Predicting 2016 Election Results in the USA

Ian Jeffries

5/16/2019

The following R Markdown outlines the steps used by Ian Jeffries to predict the 2016 election results in the USA by county, based on the demographics of that county. This is a supervised machine learning classification problem, and three models were tested to predict the election outcome: K Nearest Neighbor, Artifical Neural Networks, and Decision Trees.

Install necessary packages

The following code will install the packages used in the project:

```
#list of packages used
packages <- c("dplyr", "tidyr", "ggplot2", "class", "rpart", "rpart.plot", "neuralnet",
              "arules", "plyr", "mltools", "arulesViz", "plotly", "RCurl")

#check to see if package is already installed, if not, install
for(p in packages){
  if(!require(p, character.only = TRUE)) {
    install.packages(p)
    library(p, character.only = TRUE)
  }
}
```

Explore the Demographics Dataset

Import the data from my github page and explore.

```
#import our county features dataset
path <- "https://raw.githubusercontent.com/ianjeffries/election-predictions/master/
data/county_facts.csv"

demographics <- read.csv(path, header=TRUE)

#See the names of the columns
names(demographics)
```

```
##  [1] "fips"              "area_name"         "state_abbreviation"
##  [4] "PST045214"         "PST040210"         "PST120214"
##  [7] "POP010210"         "AGE135214"         "AGE295214"
## [10] "AGE775214"         "SEX255214"         "RHI125214"
## [13] "RHI225214"         "RHI325214"         "RHI425214"
## [16] "RHI525214"         "RHI625214"         "RHI725214"
## [19] "RHI825214"         "POP715213"         "POP645213"
## [22] "POP815213"         "EDU635213"         "EDU685213"
## [25] "VET605213"         "LFE305213"         "HSG010214"
## [28] "HSG445213"         "HSG096213"         "HSG495213"
## [31] "HSD410213"         "HSD310213"         "INC910213"
```

```
## [34] "INC110213"          "PVY020213"          "BZA010213"
## [37] "BZA110213"          "BZA115213"          "NES010213"
## [40] "SBO001207"          "SBO315207"          "SBO115207"
## [43] "SBO215207"          "SBO515207"          "SBO415207"
## [46] "SBO015207"          "MAN450207"          "WTN220207"
## [49] "RTN130207"          "RTN131207"          "AFN120207"
## [52] "BPS030214"          "LND110210"          "POP060210"
```

```
#See the structure of the dataset
str(demographics)
```

```
## 'data.frame':    3195 obs. of  54 variables:
##  $ fips              : int  0 1000 1001 1003 1005 1007 1009 1011 1013 1015 ...
##  $ area_name         : Factor w/ 1929 levels "Abbeville County",..: 1764 10 87 94 105 155 170 231 24
##  $ state_abbreviation: Factor w/ 52 levels "","AK","AL","AR",..: 1 1 3 3 3 3 3 3 3 3 ...
##  $ PST045214         : int  318857056 4849377 55395 200111 26887 22506 57719 10764 20296 115916 ...
##  $ PST040210         : int  308758105 4780127 54571 182265 27457 22919 57322 10915 20946 118586 ...
##  $ PST120214         : num  3.3 1.4 1.5 9.8 -2.1 -1.8 0.7 -1.4 -3.1 -2.3 ...
##  $ POP010210         : int  308745538 4779736 54571 182265 27457 22915 57322 10914 20947 118572 ...
##  $ AGE135214         : num  6.2 6.1 6 5.6 5.7 5.3 6.1 6.3 6.1 5.7 ...
##  $ AGE295214         : num  23.1 22.8 25.2 22.2 21.2 21 23.6 21.4 23.6 22.2 ...
##  $ AGE775214         : num  14.5 15.3 13.8 18.7 16.5 14.8 17 14.9 18 16 ...
##  $ SEX255214         : num  50.8 51.5 51.4 51.2 46.6 45.9 50.5 45.3 53.6 51.8 ...
##  $ RHI125214         : num  77.4 69.7 77.9 87.1 50.2 76.3 96 26.9 53.9 75.8 ...
##  $ RHI225214         : num  13.2 26.7 18.7 9.6 47.6 22.1 1.8 70.1 44 21.1 ...
##  $ RHI325214         : num  1.2 0.7 0.5 0.7 0.6 0.4 0.6 0.8 0.4 0.5 ...
##  $ RHI425214         : num  5.4 1.3 1.1 0.9 0.5 0.2 0.3 0.3 0.9 0.9 ...
##  $ RHI525214         : num  0.2 0.1 0.1 0.1 0.2 0.1 0.1 0.7 0 0.1 ...
##  $ RHI625214         : num  2.5 1.5 1.8 1.6 0.9 0.9 1.2 1.1 0.8 1.7 ...
##  $ RHI725214         : num  17.4 4.1 2.7 4.6 4.5 2.1 8.7 7.5 1.2 3.5 ...
##  $ RHI825214         : num  62.1 66.2 75.6 83 46.6 74.5 87.8 22.1 53.1 72.9 ...
##  $ POP715213         : num  84.9 85 85 82.1 84.8 86.6 88.7 84.7 94.6 83.6 ...
##  $ POP645213         : num  12.9 3.5 1.6 3.6 2.9 1.2 4.3 5.4 0.8 2.4 ...
##  $ POP815213         : num  20.7 5.2 3.5 5.5 5 2.1 7.3 5.2 1.7 4.5 ...
##  $ EDU635213         : num  86 83.1 85.6 89.1 73.7 77.5 77 67.8 76.3 78.6 ...
##  $ EDU685213         : num  28.8 22.6 20.9 27.7 13.4 12.1 12.1 12.5 14 16.1 ...
##  $ VET605213         : int  21263779 388865 5922 19346 2120 1327 4540 636 1497 11385 ...
##  $ LFE305213         : num  25.5 24.2 26.2 25.9 24.6 27.6 33.9 26.9 24 22.5 ...
##  $ HSG010214         : int  133957180 2207912 22751 107374 11799 8978 23826 4461 9916 53289 ...
##  $ HSG445213         : num  64.9 69.7 76.8 72.6 67.7 79 81 74.3 70.3 68.7 ...
##  $ HSG096213         : num  26 15.9 8.3 24.4 10.6 7.3 4.5 8.7 13.3 13.8 ...
##  $ HSG495213         : int  176700 122500 136200 168600 89200 90500 117100 70600 74700 100600 ...
##  $ HSD410213         : int  115610216 1838683 20071 73283 9200 7091 21108 3741 8235 45196 ...
##  $ HSD310213         : num  2.63 2.55 2.71 2.52 2.66 3.03 2.7 2.73 2.47 2.54 ...
##  $ INC910213         : int  28155 23680 24571 26766 16829 17427 20730 18628 17403 20828 ...
##  $ INC110213         : int  53046 43253 53682 50221 32911 36447 44145 32033 29918 39962 ...
##  $ PVY020213         : num  15.4 18.6 12.1 13.9 26.7 18.1 15.8 21.6 28.4 21.9 ...
##  $ BZA010213         : int  7488353 97578 817 4871 464 275 660 112 393 2311 ...
##  $ BZA110213         : int  118266253 1603100 10120 54988 6611 3145 6798 0 5711 34871 ...
##  $ BZA115213         : num  2 1.1 2.1 3.7 -5.6 7.5 3.4 0 2.7 0.6 ...
##  $ NES010213         : int  23005620 311578 2947 16508 1546 1126 3563 470 1095 6352 ...
##  $ SBO001207         : int  27092908 382350 4067 19035 1667 1385 4458 417 1769 8713 ...
##  $ SBO315207         : num  7.1 14.8 15.2 2.7 0 14.9 0 0 0 7.2 ...
##  $ SBO115207         : num  0.9 0.8 0 0.4 0 0 0 0 0 0 ...
```

```
##  $ SBO215207         : num  5.7 1.8 1.3 1 0 0 0 0 3.3 1.6 ...
##  $ SBO515207         : num  0.1 0.1 0 0 0 0 0 0 0 0 ...
##  $ SBO415207         : num  8.3 1.2 0.7 1.3 0 0 0 0 0 0.5 ...
##  $ SBO015207         : num  28.8 28.1 31.7 27.3 27 0 23.2 38.8 0 24.7 ...
##  $ MAN450207         : num  5.32e+09 1.13e+08 0.00 1.41e+06 0.00 ...
##  $ WTN220207         : num  4.17e+09 5.23e+07 0.00 0.00 0.00 ...
##  $ RTN130207         : num  3.92e+09 5.73e+07 5.98e+05 2.97e+06 1.88e+05 ...
##  $ RTN131207         : int  12990 12364 12003 17166 6334 5804 5622 3995 11326 13678 ...
##  $ AFN120207         : int  613795732 6426342 88157 436955 0 10757 20941 3670 28427 186533 ...
##  $ BPS030214         : int  1046363 13369 131 1384 8 19 3 1 2 114 ...
##  $ LND110210         : num  3531905 50645 594 1590 885 ...
##  $ POP060210         : num  87.4 94.4 91.8 114.6 31 ...
```

```
#find the state summary info and remove
#(want only counties, this dataset includes state summaries)
demographics <- demographics[-which(demographics$state_abbreviation == ""), ]
```

