Predicting Squirrel Movements in Central Park Using Statistical Learning Models

Hassan Shah

12/8/2022

Predicting Squirrel Movements in Central Park Using Statistical Learning Models

Setup:

Libraries we will need:

```r
library(geosphere)
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.
```

```r
library(gbm)
```

```
## Loaded gbm 2.1.8.1
```

Reading in squirrel data:

```r
df <- read.csv("/Users/hassanshah/Downloads/squirrel.csv", na.strings = c("","?"))
head(df)
```

```
##           X        Y Unique.Squirrel.ID Hectare Shift     Date
## 1 -73.95613 40.79408     37F-PM-1014-03     37F    PM 10142018
## 2 -73.96886 40.78378     21B-AM-1019-04     21B    AM 10192018
## 3 -73.97428 40.77553     11B-PM-1014-08     11B    PM 10142018
## 4 -73.95964 40.79031     32E-PM-1017-14     32E    PM 10172018
## 5 -73.97027 40.77621     13E-AM-1017-05     13E    AM 10172018
## 6 -73.96836 40.77259     11H-AM-1010-03     11H    AM 10102018
##   Hectare.Squirrel.Number   Age Primary.Fur.Color Highlight.Fur.Color
## 1                       3  <NA>              <NA>                <NA>
## 2                       4  <NA>              <NA>                <NA>
## 3                       8  <NA>              Gray                <NA>
## 4                      14 Adult              Gray                <NA>
## 5                       5 Adult              Gray            Cinnamon
## 6                       3 Adult          Cinnamon               White
```

```
##   Combination.of.Primary.and.Highlight.Color
## 1                                          +
## 2                                          +
## 3                                      Gray+
## 4                                      Gray+
## 5                              Gray+Cinnamon
## 6                             Cinnamon+White
##                                                                              Color.notes
## 1                                                                                   <NA>
## 2                                                                                   <NA>
## 3                                                                                   <NA>
## 4 Nothing selected as Primary. Gray selected as Highlights. Made executive adjustments.
## 5                                                                                   <NA>
## 6                                                                                   <NA>
##       Location Above.Ground.Sighter.Measurement Specific.Location Running
## 1         <NA>                             <NA>              <NA>   false
## 2         <NA>                             <NA>              <NA>   false
## 3 Above Ground                               10              <NA>   false
## 4         <NA>                             <NA>              <NA>   false
## 5 Above Ground                             <NA>     on tree stump   false
## 6         <NA>                             <NA>              <NA>   false
##   Chasing Climbing Eating Foraging Other.Activities  Kuks Quaas Moans
## 1   false    false  false    false             <NA> false false false
## 2   false    false  false    false             <NA> false false false
## 3    true    false  false    false             <NA> false false false
## 4   false    false   true     true             <NA> false false false
## 5   false    false  false     true             <NA> false false false
## 6   false    false  false     true             <NA> false false false
##   Tail.flags Tail.twitches Approaches Indifferent Runs.from Other.Interactions
## 1      false         false      false       false     false               <NA>
## 2      false         false      false       false     false               <NA>
## 3      false         false      false       false     false               <NA>
## 4      false         false      false       false      true               <NA>
## 5      false         false      false       false     false               <NA>
## 6      false          true      false        true     false               <NA>
##                                       Lat.Long
## 1 POINT (-73.9561344937861 40.7940823884086)
## 2 POINT (-73.9688574691102 40.7837825208444)
## 3 POINT (-73.97428114848522 40.775533619083)
## 4 POINT (-73.9596413903948 40.7903128889029)
## 5 POINT (-73.9702676472613 40.7762126854894)
## 6 POINT (-73.9683613516225 40.7725908847499)
```

Looking at how many rows (squirrel sightings) and columns(features) we are working with:

```r
dim(df)
```

```
## [1] 3023   31
```

Preprocessing:

The raw data has many features that require preprocessing due to issues like null values, nominal data, relevance, etc…

Finding number of null values in each columns to see if some features have many missing values:

```r
null_counts <- vector("numeric", ncol(df))
for (i in 1:ncol(df)) {
  null_counts[i] <- sum(is.na(df[[i]]))
}
col_names <- names(df)
null_counts_named <- setNames(null_counts, col_names)
print(null_counts_named)
```

```
##                                            X
##                                            0
##                                            Y
##                                            0
##                           Unique.Squirrel.ID
##                                            0
##                                      Hectare
##                                            0
##                                        Shift
##                                            0
##                                         Date
##                                            0
##                      Hectare.Squirrel.Number
##                                            0
##                                          Age
##                                          125
##                            Primary.Fur.Color
##                                           55
##                          Highlight.Fur.Color
##                                         1086
## Combination.of.Primary.and.Highlight.Color
##                                            0
##                                  Color.notes
##                                         2841
##                                     Location
##                                           64
##             Above.Ground.Sighter.Measurement
##                                          114
##                            Specific.Location
##                                         2547
##                                      Running
##                                            0
##                                      Chasing
##                                            0
##                                     Climbing
##                                            0
##                                       Eating
##                                            0
##                                     Foraging
##                                            0
##                             Other.Activities
##                                         2586
##                                         Kuks
##                                            0
##                                        Quaas
```

```
##                                       0
##                                   Moans
##                                       0
##                              Tail.flags
##                                       0
##                           Tail.twitches
##                                       0
##                              Approaches
##                                       0
##                             Indifferent
##                                       0
##                               Runs.from
##                                       0
##                      Other.Interactions
##                                    2783
##                                Lat.Long
##                                       0
```

We will remove columns with many null values. For our purposes, we will remove columns if more than 1000 values are missing.

