OM 420 FINAL PROJECT

Ananya, Lane, Yazan

Introduction

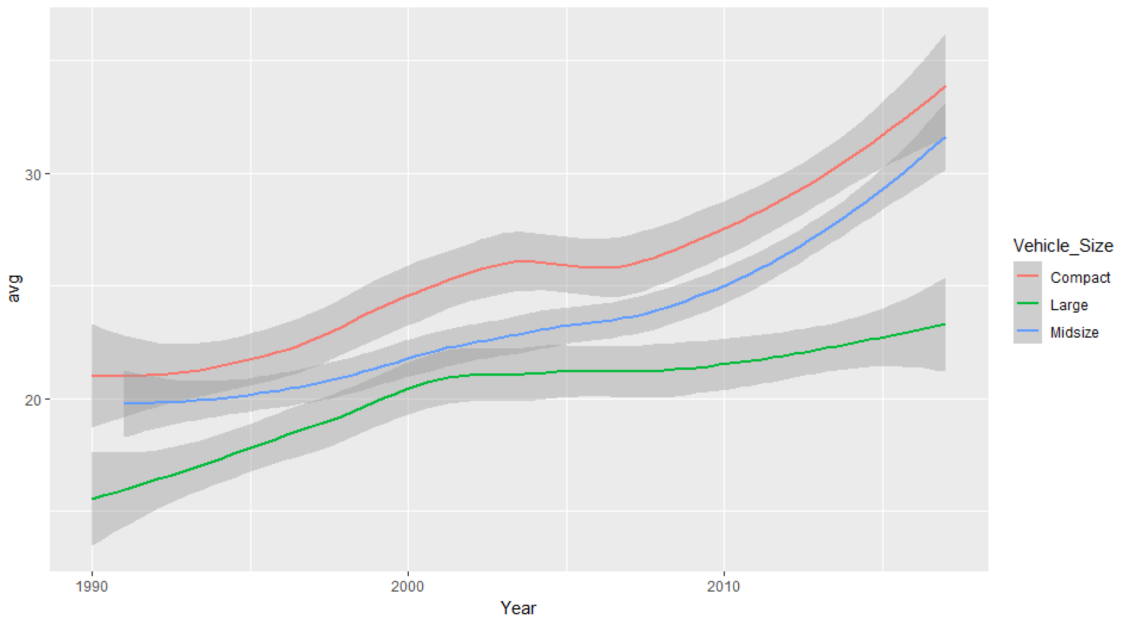
The consumer light vehicle market has shown steady growth since 2004, averaging about 2.8% growth in sales per year over that time. With this in mind, we searched for a dataset that had potential to provide us with insight that could lead to informed business decisions. After finding our dataset, we developed our business objective of *predicting MSRP to assist in making informed manufacturing decisions*. Many interesting findings were discovered while exploring the data, which we will cover before examining which of the four models we developed was the best at predicting MSRP.

Dataset

The dataset used was found at kaggle.com and includes 11914 observations of 16 variables (See Appendix A for detailed column descriptions). Some data manipulation was required to prepare the data frame before the models could be created. First, it was evident that the default value for an unknown MSRP was $2000. Therefore, there were a large number of vehicles that had this price. This had the potential to skew the results and therefore these observations were removed. The column *Market Category* was in a comma separated format, which allowed an observation to have multiple market categories. To simplify the analysis, separate logical columns were created for each possible category. After these manipulations the dataset was ready for the exploratory data analysis. Before the models were created, the dataset was filtered further. Restringing observations to only those with the 'Regular Unleaded' fuel type removed vehicles that were outside the scope of our business objective, such as exotic and high performance vehicles. This also left us with a more consistent dataset as few vehicles before 2015 used an alternate fuel type. (See Appendix B for DataFrame Manipulations)

Observation 1

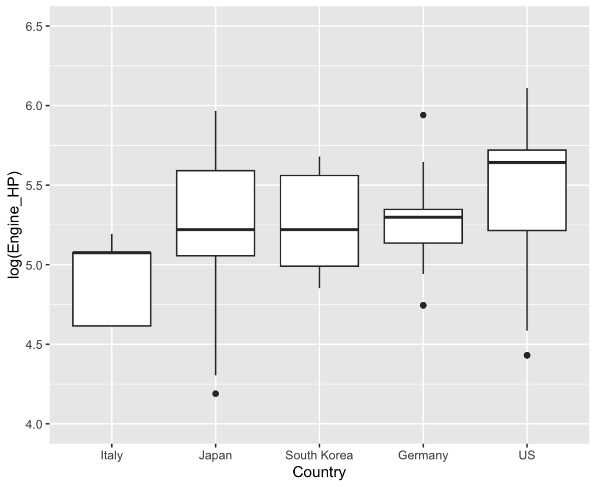
Large vehicles displayed smaller proportional improvement in fuel economy when compared to other vehicle sizes (Midsize, Compact). This was evident while investigating what attributes change over time. Fuel Economy showed consistent improvements over time, which led to an investigation into what may cause this. During this investigation, the fuel economies were compared for each unique vehicle size, which led to the following plot and this interesting finding (See Appendix C for R code):



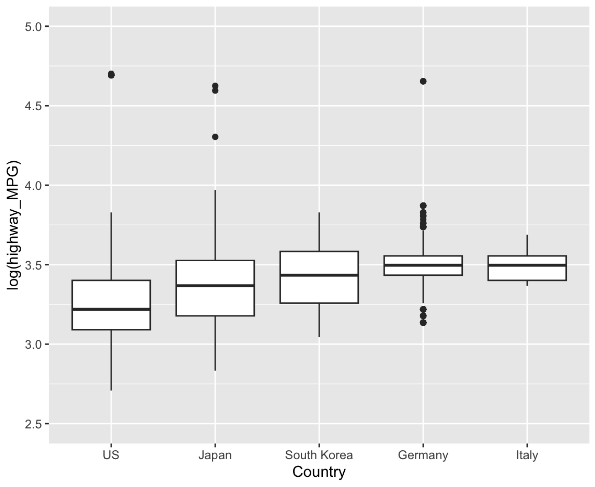
Observation 2

The second interesting observation was based on comparing leading car manufacturing countries' performances. We found that in the last decade, for "mid range" cars, certain leading car manufacturing countries compared significantly better based on their horsepower / miles per gallon values. Boxplots were used for this comparison and Horsepower (HP) and Miles per gallon (mpg) were used as estimates for determining the efficiency (performance) of a car. A higher value for both is ideal. Before plotting, the data values were log transformed to reduce outliers and fit the plot better. (See Appendix D for Observation 2 code).

