Final Project

Introduction

Can we accurately predict the happiness of a country? If so, what factors play an influential role in determining this? This project attempts to fit a model to predict nations' "happiness scores" ranked by the World Happiness Report.

The World Happiness Report

In 2012, the United Nations Sustainable Development Solutions Network published the first World Happiness Report, a ranking of countries based off their level of happiness. These rankings are obtained by asking respondents to rate their level of happiness from a scale of 0 to 10, 0 being the worst possible life and 10 being the best. Respondents' answers are then averaged depending on their country of residence and put into a ranking. Following this, each country is assigned values for six quality of life factors, such as life expectancy or Family, which are meant to display what share of a country's overall score can be explained by those factors. The idea here is that different cultures prioritize different determinants of happiness, and therefore, a factor can have different effects on happiness score depending on the country.
This project is intended to give insight into the world about how different life experiences can influence the happiness of people and their lives. This can be used for a variety of different reasons, ranging from an individual reflecting on their own life to policy-makers using it to drive particular policies for a country's well being.

Question and Purpose

So, why even attempt to predict a country's happiness if its already recorded in the World Happiness Report? Well, the methodology of the World Happiness Report follows a "bottom-up" approach in which the happiness score is found first and then the contributing factors are found retroactively. However, I will be attempting a "top-down" approach in which I will take preexisting socioeconomic factors and attempt to predict a country's happiness from there.

Additionally, a common critique of the World Happiness Report is that judging how much a variable such as "Family" or "Freedom" affects one's happiness is difficult as there is no numerical value for these categories. Furthermore, the documentation for determining contributing factors in the World Happiness Report may be lengthy to research. As a result, I believe looking at raw statistics from a model may give a better idea of factors that contribute to a country's happiness.

Loading Data and Packages

Here are all the packages utilized to create the model.

```
library(tidyverse)
library(tidymodels)
library(corrplot)
library(ggplot2)
```

```
library(dplyr)
library(janitor)
library(Hmisc)
library(readxl)
library(patchwork)
library(kknn)
library(vip)
library(mice)
library(yardstick)
```

This model was created by combining two datasets. The first dataset, data1, is a compilation of World Development Indicators from the World Bank (https://datacatalog.worldbank.org/search/dataset/0037712/World-Development-Indicators). The second dataset contains the world happiness score from countries in 2017(https://worldhappiness.report/ed/2017). 2017 data was selected because data from more recent years tends to be incomplete.

```
data1 <- read_excel("C:/Users/foamy/Downloads/Final Project/DATA/DATASET1.xlsx")
data2 <- read.csv(file = "C:/Users/foamy/Downloads/Final Project/DATA/DATASET2.csv", header=TRUE, string
```

Processing Data

Combining Data

The following are the steps involved in creating the final dataset:

1. Filter data1 with countries in data2.

```
countries <- data2$Country
data1 <- filter(data1, data1$'Country Name' %in% countries)
```

2. Subset data1 columns to utilize

```
data1_variables <- c("Country Name",'Indicator Name', '2017')
data1 <- data1[data1_variables]
```

3. Filter important values in 'Indicator Name' column and pivot wider so that values of 'Indicator Name' are now columns (Plus cleaning the data along the way!)

```
variables_focus <- c("Access to electricity (% of population)","Government expenditure on education, to
#Variables were chosen to encompass a broad amount of socioeconomic factors with a relatively small amo
data1_edit <- data1 %>% filter(data1$'Indicator Name' %in% variables_focus)%>%clean_names()%>%pivot_wid
```

4. Renaming country to country_name, then filter data2 with countries in data1 (and more cleaning!!)

```
countries2<- unique(data1_edit$country_name)
data2_clean <- clean_names(data2)
data2_edit <- data2_clean[c('country','happiness_score')] %>% filter(data2_clean$'country' %in% countri
#cleaning done before and after merging so all names match up
```

5. Now that both country_name columns are matching, datasets can be merged

```
data_final <- merge(data1_edit,data2_edit,by="country_name")
```

6. Variables are renamed for ease of convenience (even more cleaning!!!)

```
oldnames = colnames(data_final)
newnames = c("country_name","elec_access","age_dep_ratio","co2_emissions","gdp","education_expend","life
data_final <- data_final %>% rename_at(vars(oldnames), ~ newnames) %>% clean_names()
```

Splitting and Folding Data

The data was split with 70% in the training data and 30% in the testing data with happiness_score being the strata to ensure it is similar between datasets.

```
set.seed(1008) #This is my birthday!!!
happiness_split<- initial_split(data_final,prop=.80,strata = happiness_score)
happiness_train<-training(happiness_split)
happiness_test<-testing(happiness_split)
```

This split conveniently makes it so that our training data has 110 samples, allowing us to evenly divide our set into 10 folds. Because our sample is relatively small, a larger k may help reduce bias while keeping the variance manageable. Also, splitting was performed before EDA because I wish to prevent data leakage of our model. With a small dataset, this could greatly affect its performance especially because too much information from our testing set would be revealed.

```
happiness_folds <- vfold_cv(happiness_train,v=10)
#NOTE: FOLDS ARE NOT ACTUALLY EXECUTED HERE BECAUSE THEY SHOULD BE CREATED AFTER IMPUTATION OF MISSING
#I felt it was more appropriate to talk about folds here, but this code will be executed right after mi
```

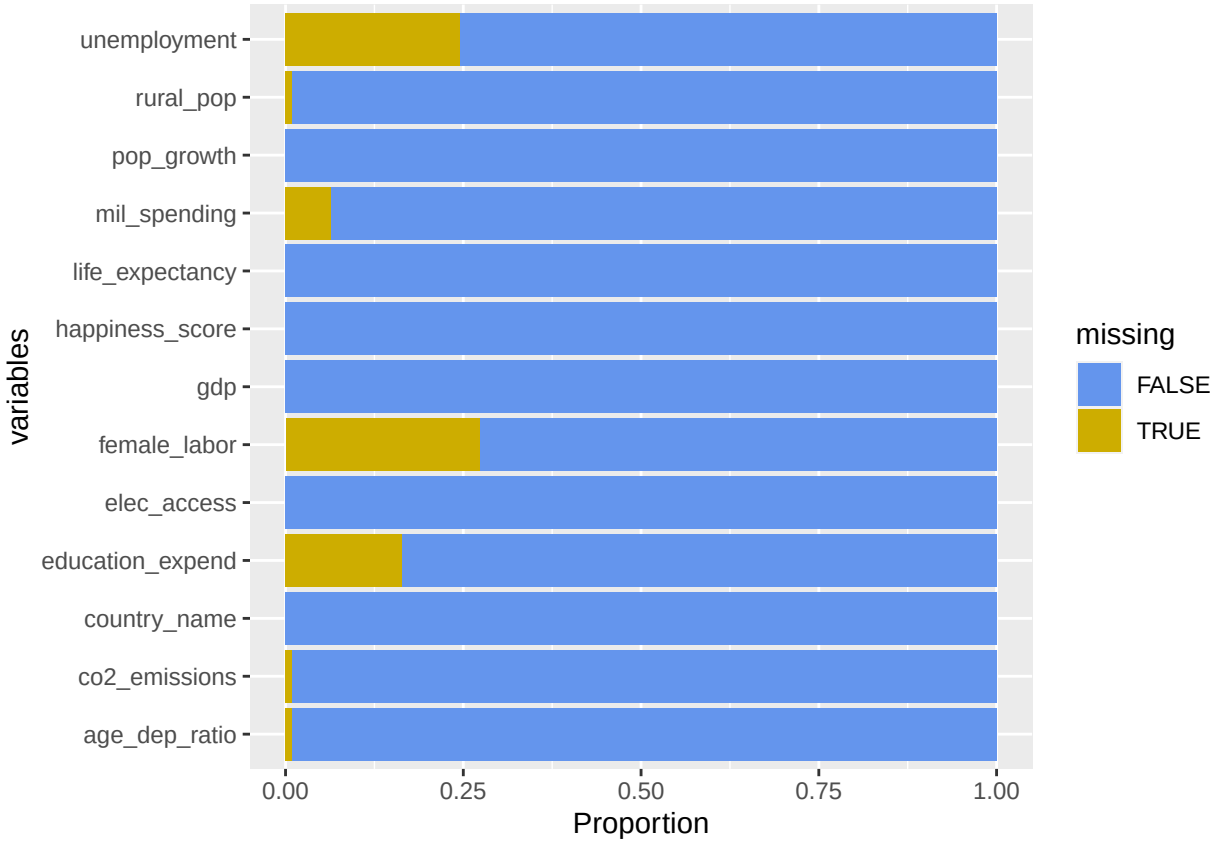
Now we can finally begin!