Add in Dictionary information

Add in the dictionary file to replace columm headers with readable values.

```
#pull in dictionary to get true names of the variables
path <- "https://raw.githubusercontent.com/ianjeffries/election-predictions/master/
data/county_facts_dictionary.csv"

var_names <- read.csv(path, header = TRUE)

#view variable names
print(var_names)
```

```
##           column_name
## 1           PST045214
## 2           PST040210
## 3           PST120214
## 4           POP010210
## 5           AGE135214
## 6           AGE295214
## 7           AGE775214
## 8           SEX255214
## 9           RHI125214
## 10          RHI225214
## 11          RHI325214
## 12          RHI425214
## 13          RHI525214
## 14          RHI625214
## 15          RHI725214
## 16          RHI825214
## 17          POP715213
## 18          POP645213
## 19          POP815213
## 20          EDU635213
## 21          EDU685213
```

```
## 22          VET605213
## 23          LFE305213
## 24          HSG010214
## 25          HSG445213
## 26          HSG096213
## 27          HSG495213
## 28          HSD410213
## 29          HSD310213
## 30          INC910213
## 31          INC110213
## 32          PVY020213
## 33          BZA010213
## 34          BZA110213
## 35          BZA115213
## 36          NES010213
## 37          SBO001207
## 38          SBO315207
## 39          SBO115207
## 40          SBO215207
## 41          SBO515207
## 42          SBO415207
## 43          SBO015207
## 44          MAN450207
## 45          WTN220207
## 46          RTN130207
## 47          RTN131207
## 48          AFN120207
## 49          BPS030214
## 50          LND110210
## 51          POP060210
## 52               fips
## 53          area_name
## 54 state_abbreviation
## 55        Percent_Dem
## 56        Percent_Rep
##                                                  description
## 1                                            Population_2014
## 2                                            Population_2010
## 3                                 Population_Percent_Change
## 4                                            Population_2010
## 5                                 Percent_Under_5_Years_Old
## 6                                Percent_Under_18_Years_Old
## 7                                Percent_65_Years_and_Older
## 8                                            Percent_Females
## 9                                       Percent_White_Alone
## 10                        Percent_Black_or_African_American
## 11                  Percent_American_Indian_or_Alaska_Native
## 12                                              Percent_Asian
## 13      Percent_Native_Hawaiian_and_Other_Pacific_Islander
## 14                                Percent_Two_or_More_Races
## 15                               Percent_Hispanic_or_Latino
## 16              Percent_White_Alone_not_Hispanic_or_Latino
## 17            Percent_Living_in_same_house_1_year_and_over
## 18                             Percent_Foreign_born_persons
```

```
## 19             Percent_Language_other_than_English_spoken_at_home
## 20                         Percent_High_school_graduate_or_higher
## 21                             Percent_Bachelors_degree_or_higher
## 22                                                Percent_Veterans
## 23                                Mean_travel_time_to_work_minutes
## 24                                                    Housing_units
## 25                                      Percent_Homeownership_rate
## 26                            Housing_units_in_multi_unit_structures
## 27                     Median_value_of_owner_occupied_housing_units
## 28                                                 Number_Households
## 29                                             Persons_per_household
## 30                        Per_capita_money_income_in_past_12_months
## 31                                             Median_household_income
## 32                               Percent_Persons_below_poverty_level
## 33                                    Private_nonfarm_establishments
## 34                                        Private_nonfarm_employment
## 35                                        Private_nonfarm_employment
## 36                                        Nonemployer_establishments
## 37                                             Total_number_of_firms
## 38                                         Percent_Black_owned_firms
## 39          Percent_American_Indian_and_Alaska_Native_owned_firms
## 40                                         Percent_Asian_owned_firms
## 41 Percent_Native_Hawaiian_and_Other_Pacific_Islander_owned_firms
## 42                                       Percent_Hispanic_owned_firms
## 43                                         Percent_Women_owned_firms
## 44                                 Manufacturers_shipments_2007_1000
## 45                               Merchant_wholesaler_sales_2007_$1000
## 46                                             Retail_sales_2007_$1000
## 47                                       Retail_sales_per_capita_2007
## 48                  Accommodation_and_food_services_sales_2007_1000
## 49                                                Building_permits
## 50                                         Land_area_in_square_miles
## 51                                       Population_per_square_mile
## 52                                                              fips
## 53                                                            county
## 54                                                             state
## 55                                                        percent_dem
## 56                                                        percent_rep
```