Boxplot: Engine HP grouped by Country



Boxplot: MPG grouped by Country

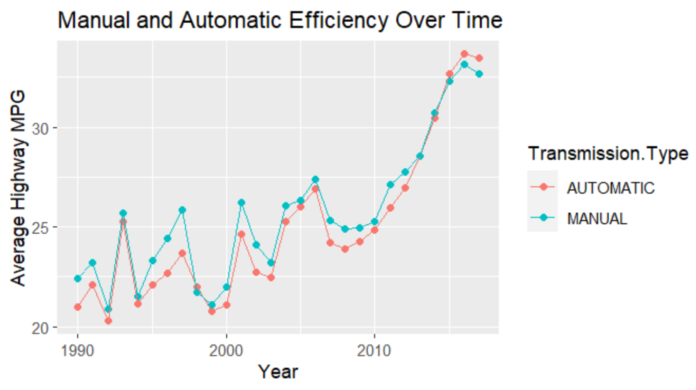


We can observe the median values and see how the countries stack up against each other. Let us look at an example. In the HP plot, we see that the US had the highest median horsepower. However, in the mpg plot, it had the lowest median mpg value. This indicates an inverse relationship between HP and mpg. This could be explained by manufacturing decisions that tend to compromise on fuel economy to increase their engine efficiency. German cars seem to have high median values for both HP and mpg, making them the best performing manufacturing country.

Observation 3

An interesting observation can be made if one were to take cars that offered manual and automatic transmissions for a given model, and compare highMPG to see which is the more efficient variant.

Our results were that up until the very latest years present in the data set, manual transmissions were largely the most efficient option, while both transmissions became more efficient overtime, automatic transmissions overtook manuals for model year 2015. During the analysis, 3 different time periods were taken to see a pattern, the oldest cars present within the database (1990-94), newer cars (2004-05) and the cars from the latest year (2017).



For the 1990-94 period, we can clearly see that manuals far outperform automatic transmissions, While we see a slight shift within the 2004-05 category, the real change is present within the cars from 2017, with automatic variants being more fuel efficient than manuals.

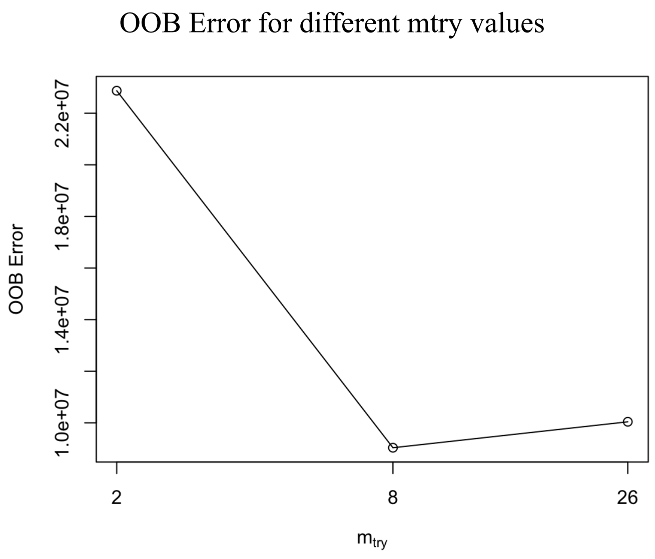
Overall, this plot shows that while manual transmissions have a clear advantage over automatics, over time we see that while both certainly improve in fuel efficiency over time, automatic transmissions eventually overtake manuals at around 2014, and widens the gap even further in 2016 and 2017. Our conclusion for this observation is that given a make and model, manual transmissions prior to the mid 2010s tend to be more fuel efficient, but this will unlikely be the case as newer and newer cars are released, as automatic transmissions now are more fuel efficient than ever. This may prove significant for buyers looking to optimize for fuel efficiency. (See Appendix E for code and plots for this observation).

Model Building

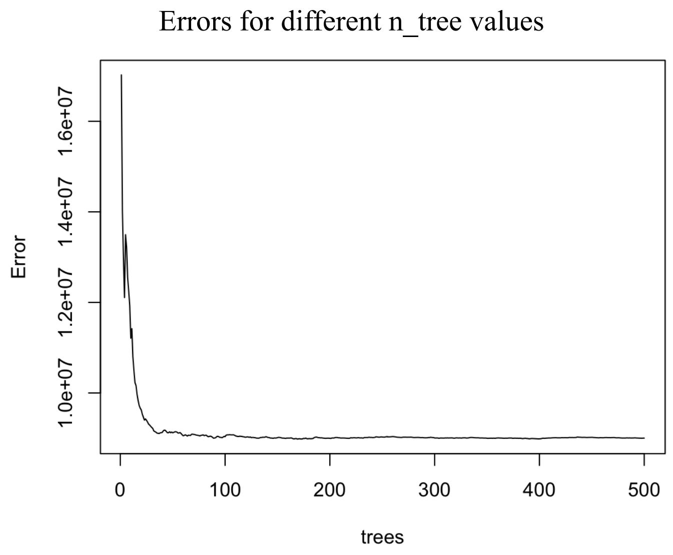
To ensure a fair comparison between our models, 25% of the data was used as holdout data. This means that it wasn't used for training models and was only used in testing when each of our 4 different models were finalized. This left 75% of the data to be used training and cross-validating different approaches to the same type of model.

Model 1: Random Forest

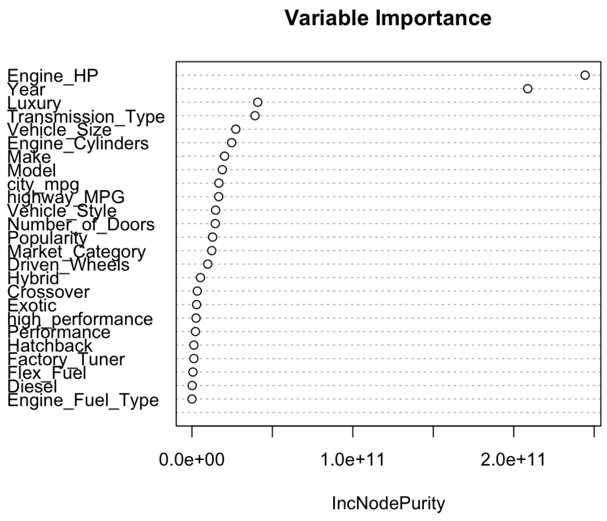
Initial analysis of our dataset suggests that compared to categorical models, regression models might be more suitable for modeling our business objective. Thus, we decided to use the random forest model.
We begin by training our model. The first step is determining optimal values for the model parameters mtry and ntree. Mtry is the number of variables that the model will randomly sample as candidates at each split in the tree. Ntree is the number of trees in the random forest. The Random Forest package in R has some features that will help us find these values.