Exploratory Data Analysis

Missing Data

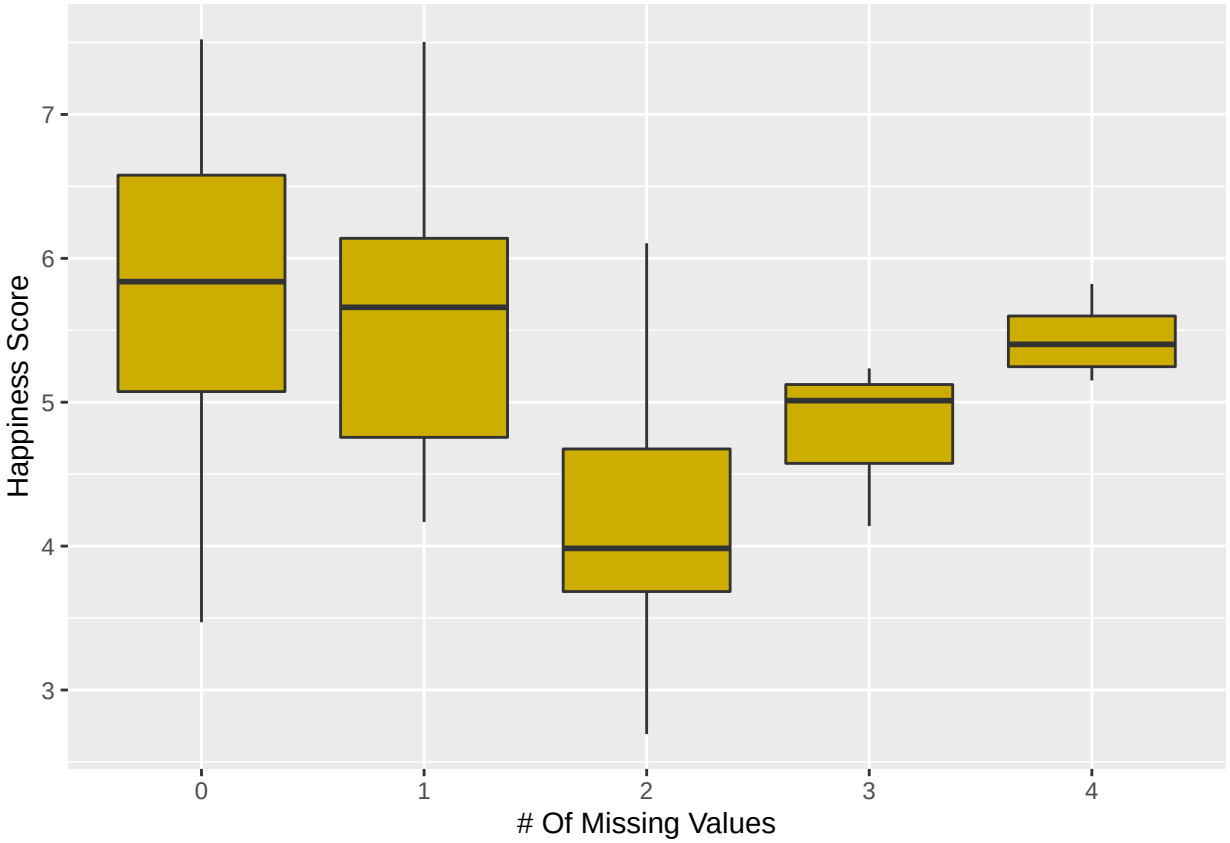
First, we should graph the missing data in our training set. This will give us a good idea of the percentage of missing data and what to do with it.

```
happiness_train  %>%
  summarise_all(list(~is.na(.)))%>%
  pivot_longer(everything(),
               names_to = "variables", values_to="missing") %>%
  count(variables, missing) %>%
  ggplot(aes(y=variables,x=n,fill=missing))+
  geom_col(position = "fill")+
  labs(x="Proportion")+
  scale_fill_manual(values=c("cornflowerblue","gold3"))
```



Looking at the variables, it makes sense that some will have more missing values. GDP is relatively easy to measure or estimate within a country unlike the percentage of female workforce or education expenditure. We could get rid of all the missing values, but our dataset is too small to afford that, Let's look further into the missing data.

```
happiness_train%>%
  mutate(as.factor(rowSums(is.na(happiness_train))))%>%
  ggplot(aes(x=as.factor(rowSums(is.na(happiness_train))),y=happiness_score))+
  geom_boxplot(fill='gold3')+
  labs( x ='# Of Missing Values', y = "Happiness Score")
```



This graph displays the five number summary of countries depending on their number of missing values. Observing it, we can see that the median happiness score varies significantly depending on the number of missing values, and therefore, imputing with median and mean may lead to misleading numbers. For example, assuming happiness score and unemployment are inversely correlated, missing unemployment values are likely to be higher than the average of the dataset. This is because there is a relationship between the missing data and other variables, known as MAR (Missing At Random). Additionally, as you will see later in this project, the distribution of variables are rather skewed.

In order to combat this, I downloaded the MICE package and utilized "predictive mean matching" to impute my missing values. This process involves finding observations with similar values to the observation with the missing point and randomly selecting a value to impute from these observations. More documentation may be found here: https://www.rdocumentation.org/packages/mice/versions/2.30/topics/mice.

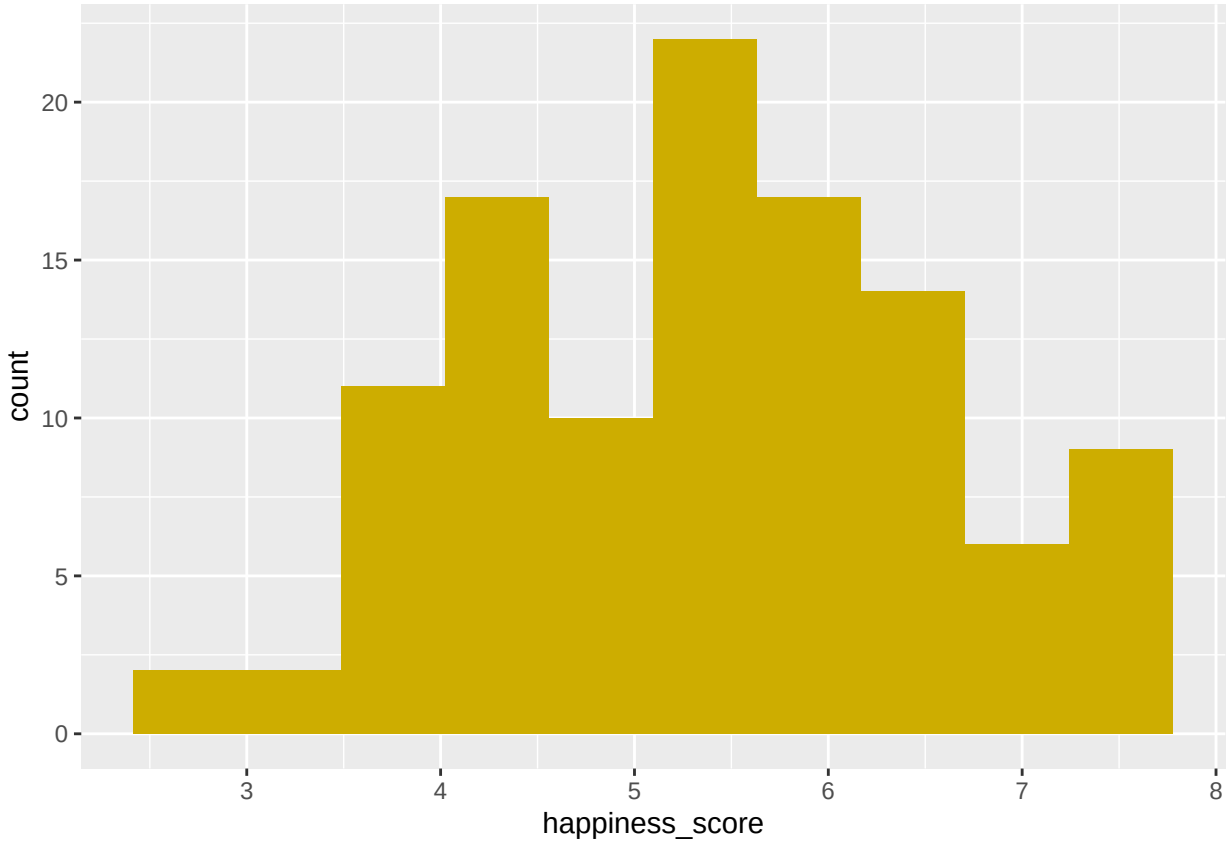
Similar to how I will be performing EDA after the test/train split, it is important that imputation also occurs after the test/train split. This can, once again, help prevent data leakage so that the test set does not influence the creation of my models.

```
imputed_Data <- mice(happiness_train, m=1, maxit = 50, method = 'pmm', seed = 1008)
#m refers to the amount of imputed datasets. because I will only be utilizing one dataset, this value w
happiness_train <- complete(imputed_Data,1)
imputed_Data2 <- mice(happiness_test, m=1, maxit = 50, method = 'pmm', seed = 1008)
happiness_test <- complete(imputed_Data2,1)
happiness_folds <- vfold_cv(happiness_train,v=10)
#HERE IS WHEN FOLDS ARE ACTUALLY EXECUTED
```