Finding features with more than 1000 values missing.

```
indices <- which(null_counts_named > 1000)
missing_cols <- names(null_counts_named[indices])
missing_cols
```

```
## [1] "Highlight.Fur.Color" "Color.notes"         "Specific.Location"
## [4] "Other.Activities"    "Other.Interactions"
```

Now, we will create a dataset without these features:

```
df_new <- df[ , !(names(df) %in% missing_cols)]
dim(df_new)
```

```
## [1] 3023   26
```

Next, we need to find a way to determine if some features are irrelevant or have textual/nominal data that might be difficult to impute. This requires looking into the features and understanding which features are viable. I decided to limit each feature to a maximum of five categories as all of the features with more than five categories are textual and computationally expensive to use as a feature in a model.

Determining the number of different values in each column:

```
val_counts_named <- sapply(df_new, function(x) length(unique(x)))
val_counts_named
```

```
##                                       X
##                                    3023
##                                       Y
##                                    3023
##                      Unique.Squirrel.ID
```

```
##                                         3018
##                                      Hectare
##                                          339
##                                        Shift
##                                            2
##                                         Date
##                                           11
##                      Hectare.Squirrel.Number
##                                           23
##                                          Age
##                                            3
##                            Primary.Fur.Color
##                                            4
## Combination.of.Primary.and.Highlight.Color
##                                           22
##                                     Location
##                                            3
##                Above.Ground.Sighter.Measurement
##                                           42
##                                      Running
##                                            2
##                                      Chasing
##                                            2
##                                     Climbing
##                                            2
##                                       Eating
##                                            2
##                                     Foraging
##                                            2
##                                         Kuks
##                                            2
##                                        Quaas
##                                            2
##                                        Moans
##                                            2
##                                   Tail.flags
##                                            2
##                                Tail.twitches
##                                            2
##                                   Approaches
##                                            2
##                                  Indifferent
##                                            2
##                                    Runs.from
##                                            2
##                                     Lat.Long
##                                         3023
```

Based on an analysis of the dataset, the columns with greater than five categories are textual notes or redundant with other columns. We will remove those columns:

```r
indices <- which(val_counts_named > 5)
val_cols <- names(val_counts_named[indices])
df_new <- df_new[ , !(names(df_new) %in% val_cols)]
```

```r
dim(df_new)
```

```
## [1] 3023   17
```

Now we are going to remove rows with NA values while keeping the original index so that we can still match the output:

```r
complete_rows <- complete.cases(df_new)
df_comp <- subset(df_new, complete_rows)
feat_idx <- row.names(df_comp)
dim(df_comp)
```

```
## [1] 2822   17
```

```r
head(df_comp)
```

```
##    Shift   Age Primary.Fur.Color     Location Running Chasing Climbing Eating
## 5     AM Adult              Gray Above Ground   false   false    false  false
## 7     AM Adult              Gray Ground Plane   false   false    false  false
## 8     AM Adult              Gray Ground Plane   false   false    false  false
## 9     PM Adult              Gray Ground Plane   false   false    false  false
## 10    AM Adult              Gray Above Ground   false   false     true  false
## 11    PM Adult              Gray Ground Plane   false   false    false  false
##    Foraging  Kuks Quaas Moans Tail.flags Tail.twitches Approaches Indifferent
## 5      true false false false      false         false      false       false
## 7      true false false false      false         false      false       false
## 8      true false false false      false         false      false        true
## 9     false false false false       true          true      false       false
## 10    false false false false      false         false      false        true
## 11     true false false false      false         false      false        true
##    Runs.from
## 5      false
## 7      false
## 8      false
## 9      false
## 10     false
## 11     false
```

After preprocessing, all features have either 2 or 3 categories:

```r
val_counts_named <- sapply(df_comp, function(x) length(unique(x)))
val_counts_named
```

```
##             Shift               Age Primary.Fur.Color          Location
##                 2                 2                 3                 2
##           Running           Chasing          Climbing            Eating
##                 2                 2                 2                 2
##          Foraging              Kuks             Quaas             Moans
##                 2                 2                 2                 2
##        Tail.flags     Tail.twitches        Approaches       Indifferent
##                 2                 2                 2                 2
##         Runs.from
##                 2
```

We are going to convert all the categorical data into numeric for the purposes of training our models. We will manually assign values based on the column and our knowledge of the dataset.

Assigning values:

```
df_comp$Shift <- replace(df_comp$Shift, df_comp$Shift == "AM", 0)
df_comp$Shift <- replace(df_comp$Shift, df_comp$Shift == "PM", 1)
df_comp$Age <- replace(df_comp$Age, df_comp$Age == "Adult", 0)
df_comp$Age <- replace(df_comp$Age, df_comp$Age == "Juvenile", 1)
df_comp$Primary.Fur.Color <- replace(df_comp$Primary.Fur.Color, df_comp$Primary.Fur.Color == "Gray", 0)
df_comp$Primary.Fur.Color <- replace(df_comp$Primary.Fur.Color, df_comp$Primary.Fur.Color == "Cinnamon"
df_comp$Primary.Fur.Color <- replace(df_comp$Primary.Fur.Color, df_comp$Primary.Fur.Color == "Black", 2
df_comp$Location <- replace(df_comp$Location, df_comp$Location == "Ground Plane", 0)
df_comp$Location <- replace(df_comp$Location, df_comp$Location == "Above Ground", 1)
df_comp <- as.data.frame(lapply(df_comp, function(x) replace(x, x == "false", 0)))
df_comp <- as.data.frame(lapply(df_comp, function(x) replace(x, x == "true", 1)))
```

```
head(df_comp)
```

```
##   Shift Age Primary.Fur.Color Location Running Chasing Climbing Eating Foraging
## 1     0   0                 0        1       0       0        0      0        1
## 2     0   0                 0        0       0       0        0      0        1
## 3     0   0                 0        0       0       0        0      0        1
## 4     1   0                 0        0       0       0        0      0        0
## 5     0   0                 0        1       0       0        1      0        0
## 6     1   0                 0        0       0       0        0      0        1
##   Kuks Quaas Moans Tail.flags Tail.twitches Approaches Indifferent Runs.from
## 1    0     0     0          0             0          0           0         0
## 2    0     0     0          0             0          0           0         0
## 3    0     0     0          0             0          0           1         0
## 4    0     0     0          1             1          0           0         0
## 5    0     0     0          0             0          0           1         0
## 6    0     0     0          0             0          0           1         0
```