For determining mtry, we use the TuneRF function with the 'plots' parameter as True. This gives us a plot showcasing how the OOB "Out of Bag" error, which is a method of measuring the prediction error of random forests, changes with different values of mtry. As observed, mtry = 8 is the optimal value. Notice that this is the value p/3 where p is the total number of predictors (26).



For determining ntree, we first create a naive random forest object without specifying any parameters. On plotting this object we get a plot that shows how the error rate changes with the number of trees . We observe that after n_tree = 200 the error stabilizes, and this is our optimal value.

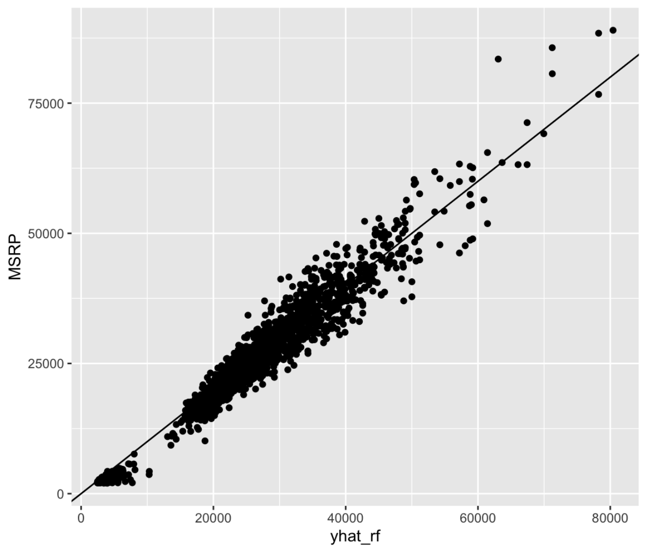


With our naive random forest object, we can also generate a variable importance plot. We notice that quite a few variables are not important to our analysis. For the next run of the model, we do not include the variables *Hatchback Factory Tuner, Flex Fuel, Diesel,* and *Engine_Fuel_Type* in our model.

Let us compare the original random forest model with the new, improved model by taking a look at the Mean Squared Error values of the training and test sets.

Model	Original Model (With default parameters and all predictors)	Final Model (With optimal parameters and important predictors)
Train_MSE	10,401,634	9,540,823
Test_MSE	11,105,798	10,293,219

The final model has a significantly better Train and Test MSE value than the original model. Finally, let us visually observe how close the model's predictions were to the actual MSRP values.



Random Forest Performance

We can see that for the most part, the model does a good job of predicting MSRP values. Most points lie on or close to the line of best fit.

For higher MSRP values, the predictions were not as accurate : Points are away from the line of best fit.
Overall, the random forest model is definitely promising when it comes to modeling our business objective.
(See Appendix F for Model Code).

Model 2: Linear Model

Given the desirable simplicity of a linear model, and the various continuous variables that could work well with a linear format, we investigated the possibility of mapping MSRP as the responding variable, and created multiple linear models with different independent variables out of the following models:

1. A model that takes all continuous variables into account
2. A model that includes Market Categories as categorical variables, and a few continuous variables
3. A model that includes Make, Transmission Type, Driven Wheels, Vehicle Size, and some continuous variables.

Of all of the models tested, model 2 produced the best MSE for the train data, as seen below:

Model	Mod1	Mod2	Mod3
Train MSE (Test Data)	57,382,690	42,519,361	42,537,649
Test MSE (Test Data)	55,698,567	43,667,638	42,184,578

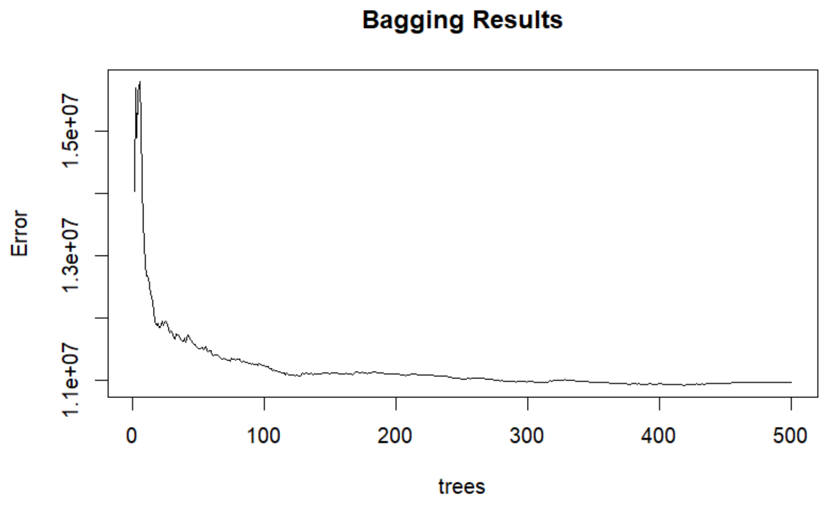
Looking at the coefficients of the second model, continuous variables, such as horsepower and year positively affect MSRP, which is logical given that newer cars tend to be more valuable than older cars, and higher engine power output is typically found with more expensive cars. HighwayMPG was reported to decrease MSRP, which also checks out as most fuel efficient vehicles tend to be subcompact vehicles that usually sell for not much money. What was surprising was that Engine cylinders played a negative role in MSRP, an explanation comes from the fact that the only high cylinder vehicles that accept regular unleaded fuel are generally an outlier, and the ones that do have over 10 cylinders that were in the database were nearly 20 year old cars that suffered extremely poorly from used car market pricing as a result. Market categories played a largely predictable role in MSRP, Automatics are usually more expensive with manuals, and this was reflected in the model. Exotic, luxury, high-performance categories increased MSRP, while hatchbacks, crossovers, and diesel cars drove down MSRP. Oddly enough, the "performance" category decreases MSRP, this is possibly due to performance and high performance having much different standards of measurement (See Appendix G for linear model code).