Histogram of Variables

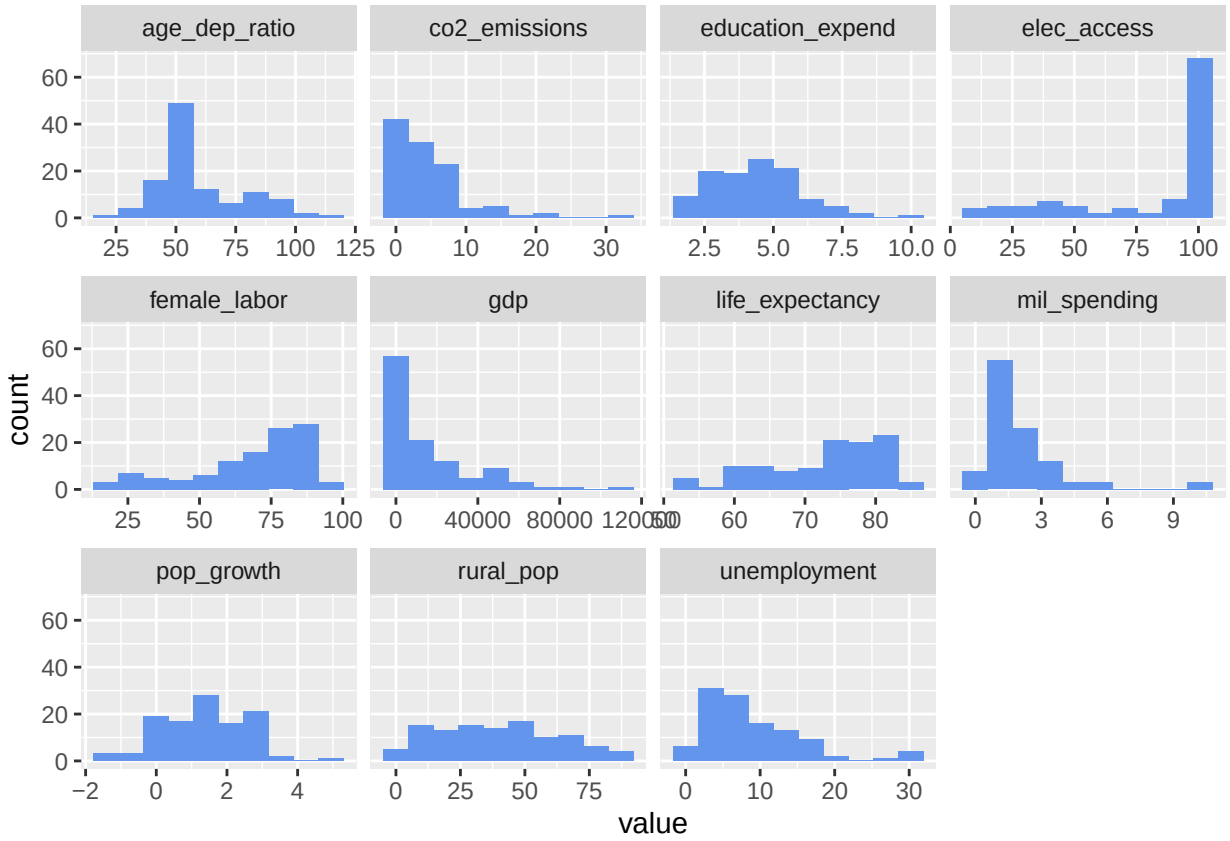
Now, we will observe the spread of variables through histograms.

```
happiness_train %>%
  ggplot(aes(happiness_score))+
  geom_histogram(bins = 10,fill=c("gold3"))
```



This histogram shows us that the distribution of happiness scores within our training set are relatively normal with a mode of values from around 5-5.5. The count from the sides of the graph are also small with no happiness scores going under 2 or above 8.

```
happiness_train %>%
  pivot_longer(!c(country_name,happiness_score), names_to = "data") %>%
  dplyr::select(-happiness_score)%>%
  ggplot(aes(value))+
  geom_histogram(bins = 10,fill=c("cornflowerblue")) +
  facet_wrap(~data, scales = 'free_x')
```



Further examining the distribution of variables, you can see that certain graphs are heavily skewed. This is especially true for gdp and elec_access, both of which I believe to be important variables in determining a country's happiness. As previously stated, this would make imputing missing values with mean potentially misleading as countries would be assigned values not indicative of their true value.

