    bind_cols(podes_test, predict(lm_fit, new_data = podes_test)) %>%
      rmse(truth = podes_test$perc_hh_sa, estimate = .pred) %>%
  mutate(method = "Linear Regression"),
  bind_cols(podes_test, predict(linear_fit, new_data = podes_test)) %>%
      rmse(truth = podes_test$perc_hh_sa, estimate = .pred) %>%
  mutate(method = "Linear Regression Polynomial"),
  bind_cols(podes_test, predict(tuned_fit, new_data = bake_test)) %>%
      rmse(truth = podes_test$perc_hh_sa, estimate = .pred) %>%
  mutate(method = "Elastic Net"),
  bind_cols(podes_test, predict(rf_fit, new_data = podes_test)) %>%
      rmse(truth = podes_test$perc_hh_sa, estimate = .pred) %>%
  mutate(method = "Random Forest"),
  bind_cols(podes_test, predict(GAM_fit, new_data = podes_test)) %>%
      rmse(truth = podes_test$perc_hh_sa, estimate = .pred) %>%
  mutate(method = "GAM")
  )

knitr::kable(result)
```

.metric	.estimator	.estimate	method
rmse	standard	21.75017	Linear Regression
rmse	standard	21.61557	Linear Regression Polynomial
rmse	standard	21.75421	Elastic Net
rmse	standard	20.11504	Random Forest
rmse	standard	23.48036	GAM