Pull in 2016 Election Results and Clean the Data

As of now, the US primarily operates in a two-party system. Because of this, only the democratic and republican candidates are of interest.

```
#add in the election results
path <- "https://raw.githubusercontent.com/ianjeffries/election-predictions/master/
data/pres16results.csv"

election_results <- read.csv(path, header = TRUE)

#see the names of the columns
names(election_results)
```

```
## [1] "county"      "fips"        "cand"        "st"          "pct_report"
## [6] "votes"       "total_votes" "pct"         "lead"
```

```
#see structure of the dataset
str(election_results)
```

```
## 'data.frame':    18475 obs. of  9 variables:
##  $ county     : Factor w/ 1969 levels "Abbeville County",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ fips       : Factor w/ 3289 levels "10001","10003",..: 3282 3282 3282 3282 3282 3282 3282 3282 328
##  $ cand       : Factor w/ 32 levels " None of these candidates",..: 7 13 11 15 9 6 12 27 1 26 ...
##  $ st         : Factor w/ 52 levels "AK","AL","AR",..: 45 45 45 45 45 45 45 45 45 45 ...
##  $ pct_report : num  0.995 0.995 0.995 0.995 0.995 ...
##  $ votes      : int  60350241 60981118 4164589 1255968 451636 180877 48308 32120 28824 23501 ...
##  $ total_votes: int  127592176 127592176 127592176 127592176 127592176 127592176 127592176 127592176
##  $ pct        : num  0.47299 0.47794 0.03264 0.00984 0.00354 ...
##  $ lead       : Factor w/ 2 levels "Donald Trump",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
#drop columns we don't need
election_results <- election_results[-c(2, 5, 8)]

#filter by the republican and democratic candidates & remove null values
ER_clean <- election_results %>%
                filter(cand %in% c("Donald Trump", "Hillary Clinton") & county != "<NA>")

#change to wide dataset to merge the election results with the demographics dataframe
ER_clean <- spread(ER_clean, key = cand, value = votes)

#add in % won by
ER_clean$Percent_Rep <- round(ER_clean$`Donald Trump` / ER_clean$total_votes, 2)
ER_clean$Percent_Dem <- round(ER_clean$`Hillary Clinton` / ER_clean$total_votes, 2)
```

Join the Demographic and Election Results Datasets

Now that cleanup is complete, the two datasets can be joined.

```
#join our election results with the demographics table
final_dataset <- left_join(demographics, select(ER_clean, Percent_Dem,
                                                Percent_Rep, st, county),
                           by = c("area_name" = "county", "state_abbreviation" = "st"))

#find number of counties with N/A election results
print(paste0("Number of counties with null election results: ",
             nrow(final_dataset[which(complete.cases(final_dataset) == FALSE), ])))
```

```
## [1] "Number of counties with null election results: 33"
```

```
#looks like these aren't counties (the majority) and
#the N is small enough I feel comfortable removing them
final_dataset <- na.omit(final_dataset)
```

Final Cleanup

The final cleanup steps are to remove irrelevant values and reassign column names for analysis. The data will also be normalized, with 0 or 1 values assigned to the target variable.

```
#change any column in dataset based on % of the population to true percentage
#(basically already normalized between 0 and 1)
final_dataset[,c(8:24, 28:29, 35, 41:46)] <- final_dataset[,c(8:24, 28:29, 35, 41:46)]/100

#drop clearly irrelevant values
final_dataset <- final_dataset[ , -c(5:8, 38)]

#assign headers to something more understandable, based on the dictionary dataframe
names(final_dataset) <- var_names$description[match(names(final_dataset),
                                                    var_names$column_name)]

#assign labels that weren't in the dictionary
names(final_dataset[ , c(1, 2, 3, 50, 51)]) <- c("fips", "area_name", "state_abbreviation",
                                                 "Percent_Dem", "Percent_Rep")

#normalize any data not described as a %
for (i in c(4, 21, 22, 23, 26, 27, 28, 29, 30, 32, 33,
            34, 35, 42, 43, 44, 45, 46, 47, 48, 49)) {
  final_dataset[,i] <- ((final_dataset[,i] - min(final_dataset[,i])) /
                          (max(final_dataset[,i]) - min(final_dataset[,i])) *
                          (1 - 0) + 0)
}

#add in 1 or 0 if they voted democratic or republican
final_dataset$Winner <- 0

for (i in 1:nrow(final_dataset)) {
    if (final_dataset[i, "percent_dem"] < .50) {
      final_dataset[i, "Winner"] <- 0
      } else {
        final_dataset[i, "Winner"] <- 1
      }}

#set winner column to factor for classification
final_dataset$Winner <- as.factor(final_dataset$Winner)
```

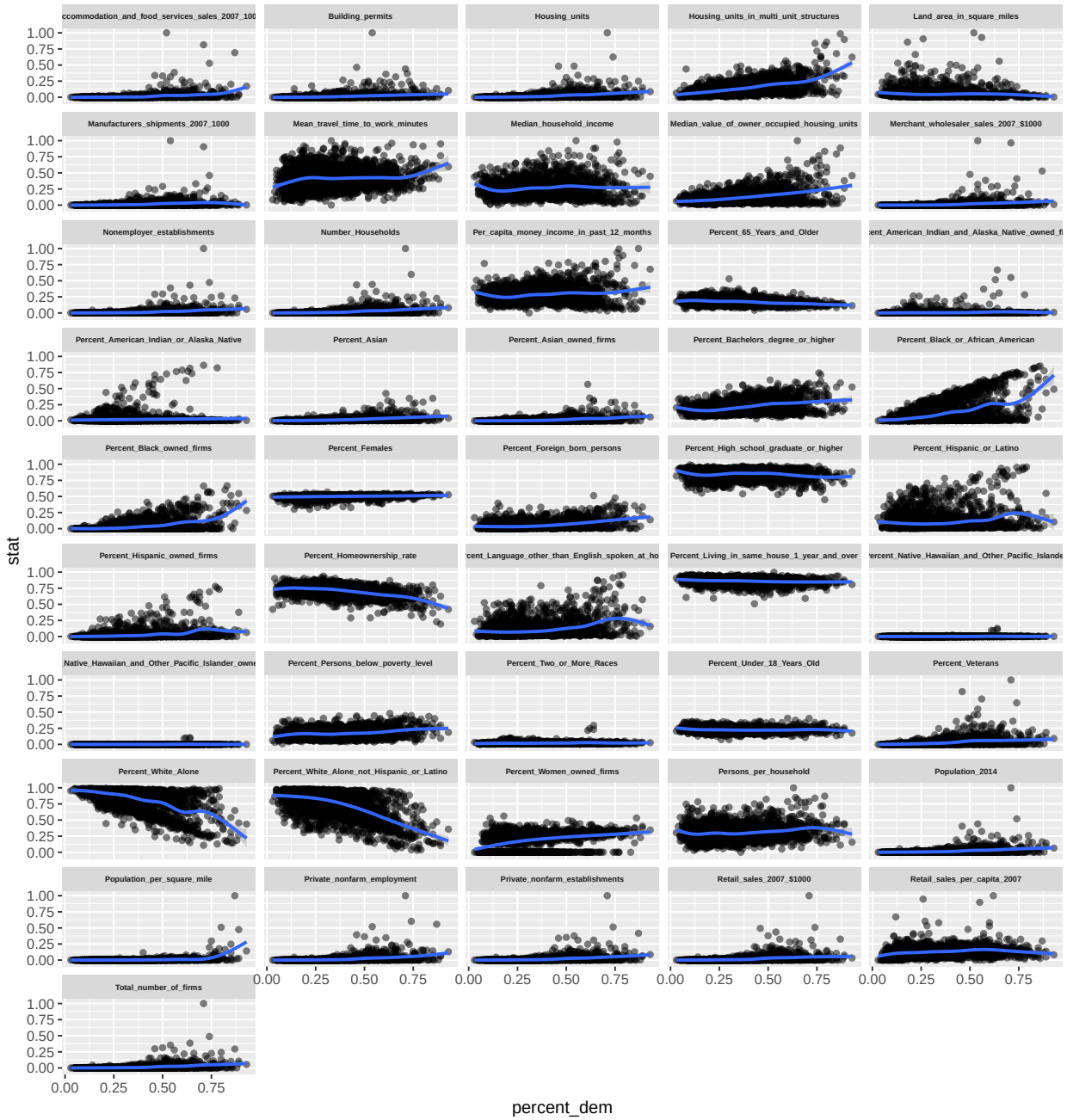
Look for Variables to use in Classification