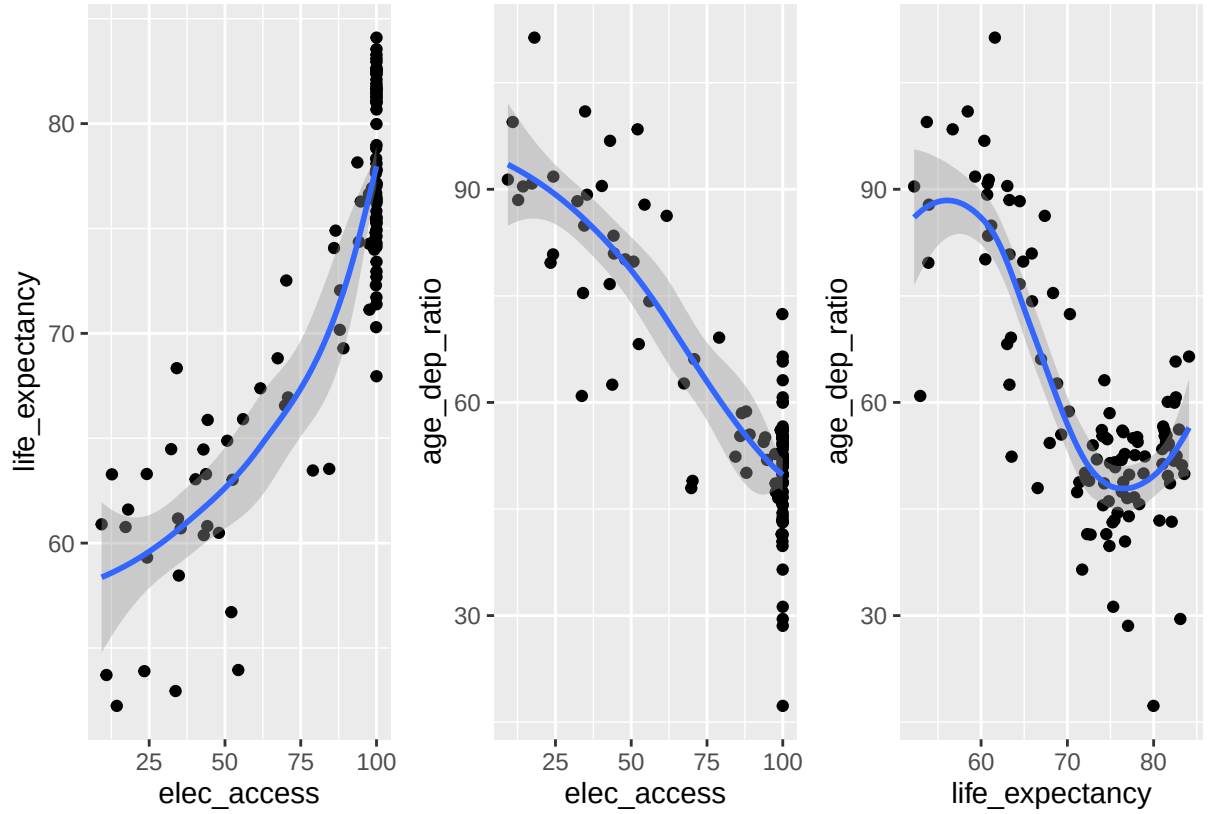
Correlation Analysis

Given that all of the variables in the dataset relate to some social/economic aspect of a country, it is likely there is some correlation between them.

```
happiness_train %>%
  dplyr::select(-country_name,-happiness_score) %>%
  cor() %>%
  corrplot(order = "hclust", addrect = 3)
```

There seems to be significant interactions between life_expectancy x elec_access, life_expectancy x age_dep_ratio, and elec_access x age_dep_ratio, all of which seem to have absolute values of correlation > 0.7. Perhaps graphing them will give a better representation.

```
graph1 <- happiness_train %>%
  ggplot(aes(x=elec_access,y=life_expectancy))+geom_point()+geom_smooth(method=loess)
graph2 <- happiness_train %>%
  ggplot(aes(x=elec_access,y=age_dep_ratio))+geom_point()+geom_smooth(method=loess)
graph3 <- happiness_train %>%
  ggplot(aes(x=life_expectancy,y=age_dep_ratio))+geom_point()+geom_smooth(method=loess)
graph1 + graph2 + graph3
```

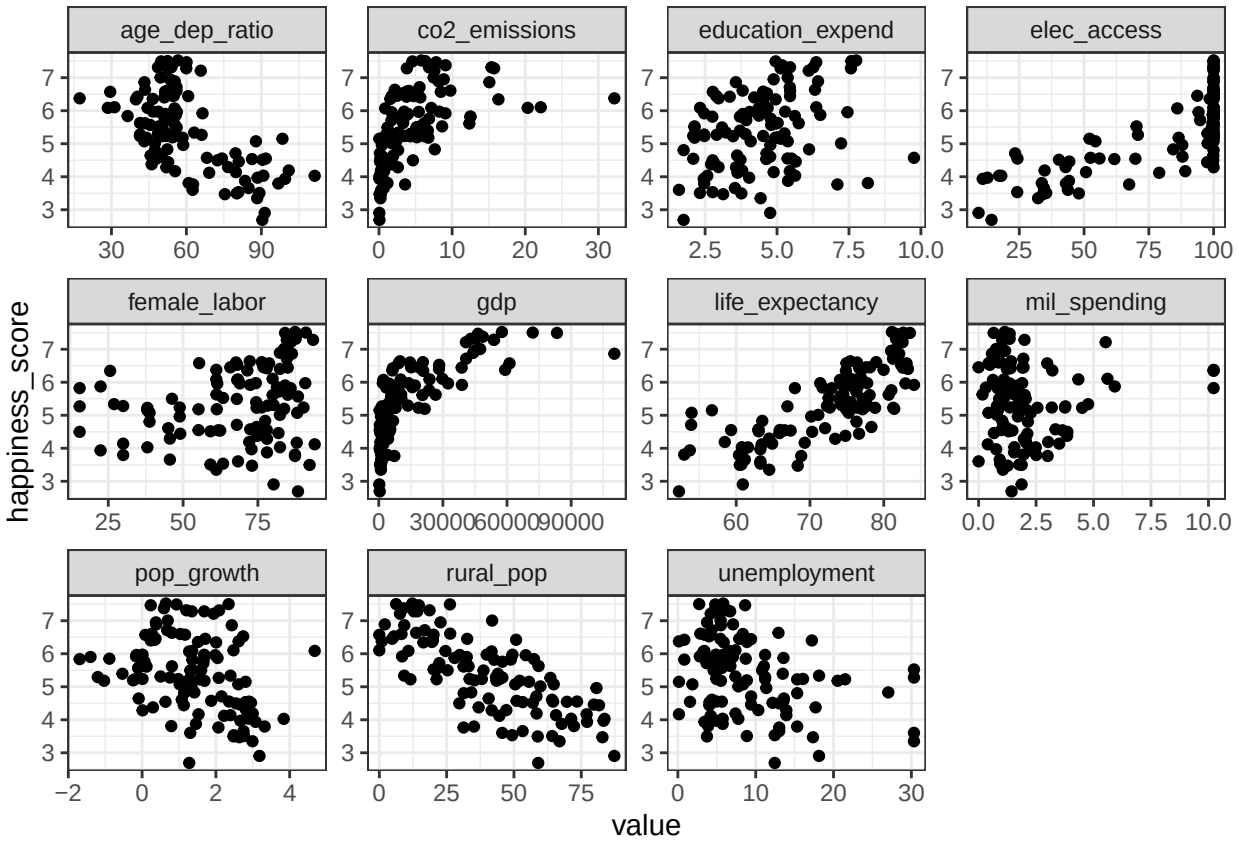


There does seem to be a high correlation between all of these variables, a positive correlation in life_expectancy x elec_access, and a negative correlation between age_dep_ratio x elec_access and age_dep ratio x life_expectancy. This is important to take note for fitting models that require independence of observations since interaction terms between variables need to be created.

Happiness Score vs. Variables

Finally, we can see the relationship between happiness score and other variables.

```
happiness_train %>%
  gather(-country_name, -happiness_score, key = "var", value = "value") %>%
  ggplot(aes(x = value, y = happiness_score)) +
    geom_point() +
    facet_wrap(~ var, scales = "free") +
    theme_bw()
```



Unsurprisingly, most variables in the dataset seem to have a fairly strong correlation with happiness score. However, not all these relationships seem to be linear. For example, co2_emissions and gdp seem to follow a logarithmic relationship and mil_spending follows a U-shaped pattern. This may be an issue for a linear regression model, so we will be doing some transformations later to fit linearity assumptions.