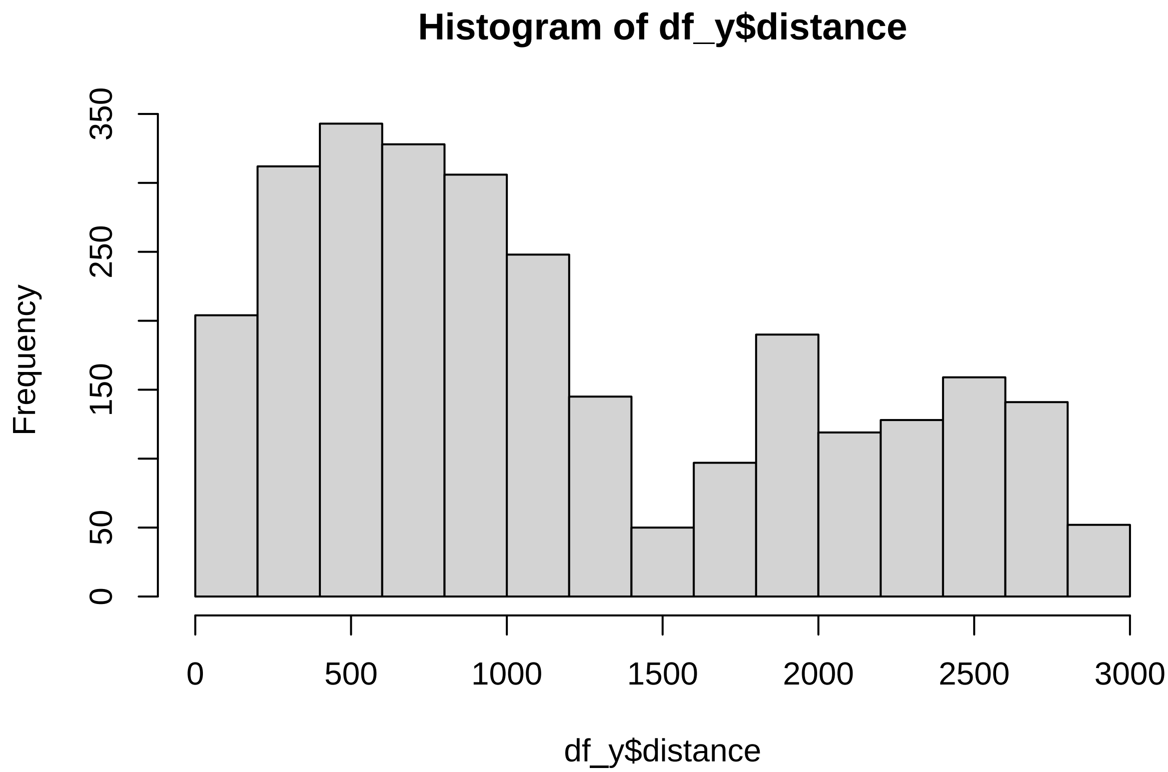
In order to create a single label to ease our efforts in predicting coordinates, we use the distHaversine() function from the geosphere package in R in order to use a single value to represent coordinates.

```
df_y <- data.frame(lat = df$X, lon = df$Y)
df_y <- df_y[feat_idx, ]
df_y$distance <- apply(df_y, 1, function(x) distHaversine(c(x[1], x[2]), c(df_y$lat[1], df_y$lon[1])))
head(df_y)
```

```
##          lat      lon  distance
## 5  -73.97027 40.77621    0.0000
## 7  -73.95412 40.79318 2328.1843
## 8  -73.95827 40.79174 2002.2996
## 9  -73.96743 40.78297  789.6257
## 10 -73.97225 40.77429  271.7242
## 11 -73.96951 40.78235  686.2951
```

Examining a distribution of the different distances:

```r
hist(df_y$distance)
```



The first pair of coordinates is assigned as north, and so we need to save the value and keep it out of model fitting in order to prevent the outlier 0 from affecting further fitting.

```r
north <- df_y[1, ]
df <- df_comp[-1, ]
df_y <- df_y[-1, ]
vec_y <- df_y$distance
df$y <- vec_y
df <- as.data.frame(lapply(df, as.numeric))
head(df)
```

```
##   Shift Age Primary.Fur.Color Location Running Chasing Climbing Eating Foraging
## 1     0   0                 0        0       0       0        0      0        1
## 2     0   0                 0        0       0       0        0      0        1
## 3     1   0                 0        0       0       0        0      0        0
## 4     0   0                 0        1       0       0        1      0        0
## 5     1   0                 0        0       0       0        0      0        1
## 6     1   0                 0        0       1       0        0      0        0
##   Kuks Quaas Moans Tail.flags Tail.twitches Approaches Indifferent Runs.from
## 1    0     0     0          0             0          0           0         0
## 2    0     0     0          0             0          0           1         0
## 3    0     0     0          1             1          0           0         0
## 4    0     0     0          0             0          0           1         0
```

```
## 5    0     0     0          0             0          0            1         0
## 6    0     0     0          0             1          0            1         0
##           y
## 1 2328.1843
## 2 2002.2996
## 3  789.6257
## 4  271.7242
## 5  686.2951
## 6  835.6689
```

Models:

Now, we will try fitting a linear regression model with all the features, a linear regression model with statistically significant (low p-value) features, a decision tree, a random forest model, and a gradient boosted model to our dataset. Our goal is to examine if any of the models predict location well, if the models rely specifically on a subset of the features, and if the model is making very generalized (biased) or variant predictions.