Model 3: Bagging

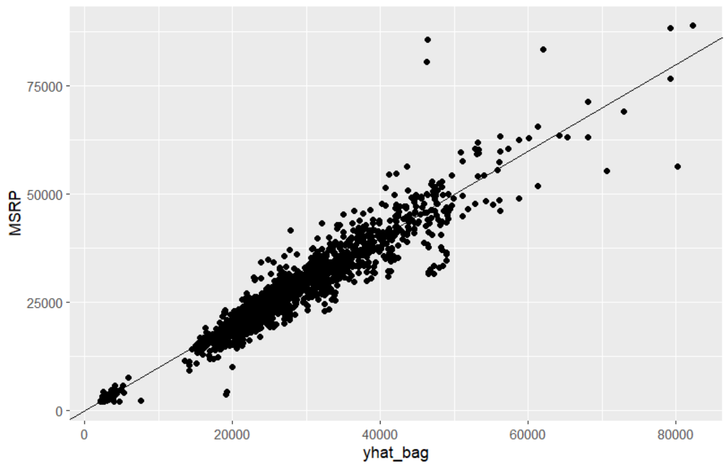
With the usage of random forests, we wanted to try experimenting with a variation of random forest. By making mtry equal to all columns present apart from MSRP, we tested the bagging approach with the training data numerous times. Our results were MSEs that fluctuated from 12,500,000 and 13,400,000 (with 12,573,932 being to lowest recorded), but setting the seed to 1 gives us a consistent result of 13,365,173.

	Attempt 1	Attempt 2	Attempt 3
MSE_bag	13,365,173	12,573,932	13,333,687

With the holdout data, the MSE was 13,596,911. Shown below, the graph reveals that the MSE of our predictions is minimized at 419 trees:



Using ŷ, we plotted it against the MSRP. Visually speaking, the predictions do largely match the line of best fit, although it does lose consistency as the plot goes on. (See Appendix H for R code)

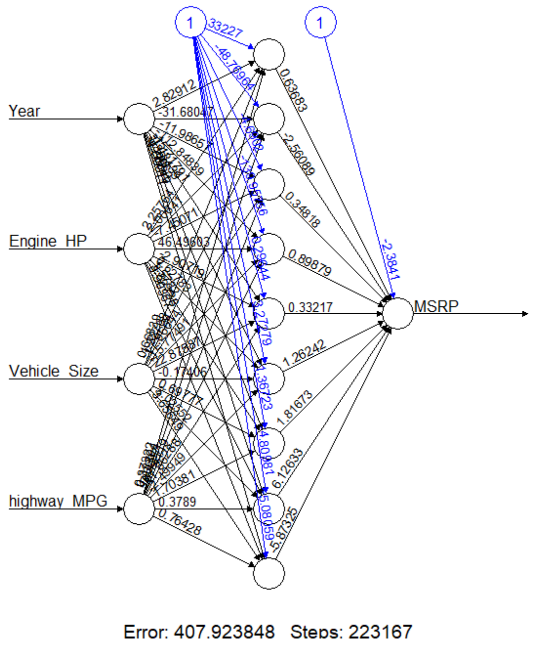


Model 4: Neural Network

The last model to be fitted to the data was a neural network. Since there is a high computational cost to training a neural network, the input neurons had to be limited to 4 before using cross-validation to select the optimal amount of hidden neurons. Year, Engine_HP, Vehicle_Style, and Highway_MPG were chosen as input parameters, based loosely on findings during the exploratory data analysis and some trial and error. 10-fold cross validation was then used to find the optimal amount of hidden neurons. All positive single digit integers were tested during the process, with the best number (h) of hidden neurons selected as the base model using MSE of the predicted MSRP vs. the actual MSRP. Below is the MSE of each model tested:

h	1	2	3	4	5	6	7	8	9
Train MSE	59555919	43427187	37349118	36498469	35659051	34262188	33768676	32841662	32058156
Test MSE	59808438	43401082	37776229	37468664	36929844	35238646	35473648	34155438	34183835

As expected, as more neurons are added the better the model performs. The model with 1 and 2 hidden neurons show significantly worse performance than the other models, which get marginally better as the amount of hidden neurons is increased. With h > 6, the model starts to show worse performance on the test set vs. the training set, but the difference is not significant enough to warn of any overfitting that might be happening. The neural network with 9 hidden neurons is the best single layer neural network. To create the final model to be used in comparison with the other models, 70% of the data points were (randomly) selected as training data, and the threshold was decreased from 0.05 to 0.01 to increase accuracy. The performance increased with the lower threshold, with a MSE of 29,577,544 on the holdout data. The model is shown below (See Appendix I for R code):



Conclusion

To compare the four models, their performance was measured by the MSE of their predictions on the holdout data set. The results were as follows:

Model	Random Forest	Linear	Bagging	Neural Network
Holdout MSE	9,760,966	42,737,938	13,596,911	29,577,544

The Linear model showed the worst performance, suggesting that our dataset might be non-linear in nature. Furthermore, there might be collinearity between variables (e.g Horsepower and miles per gallon) and interaction effects were not considered in the simple linear model. The Neural Network model had a lot of computational overhead and only managed to increase performance slightly. Methods of Bagging and Random Forest showed significant improvements. Between the two, random forest models had the lowest MSE value, and subsequently the better performance. In general, the random forest model tends to perform better than bagging due to random feature selection, which makes its trees more independent with better bias-variance trade offs. This was reflective in our model analysis.

Overall, the random forest model is the best suited model for our business objective of predicting MSRP to assist in making informed manufacturing decisions.