Rather than use all 45 variables in classification, each normalized variable was graphed against the percentage vote for the democratic candidate to see if there are any clear correlations. Any variables that didn't seem to hold a correlation were removed.

```
#create narrow dataset to understand relationship between attributes and voting preference
final_narrow <- gather(final_dataset, key = demographic_measure, value = stat, 4:49)

#create side by side scatterplots to look for possible correlations
ggplot(final_narrow, aes(x = percent_dem, y = stat)) +
  geom_point(alpha = .5) +
```

```
  geom_smooth() +
  facet_wrap(vars(demographic_measure), ncol = 5) +
  theme(strip.text = element_text(size = 5, face = "bold"))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
#remove columns that don't seem to have a correlation to voting preference
final_dataset <- final_dataset[ , -c(1,5,7,10,12,13,16,28,30,34,35,
                                     37,39,42,43,44,45,47,48)]
```

Classification Prep Work

Before the models can be created, training and test datasets must be set up. An accuracy matrix was also created to house the results from each model.

```
###### CREATE TRAINING AND TEST DATASETS ######

#take only the predictor values and class from our dataset
e_predictions <- final_dataset[ , c(3:30, 33)]

#set set to make our test data reproducable
set.seed(111692)

#get 80/20 sample of data
dist <- sample(2, size = nrow(e_predictions), replace = TRUE, prob = c(.8, .2))

#set test and training data
e_training <- e_predictions[dist == 1, ]
e_test <- e_predictions[dist == 2, ]

#show dimensions of training dataset
dim(e_training)
```

```
## [1] 2434   29
```

```
#show dimensions of test dataset
dim(e_test)
```

```
## [1] 676  29
```

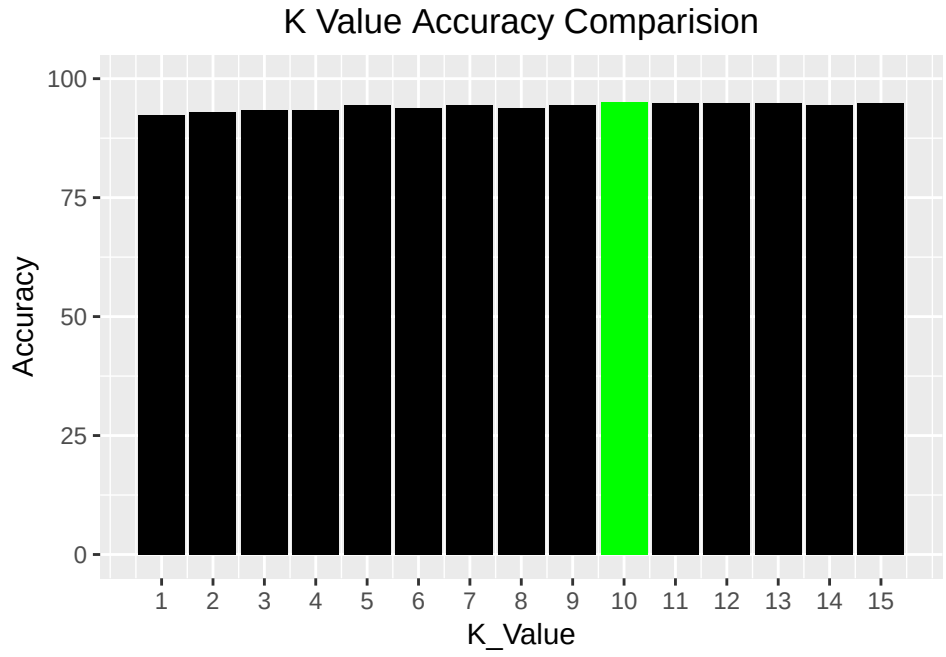
```
#create accuracy table for comparison
algorithm_results <- matrix(data = rep(0, 6), nrow = 2, ncol = 3,
                            dimnames = list(c("Accuracy %:", "Precision %:"),
                                            c("KNN", "DTREE", "ANN")))

#show accuracy table
print(algorithm_results)
```

```
##              KNN DTREE ANN
## Accuracy %:    0     0   0
## Precision %:   0     0   0
```

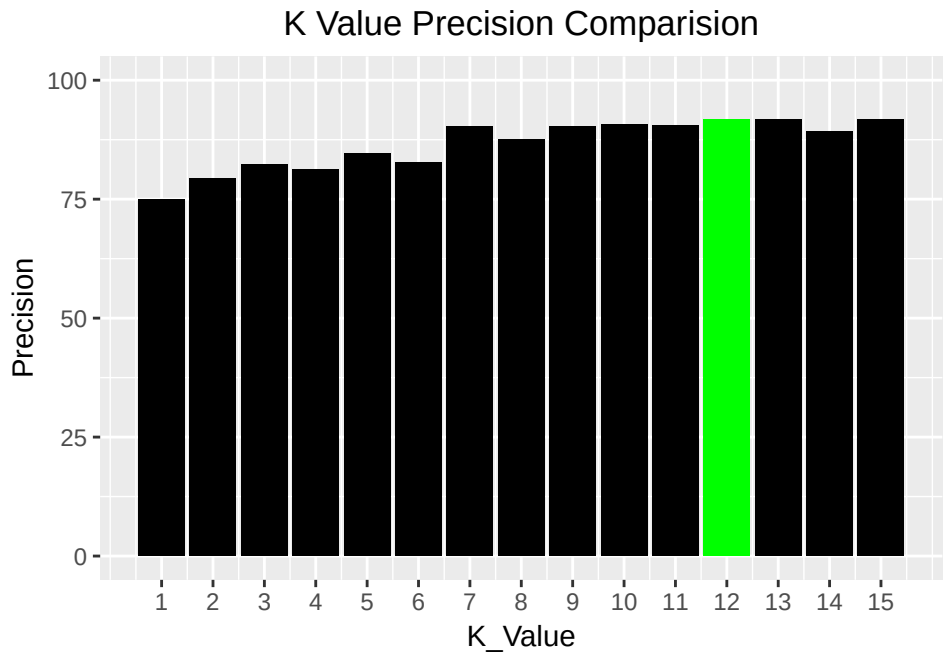
K-Nearest Neighbor Classification

```
###### KNN CLASSIFICATION ######

#create KNN accuracy matrix to compare 15 K-Values
KNN_accuracy <- matrix(data= rep(0, 45), ncol = 3, dimnames = list(c(1:15),
                                                                   c("K_Value", "Accuracy",
                                                                     "Precision")))
```

```r
#set each K value as a factor
KNN_accuracy[ , 1] <- as.factor(c(1:15))

#test multiple K values and store the results
#(For KNN function, you need to remove the columns with actual classification)
for (i in 1:15) {
  KNN_results <- as.data.frame(knn(e_training[,1:28], e_test[,1:28], e_training$Winner,
                                   k = i, prob = TRUE))

  #create a comparison table
  KNN_table <- table(KNN_results[,1], e_test$Winner)

  #find % accuracy
  KNN_accuracy[i , 2] <- round(((sum(diag(KNN_table)) / sum(KNN_table)) * 100), 2)

  #find % precision
  KNN_accuracy[i , 3] <- round((KNN_table[2, 2] / (KNN_table[2, 2] +
                                                     KNN_table[2, 1]) * 100), 2)
}

#plot accuracy results
theme_update(plot.title = element_text(hjust = 0.5))
ggplot(as.data.frame(KNN_accuracy), aes(x = K_Value, y = Accuracy)) +
  geom_bar(stat = "identity", fill = "black") +
  ggtitle("K Value Accuracy Comparision") +
  scale_x_continuous(breaks = c(1:15)) +
  scale_y_continuous(limits = c(0, 100)) +
  geom_bar(data=subset(as.data.frame(KNN_accuracy), Accuracy==max(Accuracy)),
           aes(K_Value, Accuracy),
           fill="green", stat="identity")
```