First, we will fit a linear regression model with all the features to our data and assess the p-values to determine which features are relevant and should stay in our model.

```r
model <- lm(y ~ ., data = df)
summary_model <- summary(model)
summary_model
```

```
##
## Call:
## lm(formula = y ~ ., data = df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1472.7  -642.7  -220.6   705.0  1856.2
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       1192.9555    48.8358  24.428  < 2e-16 ***
## Shift               19.2806    31.0780   0.620 0.535049
## Age                 13.5924    48.2817   0.282 0.778331
## Primary.Fur.Color    4.3473    31.6388   0.137 0.890722
## Location           -68.3390    45.7388  -1.494 0.135259
## Running             16.5562    37.8995   0.437 0.662258
## Chasing            -82.9869    55.0198  -1.508 0.131588
## Climbing           -25.4814    47.1577  -0.540 0.589002
## Eating            -138.2319    36.2858  -3.810 0.000142 ***
## Foraging           -41.8944    35.8219  -1.170 0.242295
## Kuks                76.1312    89.4135   0.851 0.394592
## Quaas              324.4981   125.9628   2.576 0.010042 *
## Moans             1535.6936   808.6984   1.899 0.057671 .
## Tail.flags         -22.6888    68.6764  -0.330 0.741144
## Tail.twitches      -34.6579    43.3218  -0.800 0.423773
## Approaches         -11.1569    67.2955  -0.166 0.868334
## Indifferent          0.7722    36.7746   0.021 0.983249
## Runs.from          221.9001    43.0594   5.153 2.74e-07 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 806.6 on 2803 degrees of freedom
## Multiple R-squared:  0.02613,    Adjusted R-squared:  0.02022
## F-statistic: 4.423 on 17 and 2803 DF,  p-value: 3.714e-09
```

It appears as though the R-squared is very low (almost 0), which suggests the model is having a very difficult time explaining the variance/fitting our data. However, there are some statistically significant features, so we will try creating a model with just those.

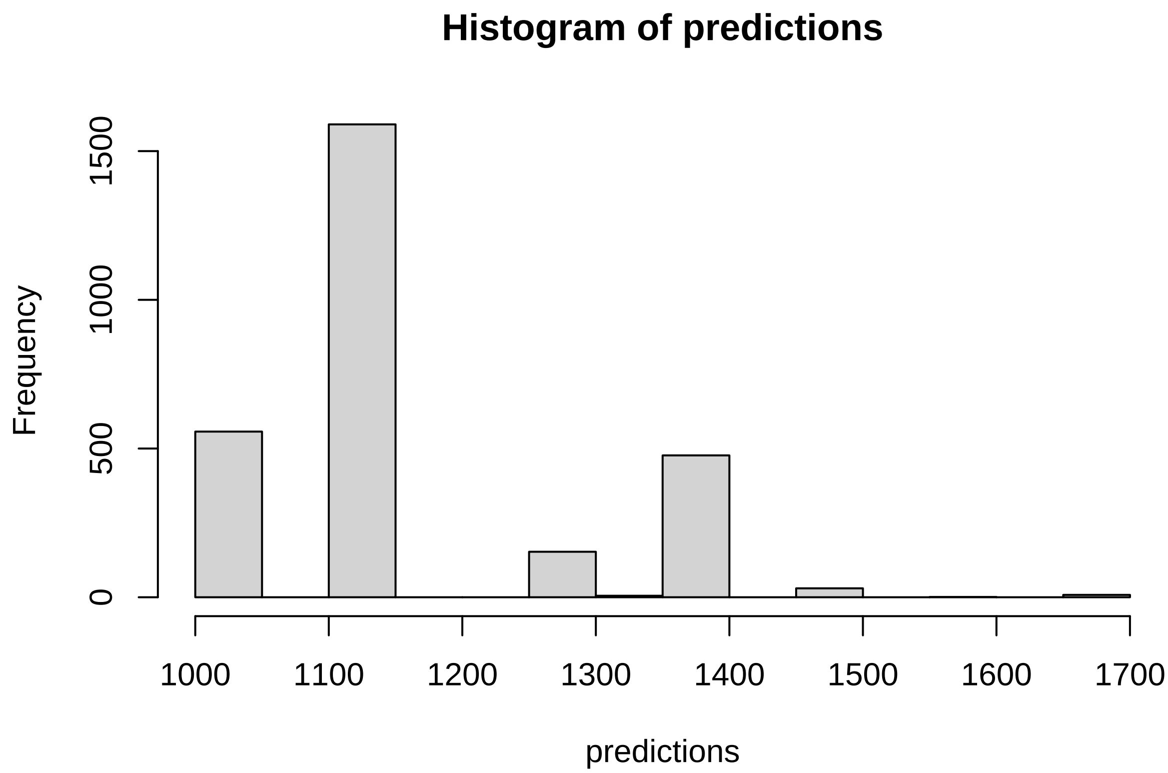
Fitting another linear model, but this time only keeping features whose p-values were less than 0.05:

```
col_names <- c("Eating", "Quaas", "Runs.from", "y")
df_sig <- df[ , col_names]
model_sig <- lm(y ~ ., data = df_sig)
summary(model_sig)
```

```
##
## Call:
## lm(formula = y ~ ., data = df_sig)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1503.5  -646.4  -226.3   713.2  1851.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1148.51      19.60  58.611  < 2e-16 ***
## Eating       -126.31      34.91  -3.618 0.000302 ***
## Quaas         312.43     122.61   2.548 0.010881 *
## Runs.from     234.74      36.28   6.471 1.15e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 806.4 on 2817 degrees of freedom
## Multiple R-squared:  0.02175,    Adjusted R-squared:  0.02071
## F-statistic: 20.88 on 3 and 2817 DF,  p-value: 2.232e-13
```