Appendix A - Column Descriptions

Column #	Column Name	Type	Extended Description
1	Make	Character	
2	Model	Character	
3	Year	Number	Model year of the vehicle
4	Engine Fuel Type	Character	Required fuel type for engine
5	Engine HP	Number	Engine power in Horsepower
6	Engine Cylinders	Number	Number of cylinders in engine
7	Transmission Type	Character	
8	Driven Wheels	Number	
9	Number of Doors	Number	
10	Market Category	Character (csv)	Performance, Exotic, etc.
11	Vehicle Size	String	Compact, Midsize or Large
12	Vehicle Style	String	Coupe, Convertible, etc.
13	Highway MPG	Number	Highway fuel efficiency in MPG
14	City MPG	Number	City fuel efficiency in MPG
15	Popularity	Number	Value based off of Twitter statistics
16	MSRP	Number	Manufacturer Suggested Retail Price (cost)

Available for download at: https://www.kaggle.com/datasets/CooperUnion/cardataset?resource=download

Appendix B - Dataframe Manipulations

```
# Load raw data from csv file
raw_data <- read_csv("data.csv")

# Rename columns with spaces
names(raw_data) <- str_replace_all(string=names(raw_data), pattern=" ",
replacement="_")

# Separate multiple market Categories
separated <- raw_data
grouped <- raw_data %>% group_by(Market_Category) %>% summarise()
while (dim(grouped)[1] != 0) {
  category <- str_split(string=grouped$Market_Category[1],pattern=",",
simplify=T)[1,1]
  separated <- separated %>% mutate("{category}" :=
str_detect(separated$Market_Category, category))
  grouped <- grouped %>% transmute(Market_Category = str_remove(Market_Category,
paste(category, ',',sep="")))
  grouped <- grouped %>% transmute(Market_Category = str_remove(Market_Category,
category))
  grouped <- grouped[grouped$Market_Category != "",]
  names(separated)
}

# cars over $2000
cars_df <- separated[separated$MSRP > 2000,]

# factors for graphing
cars_factored_df <- cars_df %>%
  mutate(Engine_Cylinders = as.factor(Engine_Cylinders), Number_of_Doors =
as.factor(Number_of_Doors))

# dataframe for model training and testing
model_df <- cars_df[complete.cases(cars_df),] %>% filter(Engine_Fuel_Type == "regular
unleaded")

# get indexes to dataframe rows
indexes = 1:nrow(model_df)
size = length(indexes)

# holdout 25% for comparison of models
set.seed(1)
holdout_index <- sample(size, floor(size * 0.25))
holdout_df <- model_df[holdout_index,]

# 75% available for training and comparing similar models
remaining_df <- model_df[-holdout_index,]
```

Appendix C - Code for Observation 1

```
library(dplyr)
data <- read.csv("data.csv")

# HOW DOES MSRP raise over many years?
data %>% ggplot(data = .) + geom_point(mapping = aes(x=Year, y=MSRP,
color=Vehicle.Style))
data %>% group_by(Year) %>% summarise(avg_msrp = mean(MSRP)) %>%
  ggplot(data = .) + geom_line(mapping = aes(x=Year, y=avg_msrp))

# Slight rise over time but not very definitive and very sporadic

# relationship between HP and fuel economy
ggplot(data = data) + geom_point(mapping = aes(x=Engine.HP, y=highway.MPG)) +
  geom_smooth(mapping = aes(x=Engine.HP, y=highway.MPG))
# At a very basic level fuel economy decreases with more HP (this makes alot of sense)

# how cylinders effect fuel economy
data %>% ggplot() + geom_point(mapping=aes(x=Engine.Cylinders, y=highway.MPG))
data %>% ggplot() +
  geom_boxplot(mapping=aes(x = Engine.Cylinders, y=highway.MPG,
group=Engine.Cylinders)) +
  ylim(0,150)
# this is a really interesting plot

# This looks vaguely like an inverse function

# How about separated by class?
style_grouped <- data %>% group_by(Vehicle.Style) %>%
  summarise(mpg = mean(highway.MPG, na.rm = T), hp = mean(Engine.HP, na.rm = T))
style_grouped %>% ggplot(data = .) + geom_line(mapping=aes(x=mpg, y=hp))
```

Appendix D - Code for Observation 2

```
# Load raw data from dataset csv file
raw_data <- read_csv("data.csv")

# some data manipulation
# Rename columns with spaces
names(raw_data) <- str_replace_all(string=names(raw_data), pattern=" ",
replacement="_")

# Separate multiple market Categories
separated <- raw_data
grouped <- raw_data %>% group_by(Market_Category) %>% summarise()
while (dim(grouped)[1] != 0) {
  category <- str_split(string=grouped$Market_Category[1],pattern=",",
simplify=T)[1,1]
  separated <- separated %>% mutate("{category}" :=
str_detect(separated$Market_Category, category))
  grouped <- grouped %>% transmute(Market_Category = str_remove(Market_Category,
paste(category, ',',sep="")))
  grouped <- grouped %>% transmute(Market_Category = str_remove(Market_Category,
category))
  grouped <- grouped[grouped$Market_Category != "",]
  names(separated)
}
names(separated) <- str_replace_all(string=names(separated), pattern=" ",
replacement="_")
test <- separated
# Changing spaces to underscores in the Make column. Without this, an error is
produced
#during left join in finding 2.

test$Make <- sub(" ", "_", test$Make)

# original dataset did not have information about country, thus manually created
another dataset with 38 makes and their origin countries.
# this dataset can be found at
#https://drive.google.com/file/d/1xsO-jpeSZqXSxmJAEzvN1b5NImHdm3wU/view?usp=sharing
make_country <- read.csv("make_origin_country.csv")

# working on a copy of raw_data to avoid making any changes to it
test2 <- raw_data

# left outer join country information to the original dataframe
test2 <- test2 %>% left_join(make_country, by = "Make")

# filter out the luxury,high performace, Exotic and Factory tuned cars to get a
# better estimate for mid-range cars. Apply other filters
test2 <- test2 %>%
  filter(Country %in% c("England","Germany","Italy","Japan","US","South Korea") &
           Exotic == FALSE & Luxury == FALSE & `High-Performance` == FALSE &
           `Factory_Tuner` == FALSE & Year >= 2010 &
           !is.na(highway_MPG) & !is.na(Engine_HP) & !is.na(MSRP))

# MPG and HP show an inverse relationship
ggplot(data = test2) + geom_point(mapping = aes(x = Engine_HP, y = highway_MPG))

# ggplot for engine HP grouped by country. Arranged in order of median.
ggplot(data = test2) +
  geom_boxplot(mapping = aes(x = reorder(Country,log(Engine_HP),median), y =
log(Engine_HP))) +
  coord_cartesian(ylim = c(4, 6.5)) + scale_x_discrete(name="Country")
```

```
# ggplot for mpg grouped by country. Arranged in order of median.
ggplot(data = test2) +
  geom_boxplot(mapping = aes(reorder(Country,log(highway_MPG),median), y =
log(highway_MPG)))+
  coord_cartesian(ylim = c(2.5, 5)) +  scale_x_discrete(name="Country")
```