Model Building

For this project, I will be fitting 4 models classes and choosing the highest performing one. These include:

1. Ridge Regression
2. Random Forest
3. Boosted Tress
4. Nearest Neighbors

Ridge Regression

Ridge regression follows the same assumptions as linear regression, and as seen from the EDA, our current data fails to follow both the independence and linearity requirements. As such, when creating the recipe, a column of 2nd degree basis expansion was added for mil_spending, and co2_emissions and gdp were log transformed. The recipe also specifies interaction terms with the highly correlated variables found in our EDA graph. Only three interaction terms were created in order to reduce degrees of freedom. Finally, the variables were normalized to have a mean of 0 and standard deviation of 1 since it helps with reading the data.

```
happiness_recipe <- recipe(happiness_score ~ elec_access+age_dep_ratio+co2_emissions+gdp+education_exper
#polynomial expansion on military spending because it has a u-shape when graphed against happiness scor
  step_poly(mil_spending, degree = 2)  %>% #polynomial expansion
  step_log(co2_emissions, base = 10) %>% #log base 10 trasnformation
  step_log(gdp, base = 10) %>%
  step_interact(terms = ~ life_expectancy:elec_access+life_expectancy:age_dep_ratio+elec_access:age_dep_
  step_center(all_predictors()) %>% #center and scale all predictors
  step_scale(all_predictors())
```

Here, I tune the penalty. Because there is no dedicated ridge functions in parsnip, I utilized linear_reg() and set mixture to zero. The mode was set to regression with a glmnet engine. Then, the model and recipe created above were added to the workflow.

```
ridge_spec <-
  linear_reg(penalty = tune(), mixture = 0) %>%
  set_mode("regression") %>%
  set_engine("glmnet")

ridge_workflow <- workflow() %>%
  add_recipe(happiness_recipe) %>%
  add_model(ridge_spec)
```

Next, I set the range of penalty from -2 to 2 in my grid. These values seem to encapsulate a significant change in performance. Because I am only tuning one hyperparameter here, I set the amount of levels to 10.

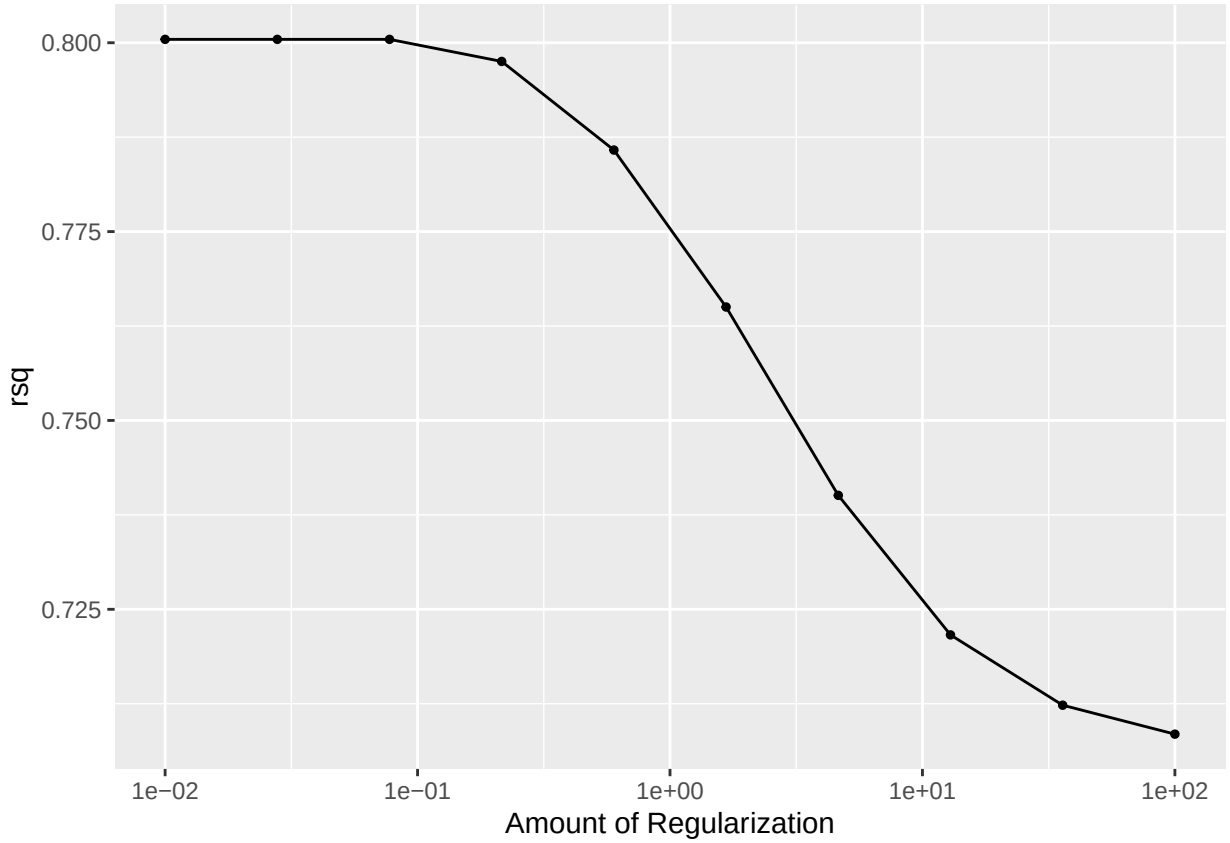
```
penalty_grid <- grid_regular(penalty(range = c(-2, 2)), levels = 10)
```

Then, I tuned the models across resamples. The results were saved to save computation time.

```
tune_res <- tune_grid(
  ridge_workflow,
  resamples = happiness_folds,
  grid = penalty_grid)

save(tune_res, file = "rr_tune.rda")
```

```
load("rr_tune.rda")
autoplot(tune_res, metric = "rsq")
```



Examining the results of the plot, we can see that the R Squared level remains rather constant, but drops significantly after a regularization level of 0.1.

Random Forest

For the sake of future models, I recreated the recipe without the transformations, polynomial features, and interaction terms. I felt this choice was wise because these models do not have the same assumptions of linearity and independence of the ridge regression model. Additionally, this will make interpretation of variables much simpler.

```
happiness_recipe2 <- recipe(happiness_score ~ elec_access+age_dep_ratio+co2_emissions+gdp+education_exp
#polynomial expansion on military spending because it has a u-shape when graphed against happiness scor
  step_normalize(all_predictors()) %>%
  step_scale(all_predictors())
```

Here, I tune min_n, mtry, and trees in a random forest function. The engine was set to 'ranger', and importance to 'impurity' for future use. Then, this and the recipe above were added to a workflow.

```
rf_spec <-
  rand_forest(
              min_n = tune(),
              mtry = tune(),
              trees = tune(),
              mode = "regression") %>%
  set_engine("ranger", importance = "impurity") #used for the vip() later in the project
```

```
rf_workflow <- workflow() %>%
  add_model(rf_spec) %>%
  add_recipe(happiness_recipe2)
```

The mtry was set from 1 to 11 because I wanted to test as many combinations of variables as possible. The trees were limited at 300 because, given the size of my dataset, I felt an increase in trees would not significantly improve performance of the model. Finally, mtry, the amount of observations needed to keep splitting nodes, was limited to 5. Because we can afford the computation time, levels were set to 8.