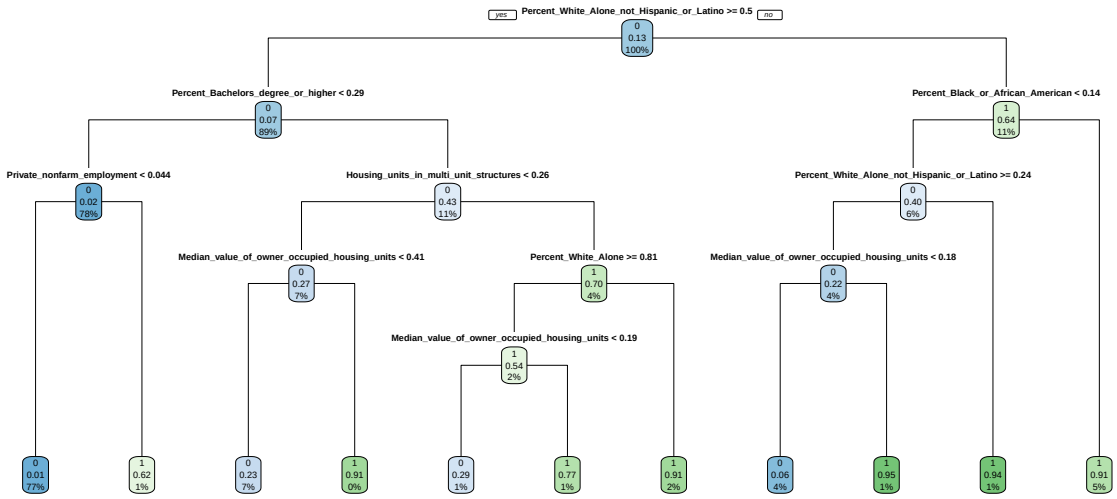
```r
#plot precision results
ggplot(as.data.frame(KNN_accuracy), aes(x = K_Value, y = Precision)) +
  geom_bar(stat = "identity", fill = "black") +
  ggtitle("K Value Precision Comparision") +
  scale_x_continuous(breaks = c(1:15)) +
  scale_y_continuous(limits = c(0, 100)) +
  geom_bar(data=subset(as.data.frame(KNN_accuracy), Precision==max(Precision)),
           aes(K_Value, Precision),
           fill="green", stat="identity")
```




```r
#add to results table (highest k value at the time I ran)
algorithm_results[1:2, 1] <- KNN_accuracy[12, 2:3]
```

Decision Tree

```r
###### DECISION TREE CLASSIFICATION ######

#Train the decision tree
e_tree <- rpart(Winner ~ ., data = e_training)

#plot the branches of the tree
rpart.plot(e_tree, type=1)
```




```
#returns a list of strings summarizing the branch path to each node
rpart.rules(e_tree, style = "tall") #useful to see what conditions predict a certain factor
```

```
## Winner is 0.01 when
##     Percent_White_Alone_not_Hispanic_or_Latino >= 0.50
##     Percent_Bachelors_degree_or_higher < 0.29
##     Private_nonfarm_employment < 0.044
##
## Winner is 0.06 when
##     Percent_White_Alone_not_Hispanic_or_Latino is 0.24 to 0.50
##     Median_value_of_owner_occupied_housing_units < 0.18
##     Percent_Black_or_African_American < 0.14
##
## Winner is 0.23 when
##     Percent_White_Alone_not_Hispanic_or_Latino >= 0.50
##     Percent_Bachelors_degree_or_higher >= 0.29
##     Median_value_of_owner_occupied_housing_units < 0.41
##     Housing_units_in_multi_unit_structures < 0.26
##
## Winner is 0.29 when
##     Percent_White_Alone_not_Hispanic_or_Latino >= 0.50
##     Percent_Bachelors_degree_or_higher >= 0.29
##     Median_value_of_owner_occupied_housing_units < 0.19
##     Housing_units_in_multi_unit_structures >= 0.26
```

```
##     Percent_White_Alone >= 0.81
##
## Winner is 0.62 when
##     Percent_White_Alone_not_Hispanic_or_Latino >= 0.50
##     Percent_Bachelors_degree_or_higher < 0.29
##     Private_nonfarm_employment >= 0.044
##
## Winner is 0.77 when
##     Percent_White_Alone_not_Hispanic_or_Latino >= 0.50
##     Percent_Bachelors_degree_or_higher >= 0.29
##     Median_value_of_owner_occupied_housing_units >= 0.19
##     Housing_units_in_multi_unit_structures >= 0.26
##     Percent_White_Alone >= 0.81
##
## Winner is 0.91 when
##     Percent_White_Alone_not_Hispanic_or_Latino < 0.50
##     Percent_Black_or_African_American >= 0.14
##
## Winner is 0.91 when
##     Percent_White_Alone_not_Hispanic_or_Latino >= 0.50
##     Percent_Bachelors_degree_or_higher >= 0.29
##     Median_value_of_owner_occupied_housing_units >= 0.41
##     Housing_units_in_multi_unit_structures < 0.26
##
## Winner is 0.91 when
##     Percent_White_Alone_not_Hispanic_or_Latino >= 0.50
##     Percent_Bachelors_degree_or_higher >= 0.29
##     Housing_units_in_multi_unit_structures >= 0.26
##     Percent_White_Alone < 0.81
##
## Winner is 0.94 when
##     Percent_White_Alone_not_Hispanic_or_Latino < 0.24
##     Percent_Black_or_African_American < 0.14
##
## Winner is 0.95 when
##     Percent_White_Alone_not_Hispanic_or_Latino is 0.24 to 0.50
##     Median_value_of_owner_occupied_housing_units >= 0.18
##     Percent_Black_or_African_American < 0.14
```