Next, we will examine a histogram of the predictions to examine the level of variance in the predicted values.

```
predictions <- predict(model_sig, df_sig)
hist(predictions)
```



Now we will use the table function:

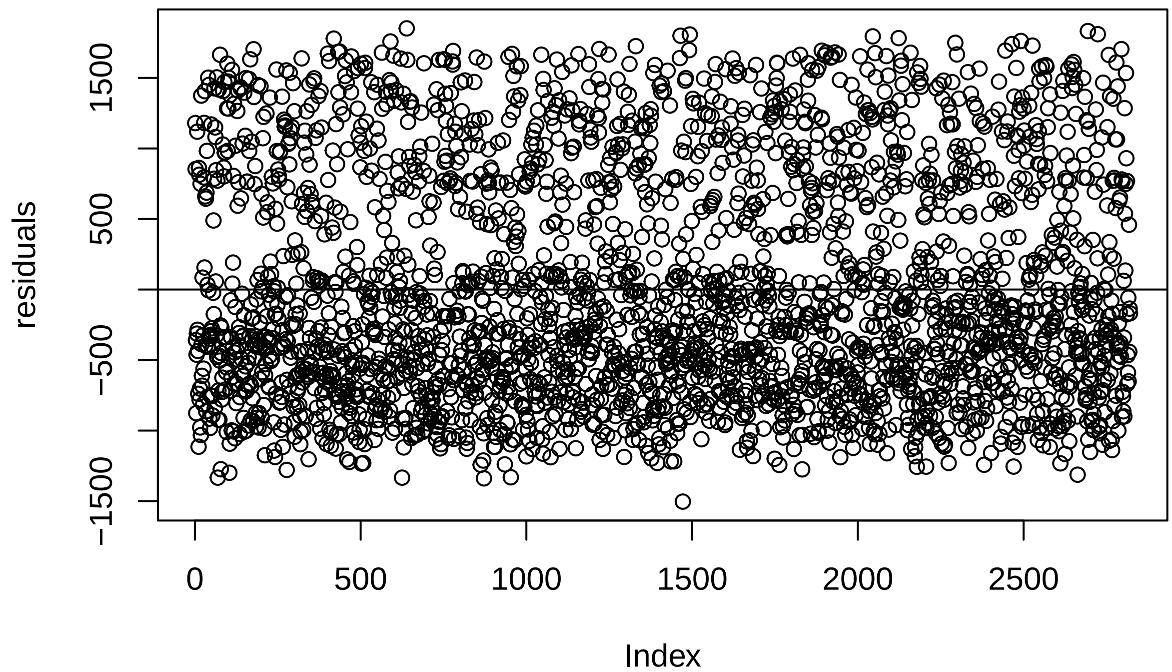
```r
table(predictions)
```

```
## predictions
##  1022.1999912314 1148.50572815006 1256.94360237567 1334.62501034323
##              557             1590              153                5
## 1383.24933929432 1460.93074726188 1569.36862148749 1695.67435840615
##              477               30                1                8
```

The linear regression model only assigned 8 different values, which suggests this method has difficult in using the features to fit the response.

Let's plot residuals to examine the difference between the predicted response and the actual response:

```r
residuals <- residuals(model_sig)
plot(residuals)
abline(h=mean(residuals))
```

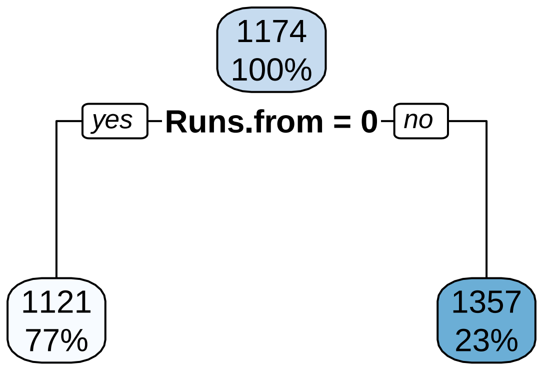


The residuals show what we would expect: a seemingly equal amount of residuals above and below the y-axis and a constant variance.

Now, we are going to try some methods with higher variance. First, we will try a decision tree.

Fitting a decision tree to all the features:

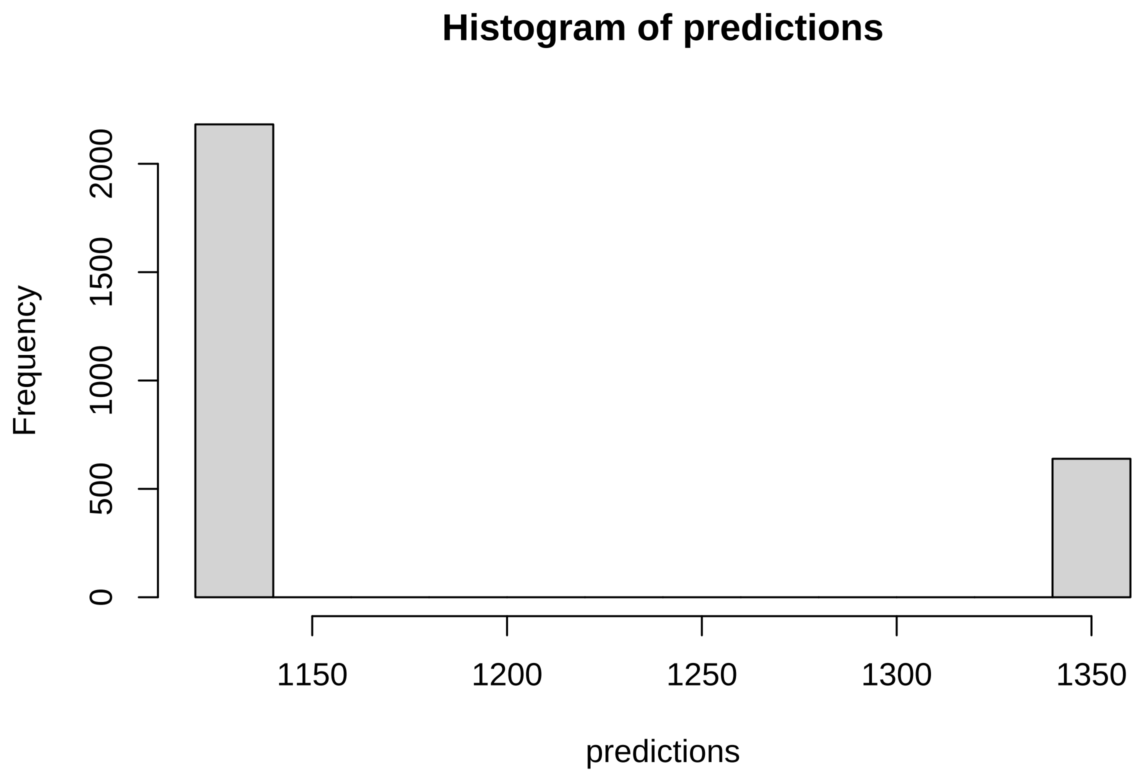
```
tree <- rpart(y ~ ., data=df)
rpart.plot(tree)
```



Interestingly, the decision tree is only using one feature to fit the model. Whether or not the squirrel runs away from the observer seems to be the only consideration for a model that is usually known to have high variance (decision tree). This means that if we only used the significant features (of which runs from is one), we create the decision tree. Additionally, the model only predicts two outputs for the entire data set, which again suggests this data set is very difficult for fitting a model.

Next, we will examine a histogram of the predictions to examine the level of variance in the predicted values.

```r
predictions <- predict(tree, df)
hist(predictions)
```



Now we will use the table function:

```
table(predictions)
```

```
## predictions
## 1120.98558677546 1357.20981141723
##             2182              639
```

And as we saw earlier, the model only predicts two different values for all distances.

Next, we are going to try a random forest model. The predictions became difficult to interpert with all the features in the data set, so we are only going to use the statistically significant features we found while fitting a linear model, and we are going to assess their importance.

Fitting a random forest model to statistically significant features and determining their relative importance:

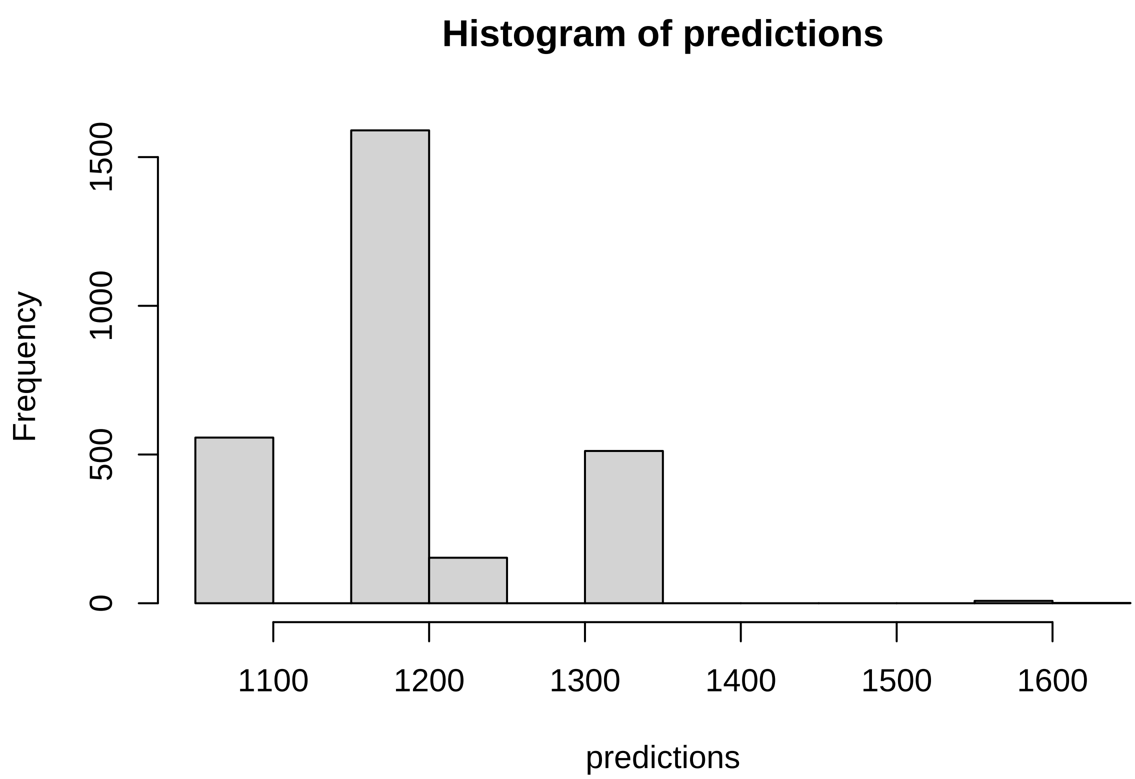
```
rand_forest <- randomForest(y ~ ., data = df_sig)
importance <- importance(rand_forest)
importance
```

```
##           IncNodePurity
## Eating          6353265
## Quaas           3529511
## Runs.from      19197479
```

As we can see, the random forest model also views the runs.from feature as the most significant, which is starting to become a pattern.