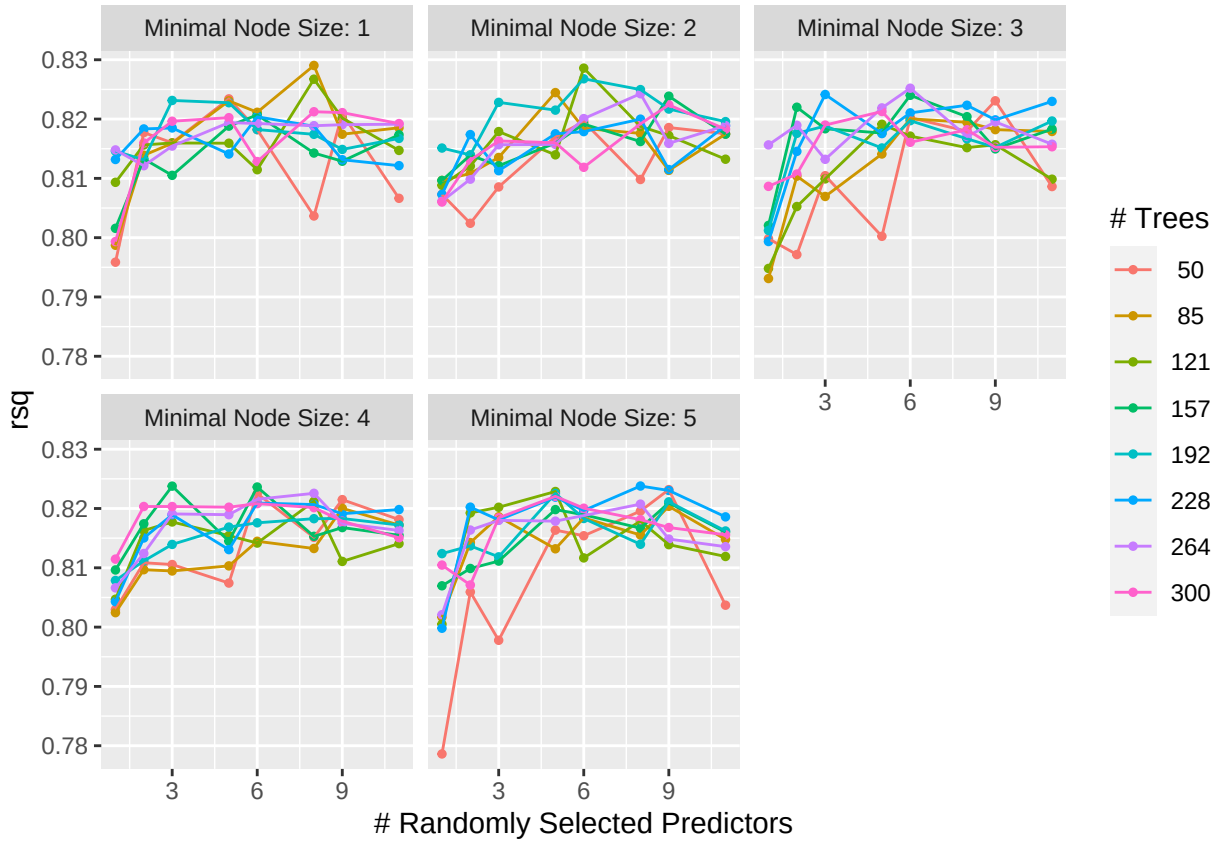
```
rf_grid <- grid_regular(
            mtry(range=c(1,11)), #11 is amount of variables, less than ridge since interaction terms no
            trees(range=c(50,300)),
            min_n(range=c(1,5)),
            levels = 8
            )
```

Once again, model is tuned across resamples and saved.

```
tune_res2 <- tune_grid(
  rf_workflow,
  resamples = happiness_folds,
  grid = rf_grid,
  metrics = metric_set(rsq)
)
save(tune_res2, file = "rf_tune.rda")
```

```
load("rf_tune.rda")

autoplot(tune_res2, metric = "rsq")
```



On average, it seems that lower minimal node sizes tend to peak higher but have more variance in performance. Similarly, a high number of trees seem to be more consistent but lower trees spike higher. Finally, the optimal amount of randomly selected predictors seems to be somewhere from 6 to 8.

Boosted Tree

The boosted tree model is similar to the random forest model. We will be tuning learn_rate and mtry, and setting the engine to xg_boost. Then, saving the bt_model and our second recipe to the workflow.

```
bt_model <- boost_tree(mode = "regression",
                       learn_rate = tune(),
                       mtry = tune()%>% #mtry was tuned because I felt, from our previous model, it had
                       set_engine("xgboost"))

bt_workflow <- workflow() %>%
  add_model(bt_model) %>%
  add_recipe(happiness_recipe2)
```

The learn rate was set to (-1,1) since I figured learning rate could make a monumental difference due to overfitting of models. The range for mtry was set from 1 to 11 for reasons stated above. I also created 8 levels, similar to the random forest model.

```
bt_grid <- grid_regular(
            learn_rate(range = c(-1,1)),
              mtry(range=c(1,11)),
```

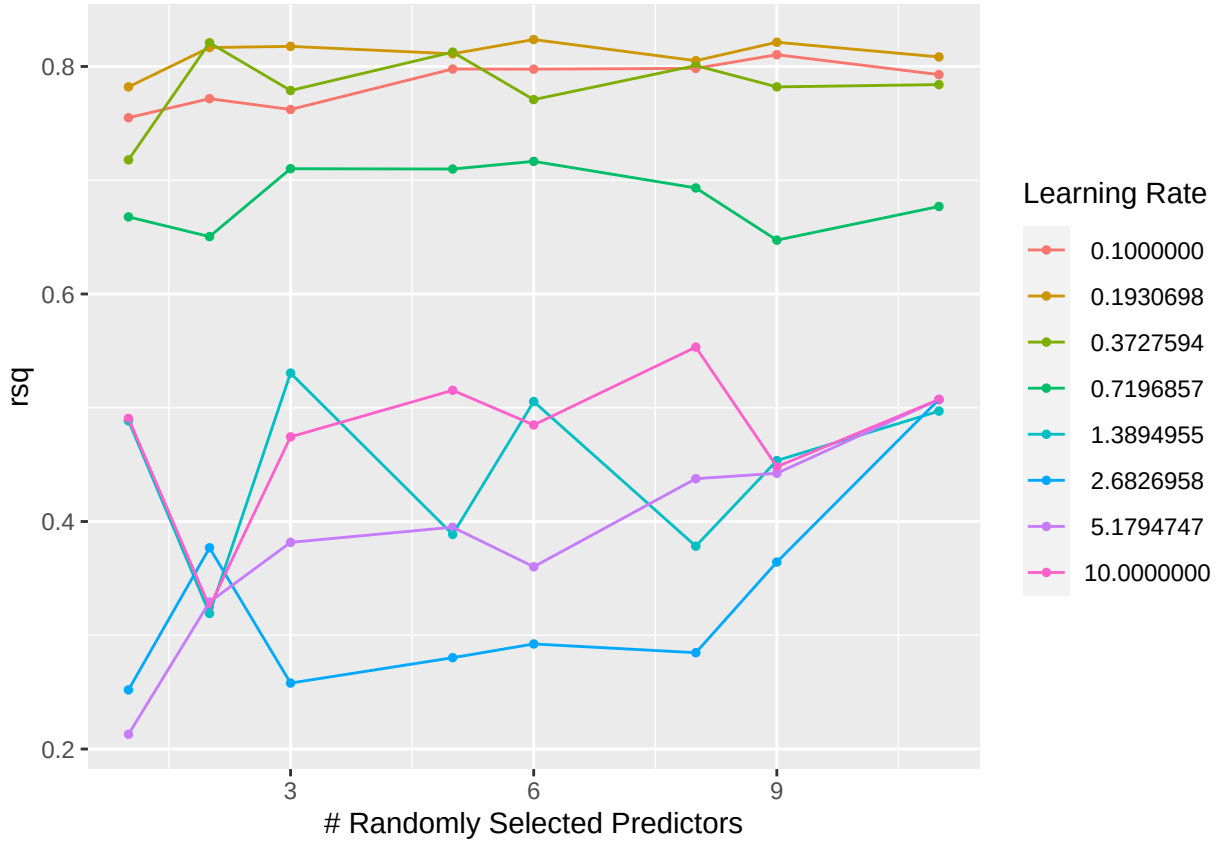
```
            levels = 8
            )
```

Model is tuned and saved.

```
tune_res3 <- tune_grid(
  bt_workflow,
  resamples = happiness_folds,
  grid = bt_grid,
  metrics = metric_set(rsq))
save(tune_res3, file = "bt_tune.rda")
```

```
load("bt_tune.rda")

autoplot(tune_res3, metric = "rsq")
```



Looking at the graph, it seems like the number of randomly selected predictors did not actually make a significant difference; however, the R- Squared varies greatly between different learning rates. Because of the size of our dataset, I think it is likely that a high learning rate leads to overfitting in almost every case.