Appendix E - Code for Observation 3

```
library(tidyverse)
library(ggplot2)
observation3Data <- read.csv("data.csv")

#Given the exact same make and model, is the automatic,manual, automated_manual type
more efficient in terms of fuel economy
autoManPair <- observation3Data %>% group_by(Make,Model,Year,Transmission.Type)%>%
filter( Transmission.Type %in% c("AUTOMATIC","MANUAL"))%>%
summarise(highway.MPG,city.mpg,count = n()) %>% unique()
asd <- observation3Data %>% group_by(Make,Model,Year) %>% filter( Transmission.Type
%in% c("AUTOMATIC","MANUAL"))%>% summarise(count =  n_distinct(Transmission.Type)) %>%
filter(count > 1)
autoManPair <- inner_join(asd,autoManPair, c("Make","Model","Year"))
autoManPair <- autoManPair %>% group_by(Make,Model,Year,Transmission.Type) %>%
summarise(avgHMPG = mean(highway.MPG),avgCMPG = mean(city.mpg))
#Compare older cars here: 1990-94
old_cars <- autoManPair %>% filter(Year <1994, Transmission.Type %in%
c("AUTOMATIC","MANUAL","AUTOMATED_MANUAL")) %>% arrange(Year) %>% mutate(Make_Model =
str_c(Make," ",Model))
old_cars %>% ggplot(data = .) + geom_bar(aes(x = Make_Model,y = avgHMPG,
fill=Transmission.Type), stat = "identity",position = "dodge",width = 0.4)+ labs(y=
"Average Highway MPG", x = "Make and Model") + scale_x_discrete(guide =
guide_axis(n.dodge=2))
#Compare millennium cars here: 2000-2005
millenium_cars <- autoManPair %>% filter(Year <=2005, Year >=2004,Transmission.Type
%in% c("AUTOMATIC","MANUAL")) %>% arrange(Year) %>% mutate(Make_Model = str_c(Make,"
",Model))
millenium_cars %>% ggplot(data = .) + geom_bar(aes(x = Make_Model,y = avgHMPG,
fill=Transmission.Type), stat = "identity",position = "dodge",width = 0.4) + labs(y=
"Average Highway MPG", x = "Make and Model") + scale_x_discrete(guide =
guide_axis(n.dodge=2))
#Compare latest cars here: 2016-2017
new_cars <- autoManPair %>% filter(Year ==2017,Transmission.Type %in%
c("AUTOMATIC","MANUAL")) %>% arrange(Year) %>% mutate(Make_Model = str_c(Make,"
",Model))
new_cars_sample <- new_cars[1:32,]
new_cars_sample %>% ggplot(data = .) + geom_bar(aes(x = Make_Model,y = avgHMPG,
fill=Transmission.Type), stat = "identity",position = "dodge",width = 0.4) + labs(y=
"Average Highway MPG", x = "Make and Model") + scale_x_discrete(guide =
guide_axis(n.dodge=2))

proportion <- old_cars %>% group_by(Make,Model,Year) %>% summarize(highwayDiff =
diff(avgHMPG), cityDiff = diff(avgCMPG)) %>% mutate(mostEfficientH =
ifelse(highwayDiff < 0, "Auto","Man"))
#If automatic is more efficient than manual than diff is negative, otherwise vice
versa
count = c(length(which(proportion$mostEfficientH ==
"Auto")),length(which(proportion$mostEfficientH == "Man")) )
label = c("Auto","Man")
pct <- round(count/sum(count)*100)
label <- paste(label, pct) # add percents to labels
label <- paste(label,"% ",sep="") # ad % to labels
label <- paste(label,"(",sep="") # ad % to labels
label <- paste(label,count,sep="") # ad % to labels
label <- paste(label,")",sep="") # ad % to labels
count %>% pie(labels = label, main="Which drivetrain is more efficient for cars from
1990-94?")
```

```
proportion <- millenium_cars %>% group_by(Make,Model,Year) %>% summarize(highwayDiff =
diff(avgHMPG), cityDiff = diff(avgCMPG)) %>% mutate(mostEfficientH =
ifelse(highwayDiff < 0, "Auto","Man"))
#If automatic is more efficient than manual than diff is negative, otherwise vice
versa
count = c(length(which(proportion$mostEfficientH ==
"Auto")),length(which(proportion$mostEfficientH == "Man")) )
label = c("Auto","Man")
pct <- round(count/sum(count)*100)
label <- paste(label, pct) # add percents to labels
label <- paste(label,"% ",sep="") # ad % to labels
label <- paste(label,"(",sep="") # ad % to labels
label <- paste(label,count,sep="") # ad % to labels
label <- paste(label,")",sep="") # ad % to labels
count %>% pie(labels = label, main="Which drivetrain is more efficient for cars from
2004-05?")

proportion <- new_cars %>% group_by(Make,Model,Year) %>% summarize(highwayDiff =
diff(avgHMPG), cityDiff = diff(avgCMPG)) %>% mutate(mostEfficientH =
ifelse(highwayDiff < 0, "Auto","Man"))
#If automatic is more efficient than manual than diff is negative, otherwise vice
versa
count = c(length(which(proportion$mostEfficientH ==
"Auto")),length(which(proportion$mostEfficientH == "Man")) )
label = c("Auto","Man")
pct <- round(count/sum(count)*100)
label <- paste(label, pct) # add percents to labels
label <- paste(label,"% ",sep="") # ad % to labels
label <- paste(label,"(",sep="") # ad % to labels
label <- paste(label,count,sep="") # ad % to labels
label <- paste(label,")",sep="") # ad % to labels
count %>% pie(labels = label, main="Which drivetrain is more efficient for cars from
2017?")


all_cars <- autoManPair %>% filter(Transmission.Type %in% c("AUTOMATIC","MANUAL")) %>%
arrange(Year)
proportion <- all_cars %>% group_by(Year,Transmission.Type) %>% summarise(avgHMPG =
mean(avgHMPG),avgCMPG = mean(avgCMPG))
proportion %>% ggplot(data = .) + geom_point(mapping = aes(x = Year, y = avgHMPG,
color = Transmission.Type)) + geom_line(mapping = aes(x = Year, y = avgHMPG, color =
Transmission.Type)) + labs(y= "Average Highway MPG", x = "Year")#If automatic is more
efficient than manual than diff is negative, otherwise vice versa
```