Next, we will examine a histogram of the predictions to examine the level of variance in the predicted values.

```
predictions <- predict(rand_forest, df_sig)
hist(predictions)
```



Now we will use the table function:

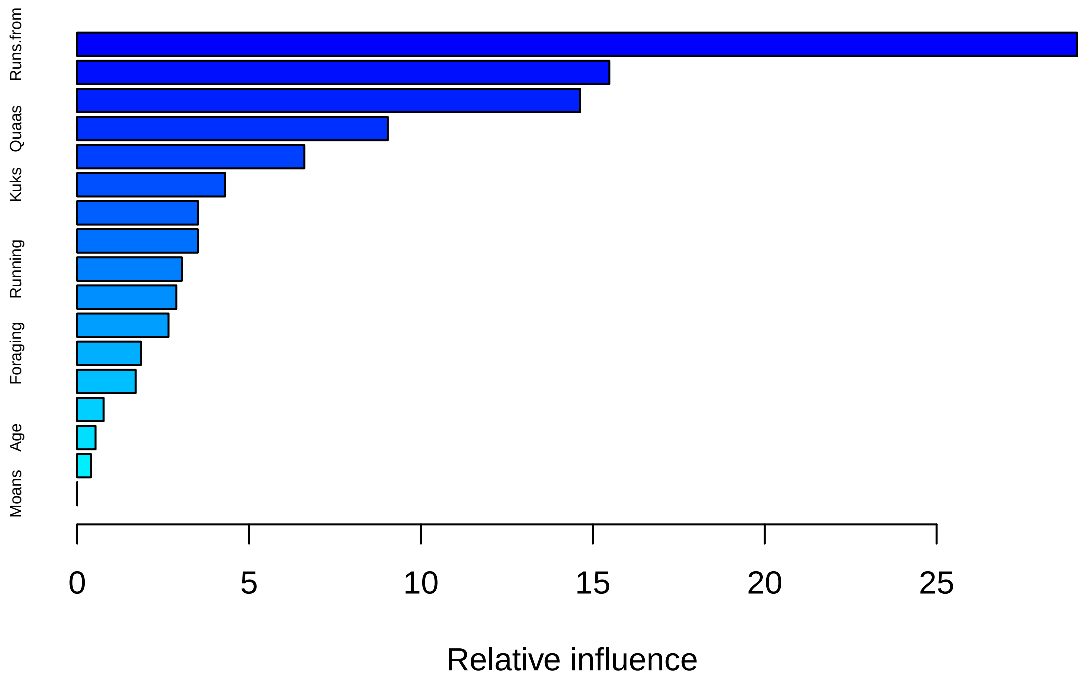
```
table(predictions)
```

```
## predictions
## 1074.80467745892 1158.25701262663 1244.72275838425 1308.24642902251
##              557             1590              153              477
## 1325.64129909451 1333.24334956745 1561.10009736523 1619.05537321963
##               30                5                8                1
```

Lastly, we are going to fit a gradient boosted model to the data set. This model is known for having high variance in its predictions, so we will see how many different predictions it outputs.

Fitting a gradient boosted model to all features and determining their relative importance:

```
gbm <- gbm(y ~ ., data = df, distribution = "gaussian")
summary(gbm, cex.names = 0.5)
```




```
##                                  var    rel.inf
## Runs.from                  Runs.from 29.0876249
## Eating                        Eating 15.4815557
## Primary.Fur.Color Primary.Fur.Color 14.6251184
## Quaas                          Quaas  9.0334399
## Location                    Location  6.6086429
## Kuks                            Kuks  4.3053690
## Shift                          Shift  3.5171661
## Chasing                      Chasing  3.5070821
## Running                      Running  3.0430848
## Climbing                    Climbing  2.8852146
## Tail.twitches          Tail.twitches  2.6569126
## Foraging                    Foraging  1.8515412
## Indifferent              Indifferent  1.7005194
## Tail.flags                Tail.flags  0.7676456
## Age                              Age  0.5338093
## Approaches                Approaches  0.3952736
## Moans                          Moans  0.0000000
```