K-Nearest Neighbors

For the final model, I tuned the neighbors parameter in a KNN function, and then saved it into a workflow along with the second recipe.

```
knn_model <-
  nearest_neighbor(
    neighbors = tune(),
    mode = "regression") %>%
  set_engine("kknn")

knn_workflow <- workflow() %>%
  add_model(knn_model) %>%
  add_recipe(happiness_recipe2)
```

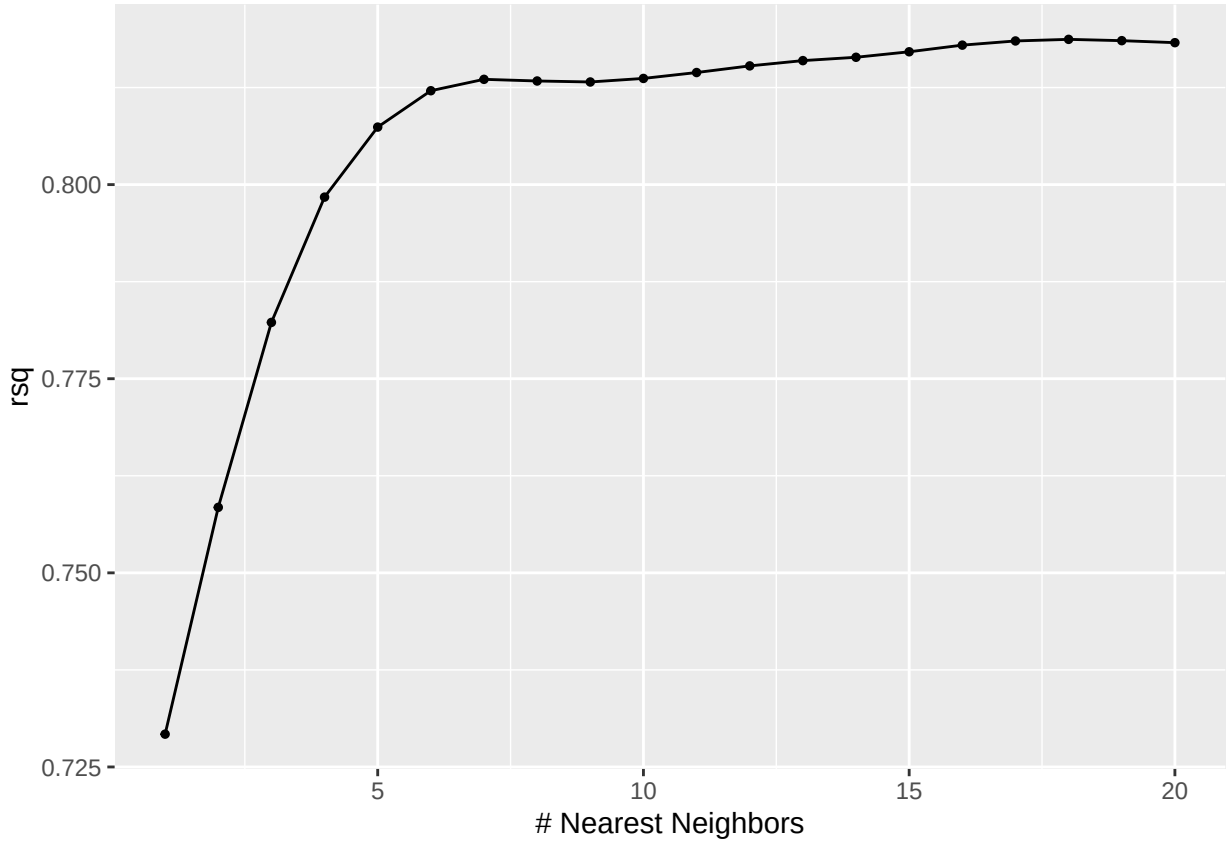
The range of neighbors was set from 1 to 20 because I felt the optimal range to prevent under/overfitting the model would be inbetween these values. I also set the levels to 20 so that every # of neighbors from 1 to 20 can be accounted for in the model.

```
knn_grid <- grid_regular(neighbors(range=c(1,20)), levels = 20)
```

Model is tuned and saved.

```
tune_res4 <- knn_workflow %>%
    tune_grid(
    resamples = happiness_folds,
    grid = knn_grid)
save(tune_res4, file = "knnt_tune.rda")
```

```
load("knnt_tune.rda")
autoplot(tune_res4, metric = "rsq")
```



It seems that our model makes a significant increase in R Squared from 1 to 7 nearest neighbors. However, after this, the number seems to stagnate significantly.

Final Model and Analysis

It's time to compile the results of all of my models and see which one performed the best! Here, I collect the metrics for all my models, take the highest R-Squared value for each model, then I compile all the values into one table.

```
#process of collecting metrics for each model, then compiling it into one chart.
a<-collect_metrics(tune_res)[collect_metrics(tune_res)$'.metric' == 'rsq',]%>%
  arrange(-mean)%>%
  filter(row_number()==1)
a['type']<-'rr' #Ridge Regression
b<-collect_metrics(tune_res2)[collect_metrics(tune_res2)$'.metric' == 'rsq',]%>%
  arrange(-mean)%>%
  filter(row_number()==1)
b['type']<-'rf' #Random Forest
c<-collect_metrics(tune_res3)[collect_metrics(tune_res3)$'.metric' == 'rsq',]%>%
  arrange(-mean)%>%
  filter(row_number()==1)
c['type']<-'bt' #Boosted Tree
d<-collect_metrics(tune_res4)[collect_metrics(tune_res4)$'.metric' == 'rsq',]%>%
  arrange(-mean)%>%
  filter(row_number()==1)
d['type']<-'knn' # K nearest Neighbors

inner<-rbind(a[2:8],b[4:10])%>%
        rbind(c[3:9])%>%
        rbind(d[2:8])%>%
         arrange(-mean)
inner
```

```
## # A tibble: 4 x 7
##   .metric .estimator  mean     n std_err .config                type
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>                  <chr>
## 1 rsq     standard   0.829    10  0.0227 Preprocessor1_Model014 rf
## 2 rsq     standard   0.824    10  0.0216 Preprocessor1_Model34  bt
## 3 rsq     standard   0.819    10  0.0321 Preprocessor1_Model18  knn
## 4 rsq     standard   0.800    10  0.0387 Preprocessor1_Model01  rr
```

Interestingly, all of our models performed rather similar to each other with the Random Forest model marginally performing the best with an R-Squared value of .829. This value represents the proportion of variance in the outcome variable, happiness score, that is explained by the predictor variables. Now, we will be creating a final workflow with our best performing random forest model. Then, we fit it with the training set.

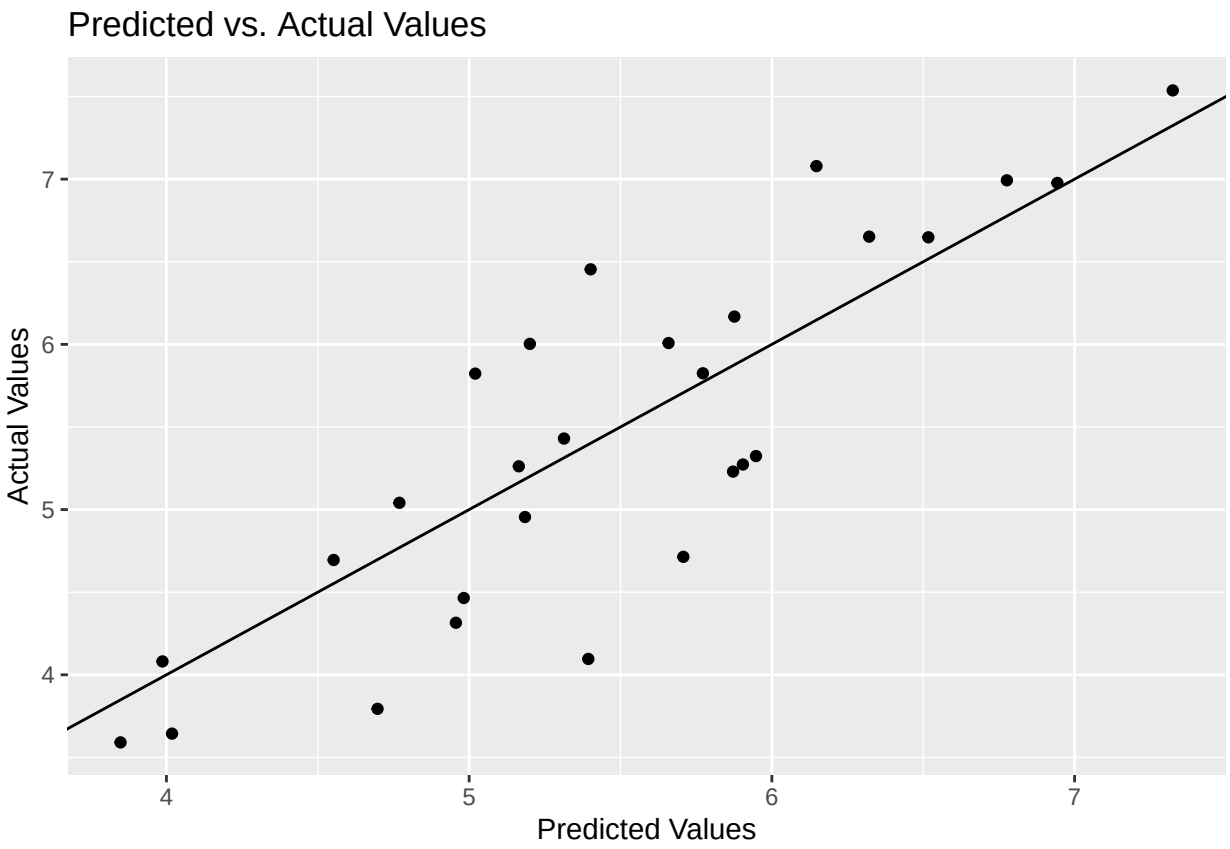
```
best_complexity <- select_best(tune_res2)
rf_final <- finalize_workflow(rf_workflow, best_complexity)
rf_final_fit <- fit(rf_final, data = happiness_train)
```