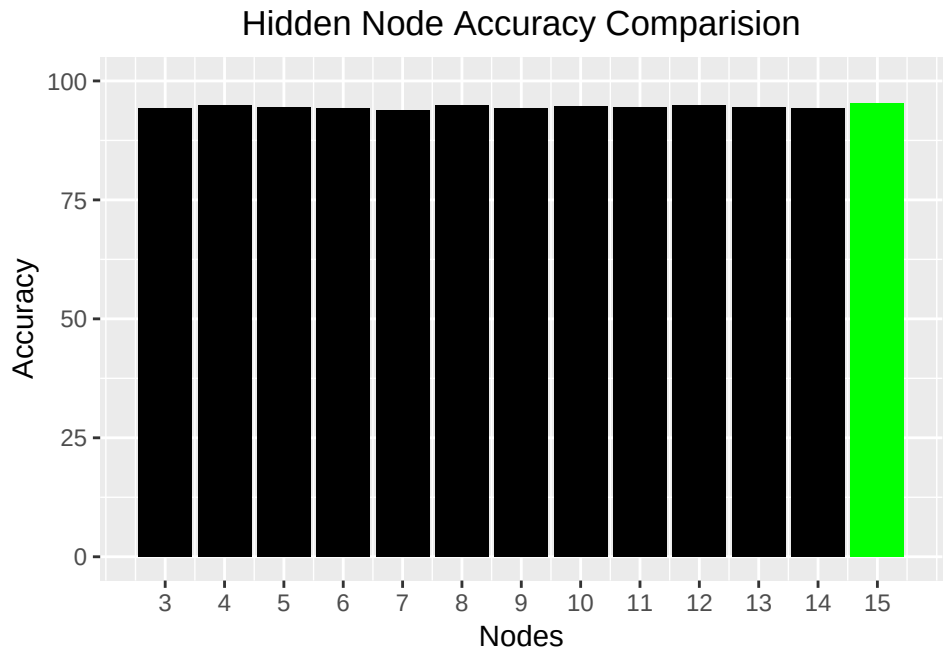
```r
#predict using that model on test data
tree_results <- as.data.frame(predict(e_tree, newdata = e_test, type = "class"))

#create a comparison table
tree_table <- table(tree_results[ , 1], e_test$Winner)

#create an accuracy table
tree_accuracy <- matrix(data= rep(0, 2), ncol = 2, dimnames =
                          list("%",c("Accuracy", "Precision")))

#find accuracy
tree_accuracy[1, 1] <- round(((sum(diag(tree_table)) / sum(tree_table)) * 100), 2)

#find % precision
tree_accuracy[1 , 2] <- round((tree_table[2, 2] / (tree_table[2, 2] +
```

```
                                                    tree_table[2, 1]) * 100), 2)

#add to results table
algorithm_results[1:2, 2] <- tree_accuracy
```

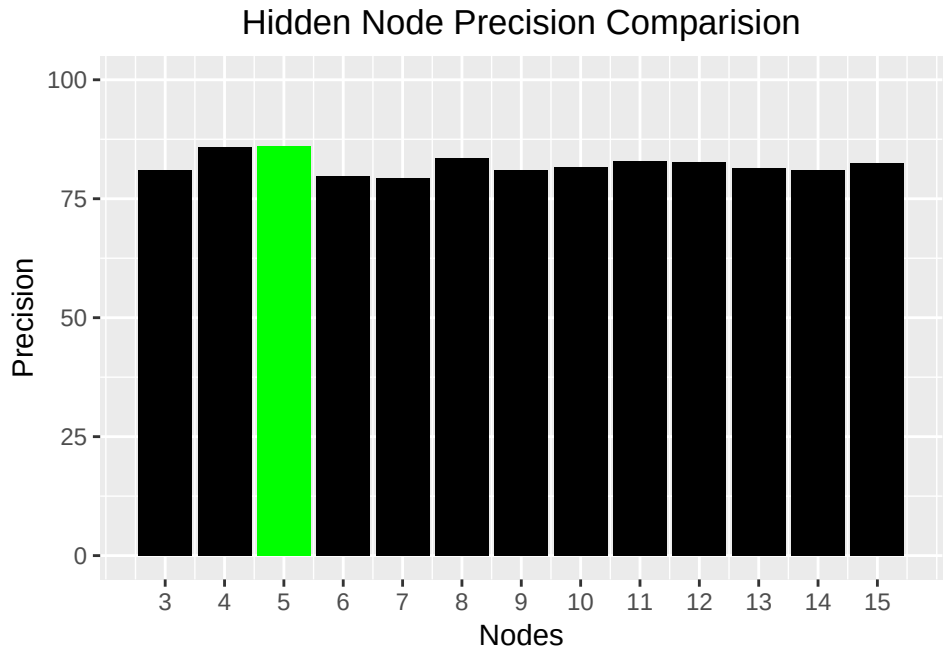
Neural Networks

```
###### ANN CLASSIFICATION ######

#build the neural network

#neural networks do not work for categorical variables (ie 1, 2 or 3 ranking)
#To get around this we put each variable in it's own column, either a 1 or 0
#for columns 1, 2 or 3 (use the model.matrix function to do this)

#first move categorical variables into their own columns
#(I believe this formula finds columns that are categorical classes)
e_training_m <- model.matrix(~0 + ., data = e_training)

#do the same for the test dataset
e_test_m <- model.matrix(~0 + ., data = e_test)

#create an accuracy table
ANN_accuracy <- matrix(data= rep(0, 39), ncol = 3, dimnames = list(c(3:15),
                                                                   c("Nodes", "Accuracy",
                                                                     "Precision")))

ANN_accuracy[ , 1] <- c(3:15)

#test multiple hidden nodes in our model
for (i in 3:15) {

  #build the artifical neural network using neuralnet function
  e_network <- neuralnet(Winner0 + Winner1 ~ Population_2014 + Percent_65_Years_and_Older +
                           Percent_White_Alone + Percent_Black_or_African_American +
                           Percent_Asian + Percent_Hispanic_or_Latino +
                           Percent_White_Alone_not_Hispanic_or_Latino +
                           Percent_Foreign_born_persons +
                           Percent_Language_other_than_English_spoken_at_home +
                           Percent_High_school_graduate_or_higher +
                           Percent_Bachelors_degree_or_higher +
                           Percent_Veterans + Mean_travel_time_to_work_minutes +
                           Housing_units + Percent_Homeownership_rate +
                           Housing_units_in_multi_unit_structures +
                           Median_value_of_owner_occupied_housing_units +
                           Number_Households + Per_capita_money_income_in_past_12_months +
                           Percent_Persons_below_poverty_level +
                           Private_nonfarm_establishments + Private_nonfarm_employment +
                           Percent_Black_owned_firms + Percent_Asian_owned_firms +
                           Percent_Hispanic_owned_firms + Percent_Women_owned_firms +
                           Accommodation_and_food_services_sales_2007_1000 +
```

```
                          Population_per_square_mile,
                        data = e_training_m,
                        hidden = i,
                        lifesign = "full",
                        linear.output = FALSE,
                        threshold = .1)

  #test the ANN on the test data
  ANN_results <- compute(e_network, e_test_m[ ,1:28])
                #^ computes result of each row, remove winner variables

  #extract the results
  prediction <- ANN_results$net.result

  #use max.col function to pull the original values from our test dataset into one column
  original_values <- max.col(e_test_m[ ,29:30]) - 1

  #use max.col function to get prediction values
  prediction_values <- max.col(prediction) - 1

  #create a dataframe with both values
  results <- data.frame(actual = original_values, prediction = prediction_values)

  #create a comparison table
  ANN_table <- table(results$prediction, e_test$Winner)

  #find accuracy
  ANN_accuracy[i-2, 2] <- round(((sum(diag(ANN_table)) / sum(ANN_table)) * 100), 2)

  #find % precision
  ANN_accuracy[i-2 , 3] <- round((ANN_table[2, 2] / (ANN_table[2, 2] + ANN_table[2, 1]) *
                                    100), 2)
}

#plot the accuracy results
ggplot(as.data.frame(ANN_accuracy), aes(x = Nodes, y = Accuracy)) +
  geom_bar(stat = "identity", fill = "black") +
  ggtitle("Hidden Node Accuracy Comparision") +
  scale_x_continuous(breaks = c(3:15)) +
  scale_y_continuous(limits = c(0, 100)) +
  geom_bar(data=subset(as.data.frame(ANN_accuracy), Accuracy==max(Accuracy)),
           aes(Nodes, Accuracy),
           fill="green", stat="identity")
```




```
#plot the precision results
ggplot(as.data.frame(ANN_accuracy), aes(x = Nodes, y = Precision)) +
  geom_bar(stat = "identity", fill = "black") +
  ggtitle("Hidden Node Precision Comparision") +
  scale_x_continuous(breaks = c(3:15)) +
  scale_y_continuous(limits = c(0, 100)) +
  geom_bar(data=subset(as.data.frame(ANN_accuracy), Precision==max(Precision)),
           aes(Nodes, Precision),
           fill="green", stat="identity")
```