Appendix F : Code for Model 1 (Random Forest)

```
# after dataframe manipulations in Appendix B
# remaining df is all data except for the holdout data

library(randomForest)
# set seed for reproducible results
set.seed(1)
# Randomly sample training and test set
names(holdout_df) <- str_replace(string=names(holdout_df), pattern="High-Performance",
replacement="high_performance")
names(remaining_df) <- str_replace(string=names(remaining_df),
pattern="High-Performance", replacement="high_performance")

remaining_size <- nrow(remaining_df)
training_index <- sample(remaining_size,floor(remaining_size*0.7))
train_df <- remaining_df[training_index,]
test_df <- remaining_df[-training_index,]

# Training the model
# Use TuneRF to generate plot for finding optimal m_try value
# removing the 16th column as it is the MSRP we do not want to use MSRP to predict
# MSRP
t <- tuneRF(train_df[,-16],train_df$MSRP,
            plot = TRUE,stepFactor = 7,
            trace = TRUE,ntreeTry = 1000)
# ------------------------------ ORIGINAL MODEL --------------------------------
# naive run to get optimal parameters
# removing 27th col as it is N/A category
rf_model <- randomForest(MSRP ~ .,
                         data = remaining_df[-27],
                         subset = training_index)

# plotting the random forest model gives plot to find optimal ntree value
plot(rf_model)

# MSE for training set can be found by printing the rf_model
print(rf_model)

# Run model on test data now
yhat_rf <- predict(rf_model,
                   newdata = test_df)
# mutate a column to store predictions
# store in a new df; dont want to mutate test_df
test <- test_df %>% mutate(yhat_rf = yhat_rf)

# get test MSE
test %>% summarise(MSE_test = mean((yhat_rf - MSRP)^2))

# get the variable importance plot
varImpPlot(rf_model, sort = TRUE , n.var = 26 , main = "Variable Importance")

# ------------------------------ FINAL MODEL --------------------------------
# This rf model will have the optimal parameters, and important variables unlike the
# original model

# remove unimportant variables variables from train and test df
train_df <- select(train_df,-Engine_Fuel_Type, -Diesel, -Hatchback, -Factory_Tuner,
-Flex_Fuel)

test_df <- select(test_df,-Engine_Fuel_Type, -Diesel, -Hatchback, -Factory_Tuner,
-Flex_Fuel)
```

```
remaining_df2 <- select(remaining_df,-Engine_Fuel_Type, -Diesel, -Hatchback,
-Factory_Tuner,  -Flex_Fuel)



# create final rf model with optimal parameters and imp variables

rf_model <- randomForest(MSRP ~ .,
                         data = remaining_df2[-22],
                         subset = training_index,
                         n_tree = 200,
                         m_try = 8)

# MSE for training set can be found by printing the rf_model
print(rf_model)

# Run model on test data now
yhat_rf2 <- predict(rf_model,
                    newdata = test_df)
# mutate a column to store predictions
# store in a new df; dont want to mutate test_df
test <- test_df %>% mutate(yhat_rf2 = yhat_rf2)

# get test MSE
test %>% summarise(MSE_test = mean((yhat_rf2 - MSRP)^2))

#  ------------------------------HOLDOUT DATA--------------------------------
# holdout_df has the holdout test data
# need to match the number of variables the model was trained on

holdout_df <- select(holdout_df,-Engine_Fuel_Type, -Diesel, -Hatchback,
-Factory_Tuner,  -Flex_Fuel)

# using the final random forest model
yhat_rf_holdout <- predict(rf_model,
                           newdata = holdout_df)

# mutate a column to store predictions
# store in a new df; dont want to mutate test_df
test <- holdout_df %>% mutate(yhat_rf_holdout = yhat_rf_holdout)

# get holdout test MSE
test %>% summarise(MSE_test = mean((yhat_rf_holdout - MSRP)^2))
```

Appendix G : Code for Model 2 (Linear Model)

```
remaining_df <- model_df[-holdout_index,] %>% filter(Transmission_Type !=
"DIRECT_DRIVE")
remaining_size <- nrow(remaining_df)
train_index <- sample(remaining_size,floor(remaining_size*0.7))
train_df <- remaining_df[train_index,]
test_df <- remaining_df[-train_index,]

mod1 <- train_df %>% lm(MSRP ~ Year + Engine_HP + Engine_Cylinders +
                          Number_of_Doors+ highway_MPG + city_mpg + Popularity, data =
.)
mod2 <- train_df %>% lm(MSRP ~ Make + Year + Engine_HP + Engine_Cylinders +
Transmission_Type +
                          Driven_Wheels + Number_of_Doors + Vehicle_Size +
highway_MPG, data = .)
mod3 <- train_df %>% lm(MSRP ~ Year + Engine_HP + Engine_Cylinders + highway_MPG +
Transmission_Type
                        + Crossover + Diesel + Exotic + Luxury +
`High-Performance`+Performance +`Factory Tuner`+
                          `Flex Fuel`+Hatchback + Hybrid + Driven_Wheels, data = .)

train_pred <- predict(
  mod1,
  type = "response"
)


train_e <- train_df$MSRP - train_pred
train_mse <- mean(train_e ^ 2,na.rm = TRUE)

test_pred <- predict(
  mod1,
  type = "response",
  newdata = test_df
)

test_e <- test_df$MSRP - test_pred
test_mse <- mean(test_e ^ 2,na.rm = TRUE)

train_pred <- predict(
  mod2,
  type = "response"
)


train_e <- train_df$MSRP - train_pred
train_mse <- mean(train_e ^ 2,na.rm = TRUE)

test_pred <- predict(
  mod2,
  type = "response",
  newdata = test_df
)

test_e <- test_df$MSRP - test_pred
test_mse <- mean(test_e ^ 2,na.rm = TRUE)


train_pred <- predict(
  mod3,
  type = "response"
)
```

```
train_e <- train_df$MSRP - train_pred
train_mse <- mean(train_e ^ 2,na.rm = TRUE)

test_pred <- predict(
  mod3,
  type = "response",
  newdata = test_df
)

test_e <- test_df$MSRP - test_pred
test_mse <- mean(test_e ^ 2,na.rm = TRUE)


test_pred <- predict(
  mod3,
  type = "response",
  newdata = holdout_df
)

test_e <- holdout_df$MSRP - test_pred
test_mse <- mean(test_e ^ 2,na.rm = TRUE)
```