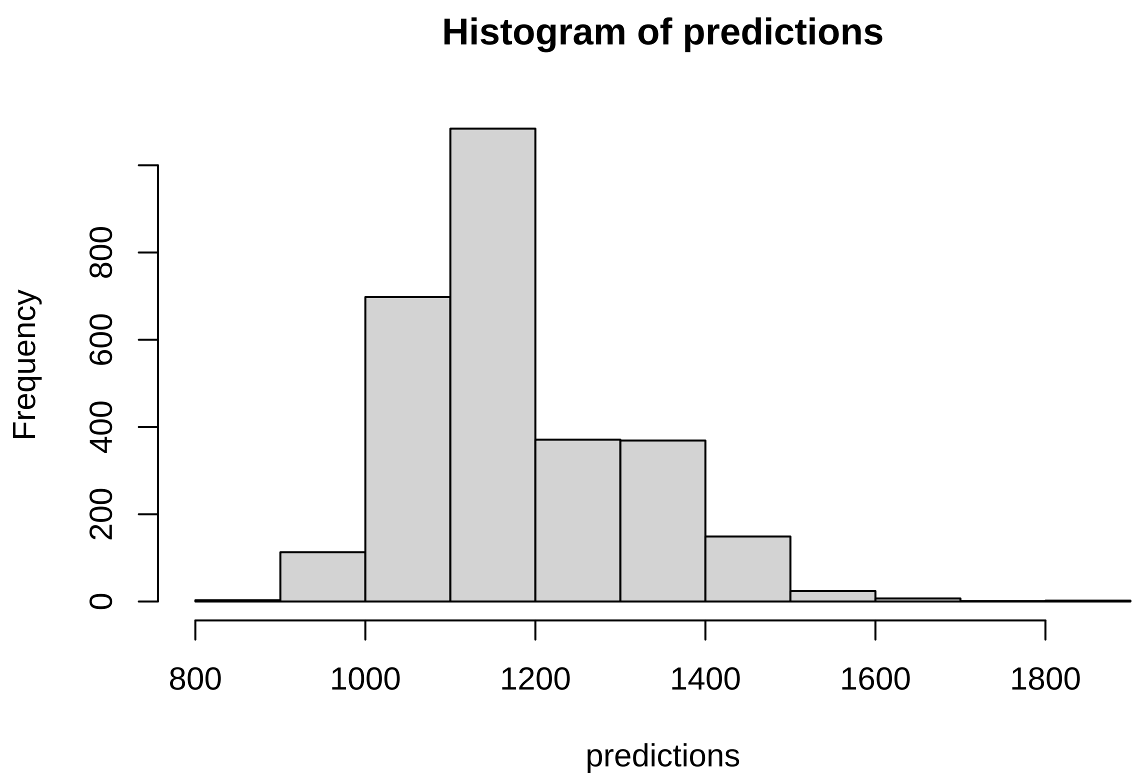
This model also suggests that runs.from is the most significant feature for fitting the data.

Next, we will examine a histogram of the predictions to examine the level of variance in the predicted values.

```
predictions <- predict(gbm, df)
```

```
## Using 100 trees...
```

```r
hist(predictions)
```



Now we will use the table function and the length function, as we are predicting many different responses for this algorithm.

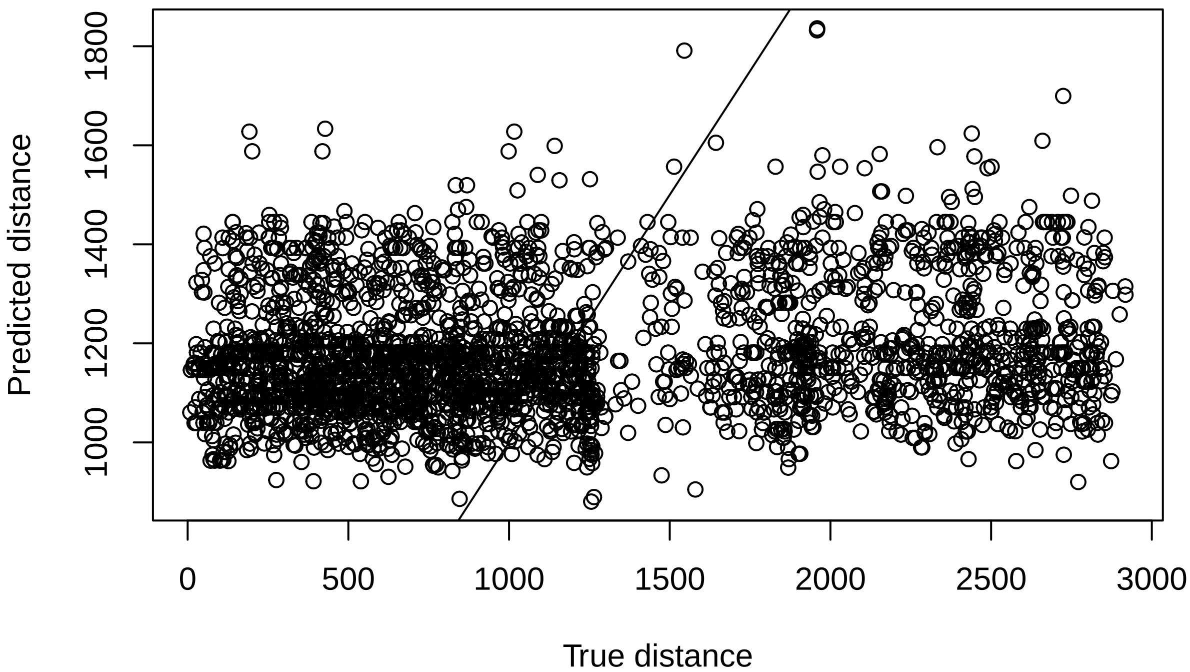
```r
length(table(predictions))
```

```
## [1] 890
```

The gradient boosted model, unlike the other models, predicted over 800 different values for the outputs, significantly lowering the bias in comparison to previous models.

Showing a plot of true vs predicted output and seeing deviation from y=x (correct response) line.

```r
vec_diff <- vec_y - predictions
plot(vec_y, predictions, xlab = "True distance", ylab = "Predicted distance")
abline(a=0, b=1)
```



This model is the only one to include many different outputs. However, as we can see, this model also seems to not do well at actually predicting distance, along with the other models. It seems like none of the models we used can fit the data well enough to predict accurately or precisely. However, we did learn that each model valued "runs.from", a category assessing whether or not the squirrel ran away from humans, as the most significant category for predicting the squirrel's distance from our bearing (i.e. north). While we cannot come out of this exercise with having fit a model that predicts successfully, we do finish with the insight that maybe the best method for predicting a squirrel's location is whether or not it is running away from bystanders at the time of sighting.