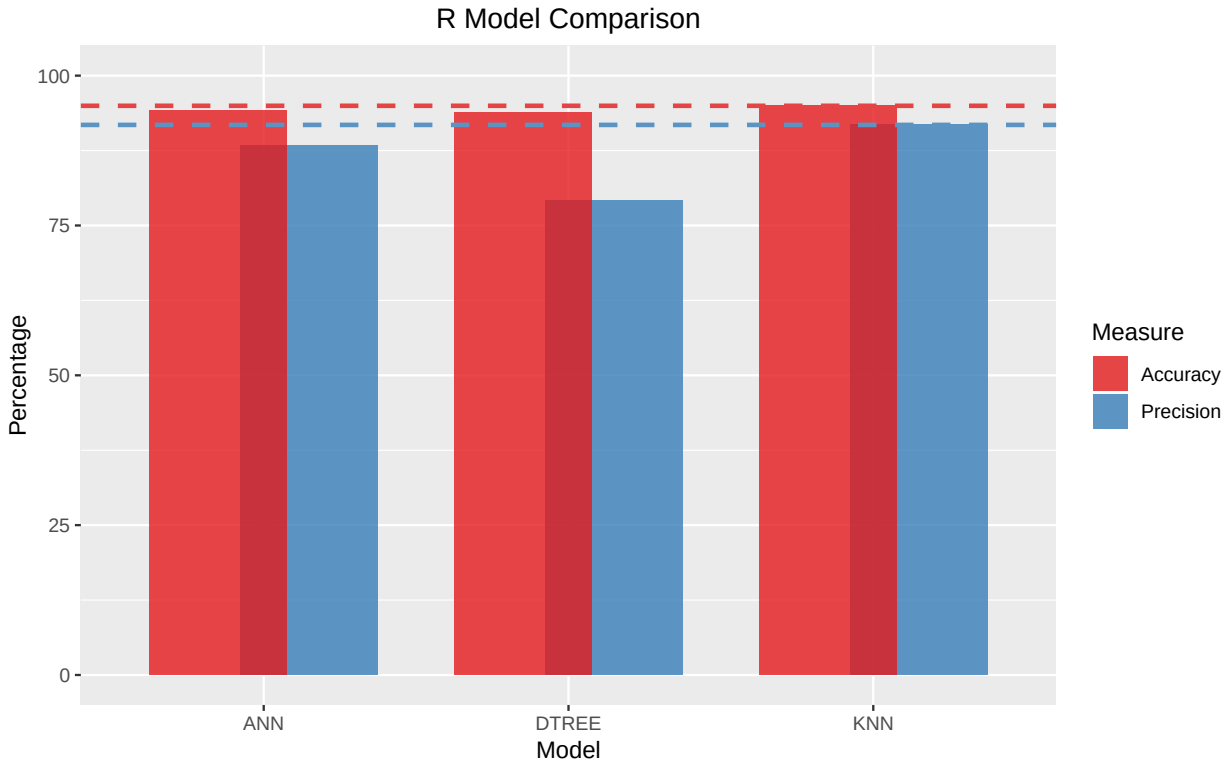


```r
#add to results table
algorithm_results[1:2, 3] <- ANN_accuracy[1, 2:3]
print(algorithm_results)
```

```
##                 KNN DTREE   ANN
## Accuracy %:  94.97 93.79 94.23
## Precision %: 91.78 79.12 81.11
```

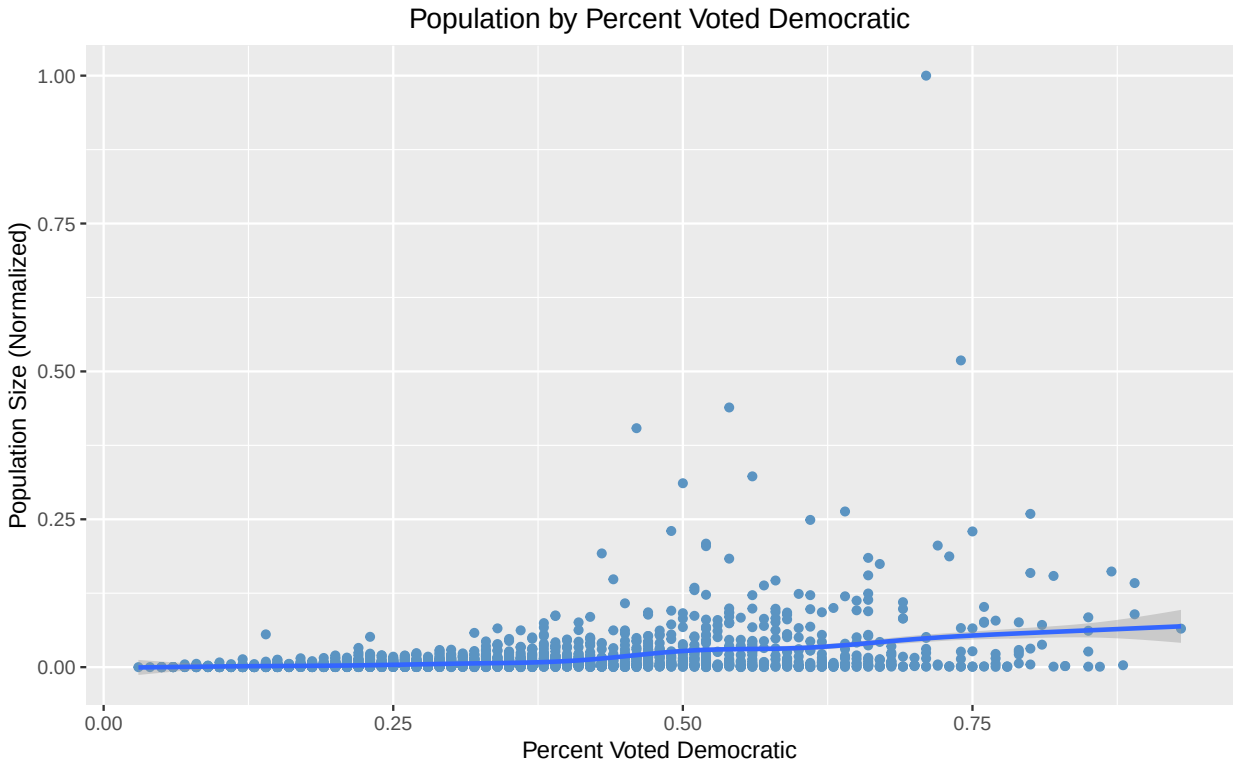
Model Comparison

```r
###### COMPARISON OF CLASSIFICATION MODELS ######

#create final results (results kept changing as I used different work stations,
#                       using values from last test for the write up)
final_results <- matrix(data= rep(0, 18), ncol = 3, dimnames = list(c(1:6),
                                                                    c("Model", "Measure",
                                                                      "Percentage")))
final_results <- as.data.frame(final_results)

#add results manually
final_results$Model <- c("KNN", "DTREE", "ANN", "KNN", "DTREE", "ANN")

final_results$Measure <- c("Accuracy", "Accuracy", "Accuracy",
                           "Precision", "Precision", "Precision")

final_results$Percentage <- c(94.97, 93.79, 94.23, 91.78, 79.12, 88.31)

#plot the accuracy results
ggplot(final_results, aes(x = Model, y = Percentage, fill = Measure)) +
  geom_bar(stat = "identity", position = position_dodge(width = .6), alpha = .8) +
  ggtitle("R Model Comparison") +
  scale_y_continuous(limits = c(0, 100)) +
  geom_hline(yintercept = 94.97, linetype = "dashed", size = 1, color = "#E54445") +
  geom_hline(yintercept = 91.78, linetype = "dashed", size = 1, color = "#5B94C2") +
  scale_fill_brewer(palette = "Set1")
```