Appendix H : Code for Model 3 (Bagging Model)

```
#This assumes that you have model_df, test_df, and holdout_df from earlier models.

library(randomForest)
names(model_df) <- make.names(names(model_df))
names(test_df) <- make.names(names(test_df))
names(holdout_df) <- make.names(names(holdout_df))

# Make 4 random forests, using train index as subset, use the test_df as the predict
to get the lowest MSE of the 4, and then use the best one for the holdout.
set.seed(1)
bag_newer_cars <- randomForest(MSRP ~ .,
                           data = model_df,
                           subset = train_index,
                           mtry = 27,
                           importance = TRUE)
plot(bag_newer_cars,main = "Bagging Results")
bag_newer_cars
yhat_bag <- predict(bag_newer_cars,
                    newdata = test_df)
test_df <- test_df %>% mutate(yhat_bag = yhat_bag)

test_df %>%
  ggplot(aes(yhat_bag, MSRP)) +
  geom_point() +
  geom_abline(slope = 1, intercept = c(0, 0))
test_df %>% summarise(MSE_bag = mean((yhat_bag - MSRP)^2, na.rm = TRUE))
MSE_bag_holdout <- test_df %>% summarise(MSE_bag = mean((yhat_bag - MSRP)^2, na.rm =
TRUE))

plot(bag_newer_cars,main = "Bagging Results")

yhat_bag <- predict(bag_newer_cars,
                    newdata = holdout_df)
holdout_df <- holdout_df %>% mutate(yhat_bag = yhat_bag)

holdout_df %>%
  ggplot(aes(yhat_bag, MSRP)) +
  geom_point() +
  geom_abline(slope = 1, intercept = c(0, 0))
holdout_df %>% summarise(MSE_bag = mean((yhat_bag - MSRP)^2, na.rm = TRUE))
MSE_bag_holdout <- holdout_df %>% summarise(MSE_bag = mean((yhat_bag - MSRP)^2, na.rm
= TRUE))

varImpPlot(bag_newer_cars)
which.min(bag_newer_cars$mse)
```

Appendix I: Code for Model 4 (Neural Network)

```
# normalize function
normalize <- function(x) {
  return((x - mean(x)) / sd(x))
}

## Split into 10 folds
indx <- rep(1:10, length.out = nrow(remaining_df))
set.seed(1)
indx <- sample(indx)

# select relevant columns
nn_df <- remaining_df %>% select(MSRP, Year, Engine_HP, Engine_Cylinders, highway_MPG,
city_mpg, Vehicle_Size) %>%
  mutate(Vehicle_Size = if_else(Vehicle_Size == "Compact", 1, if_else(Vehicle_Size ==
"Midsize", 2, 3)))
original_msrp <- nn_df$MSRP

# normalize the data
nn_df[,1:7] <- lapply(nn_df[,1:7], normalize)
networks <- rep(NULL, 10)

# mean and sd for de-normalizing
sd_msrp <- sd(original_msrp)
mean_msrp <- mean(original_msrp)

# iterate through models with different amount of hidden neurons -- 10-fold
cross-validation
total_train_errors <- rep(0,10)
total_test_errors <- rep(0,10)
for (j in 1:9) {
  test_error = rep(0,10)
  train_error = rep(0,10)

  # average MSE of each fold in 10-fold cross validation
  for (i in 1:10) {

    # separate test and training data
    indexes <- indx != i
    train_df <- nn_df[indexes,]
    test_df <- nn_df[!indexes,]

    # the model:
    nn <- neuralnet(MSRP ~ Year + Engine_HP + Vehicle_Size + highway_MPG, hidden =
c(j), data = train_df, stepmax = 1e+06, linear.output = TRUE, lifesign = "minimal",
threshold = 0.05)

    # predictions
    training_predictions <- neuralnet::compute(nn, train_df)
    testing_predictions <- neuralnet::compute(nn, test_df)

    # predictions in un-normalized form
    msrp_training_predictions <- training_predictions$net.result * sd_msrp + mean_msrp
    msrp_testing_predictions <- testing_predictions$net.result * sd_msrp + mean_msrp

    # MSE for the fold
    train_error[i] <- mean((msrp_training_predictions - original_msrp[indexes])^2)
    test_error[i] <- mean((msrp_testing_predictions - original_msrp[!indexes])^2)
  }

  # mean of folds
  total_train_errors[j] <- mean(train_error)
  total_test_errors[j] <- mean(test_error)
}

# plot errors
error_df <- tibble(Test = total_test_errors, Train = total_train_errors, h = 1:10)
error_df %>% ggplot() + geom_line(mapping = aes(x = h, y = Test), color = "Red", ) +
  geom_line(mapping = aes(x = h, y = Train), color = "Blue")

# finalize model using decreased threshold:
set.seed(1)
train_df <- sample(nn_df, floor(size * 0.7))
nn <- neuralnet(MSRP ~ Year + Engine_HP + Vehicle_Size + highway_MPG, hidden = c(9),
data = train_df, linear.output = TRUE, lifesign = "full", threshold = 0.01, stepmax =
1e+06)

# normalize holdout test set
holdout_df <- holdout_df %>%
  select(MSRP, Year, Engine_HP, Engine_Cylinders, highway_MPG, city_mpg, Vehicle_Size)
%>%
  mutate(Vehicle_Size = if_else(Vehicle_Size == "Compact", 1, if_else(Vehicle_Size ==
"Midsize", 2, 3)))
holdout_normalized <- holdout_df
holdout_normalized[,1:7] <- lapply(holdout_df[,1:7], normalize)

# Calculate predictions
holdout_predictions <- neuralnet::compute(nn, holdout_normalized)
holdout_msrp <- holdout_predictions$net.result * sd_msrp + mean_msrp

# Get error for holdout set
holdout_mse <- mean((holdout_df$MSRP - holdout_msrp)^2)

# graph predictions
holdout_plot_data <- tibble(holdout_df$MSRP, holdout_msrp) %>%
  rename(Predicted = `holdout_msrp`, Actual = `holdout_df$MSRP`)
holdout_plot_data %>% ggplot() +
  geom_point(mapping=aes(x=Actual, y=Predicted)) + geom_abline(intercept=0, slope=1,
color='Blue') +
  geom_abline(intercept=0, slope=1.05, color='Red') + geom_abline(intercept=0,
slope=0.95, color='Red')

# graphical for the neural network
plot(nn)
```