Now, we can use our final model and use it to predict values in the testing set. The expected values are put into a table and then graphed below.

```
happiness_metric <- metric_set(rsq)

model_test_predictions <- predict(rf_final_fit, new_data = happiness_test) %>%
  bind_cols(happiness_test %>% select(happiness_score))

ggplot(model_test_predictions, aes(x=.pred, y= happiness_score)) +
  geom_point() +
  geom_abline(intercept=0, slope=1) +
  labs(x='Predicted Values', y='Actual Values', title='Predicted vs. Actual Values')
```



Our final predicted vs actual values plot shows that our values seem to follow the y = x line with relative accuracy. Ideally, we want these points to go along this line because that means we correctly guessed the happiness score value. Interestingly, it seems that high predicted values on our plot are consistently close to their actual values. Therefore, given what we can see from the testing data, our model predicts high happiness scores with good accuracy.

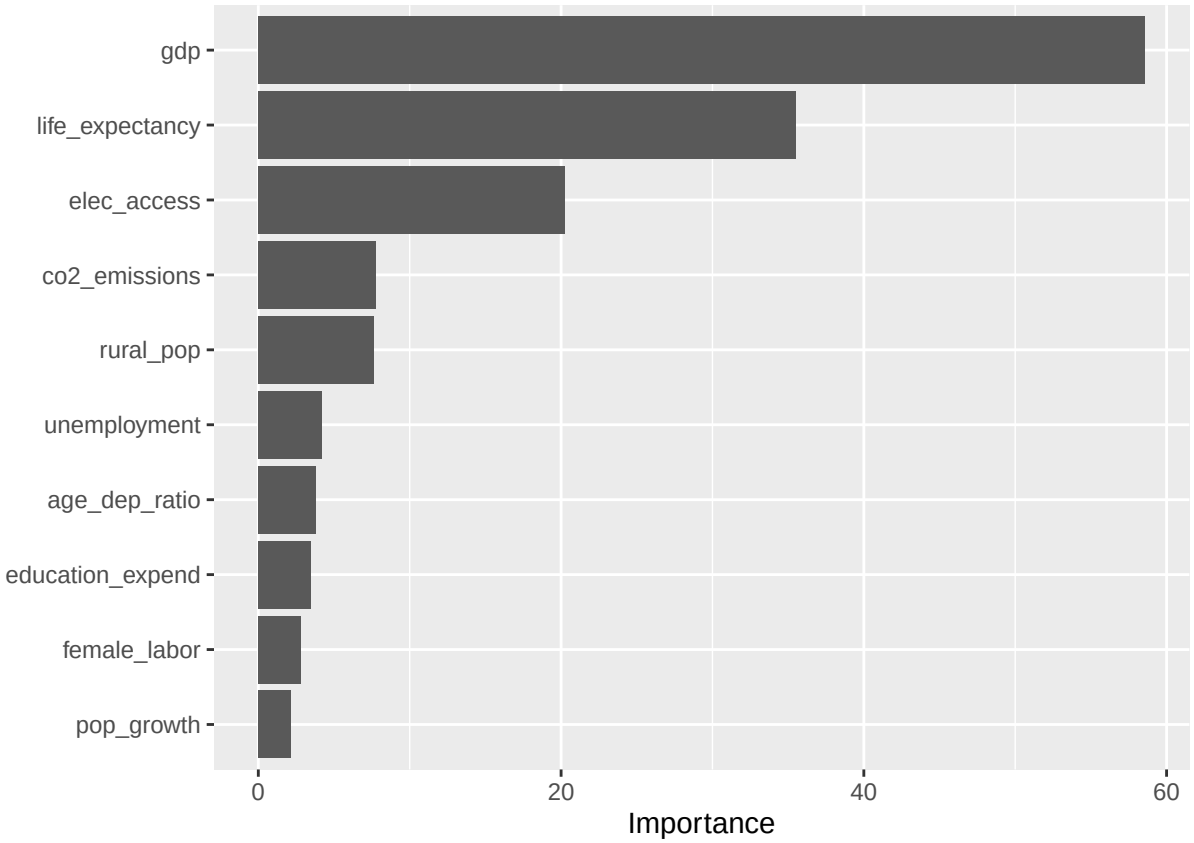
```
model_test_predictions %>%
  happiness_metric(truth = happiness_score, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.731
```

Further analyzing our data, we can see that our R-Squared value is ~73.1, around 10% less than the value on the training set. This would indicate that we did overfit our model.

Finally, it would be useful to identify which explanatory variables are influential in determining happiness score. The vip, variable importance plot, function allows us to visualize this.

```
rf_final_fit %>%
  extract_fit_engine() %>%
  vip()
```



GDP per capita seems to be the most important independent variable in our model. This is followed by average life expectancy and electricity access(% of population); given the correlation we found in our EDA, it would makes sense that both of these variables are influential. However, unexpectedly, the age dependency ratio did not seem to have a large effect despite its correlation with the other two variables. From a more intuitive approach, GDP and electricity access are things that can increase many qualities of life (health, entertainment, etc), thereby increasing happiness score as well.

Conclusion

One of my main concerns when starting this project involved my choice in explanatory variables since limiting the determinants of happiness to a few variables is difficult. Ultimately, I chose 11 socioeconomic factors that I believed to encompass a wide range of facets of life. I also had to ensure that my choice of variables had an appropriate amount of data for all the countries in the dataset since missing values could heavily skew results.

Initially, I was rather skeptical if GDP was an appropriate indicator of wealth since it does not account for wealth disparity or cost of goods in a nation. However, it does seem that, regardless of that fact, GDP was

a crucial indicator of happiness score. Furthermore, it seems that any factors associated with wealth seemed to be important in our model.

Ultimately, all of our models ended up performing exceptionally well. Ridge regression likely performed the worst due to the parametric nature of the model. Although transformations and other methods were utilized to follow assumptions, some variables still had rather large correlations with eachother. Additionally, the relationship between happiness score and some explanatory variables were only moderately linear.

Ideally, for future research, there should be more observations to conduct analysis on. Unfortunately, there is limitations on the amount of countries that exist. Potentially, you can take data from the same countries in different years, but it is likely that the data for countries does not change significantly from year to year. Alternatively, with more resources, countries can be further divided into states/provinces/cities to increase the number of observations.

Our model using the World Happiness Report and World Bank data is an interesting tool in visualizing what variables contribute to a country's happiness. Our model was largely successful in predicting the happiness levels of countries and was hopefully insightful into how you view happiness yourself.