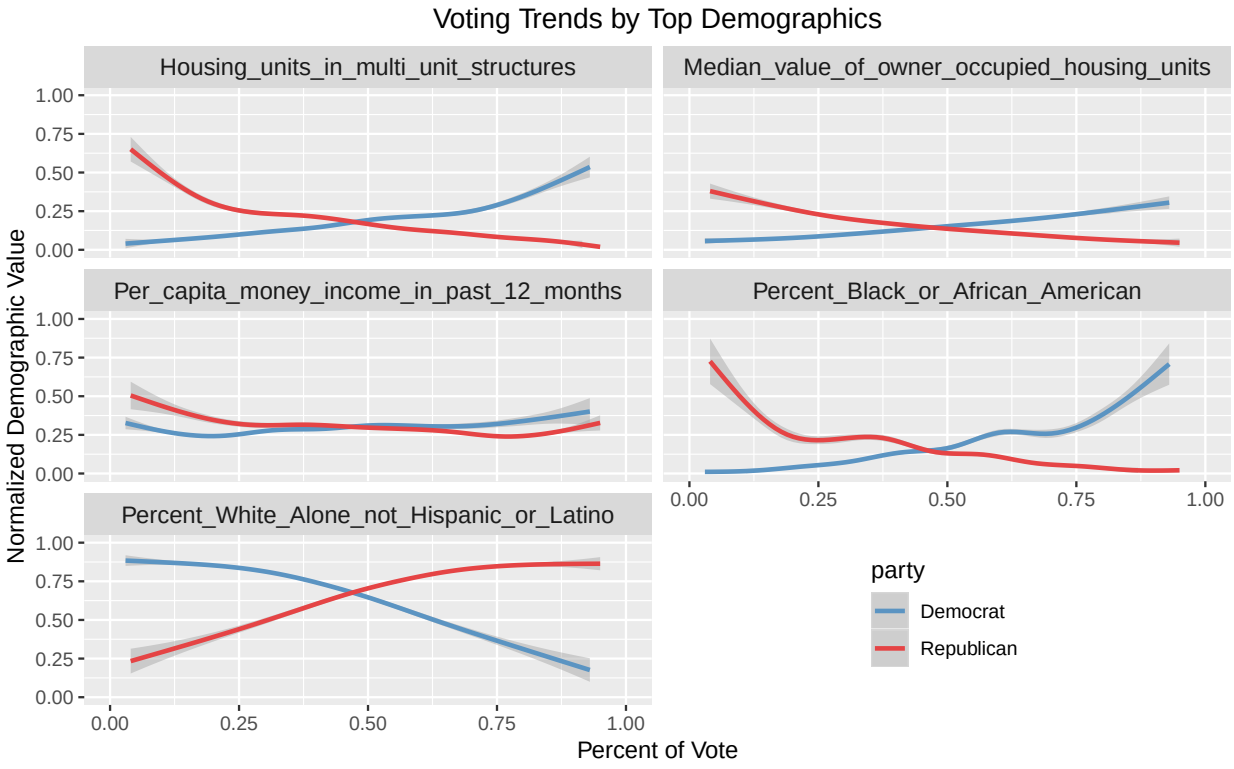

```
#make narrow dataset even more narrow to plot by party
final_narrow2 <- gather(final_narrow, key = party, value = percent, 4:5)

#plot vote by population (to see if high populations voted democratic)
final_narrow2 %>%
  filter(demographic_measure == "Population_2014",
         party == "percent_dem") %>%
  ggplot(aes(x = percent, y = stat)) +
  geom_point(color = "#5B94C2") +
  geom_smooth() +
  xlab("Percent Voted Democratic") +
  ylab("Population Size (Normalized)") +
  ggtitle("Population by Percent Voted Democratic")
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
#plot top demographics
final_narrow2 %>%
  filter(demographic_measure == "Percent_White_Alone_not_Hispanic_or_Latino" |
         demographic_measure == "Percent_Black_or_African_American" |
         demographic_measure == "Housing_units_in_multi_unit_structures" |
         demographic_measure == "Per_capita_money_income_in_past_12_months" |
         demographic_measure == "Median_value_of_owner_occupied_housing_units") %>%
  ggplot(aes(x = percent, y = stat, color = party)) +
  geom_smooth() +
  facet_wrap(~demographic_measure, ncol = 2) +
  ggtitle("Voting Trends by Top Demographics") +
  xlab("Percent of Vote") +
  ylab("Normalized Demographic Value") +
  theme(strip.text = element_text(size=11), legend.position = c(0.75, 0.15)) +
  scale_colour_manual(labels = c("Democrat", "Republican"), values = c("#5B94C2",
                                                                       "#E54445")) +
  scale_x_continuous(limits = c(0, 1), breaks = c(0, 0.25, 0.5, 0.75, 1)) +
  scale_y_continuous(limits = c(0, 1), breaks = c(0, 0.25, 0.5, 0.75, 1))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```



Conclusion

In conclusion, KNN was the best performing model to use when predicting election results by county. A full write-up of the results can be found on Ian Jeffries' github page.