Class 8: Interpretable Machine Learning

BUS 696

Prof. Jonathan Hersh

Class 8: Announcements

1. Pset 3 Due Oct 26
2. Exam Next Week
3. Additional Office Hours?

Midterm Exam details

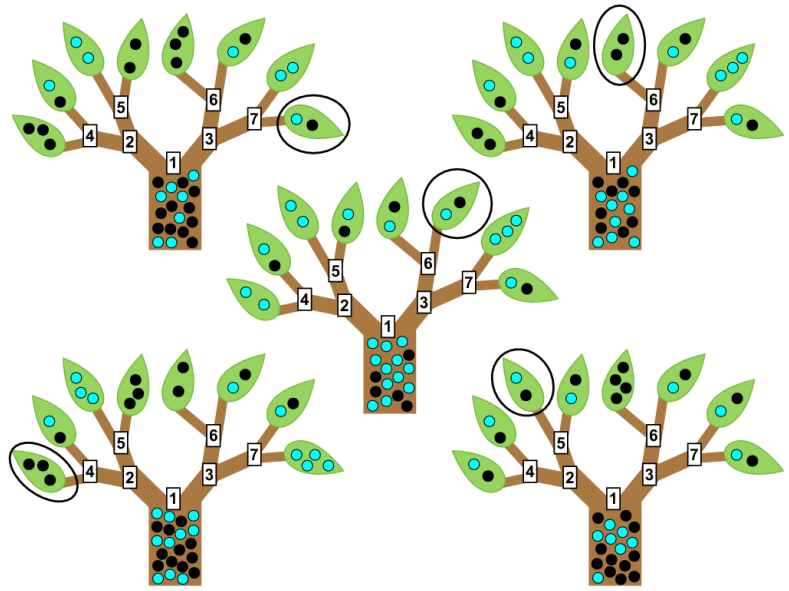
1. Exam: Posted 7pm on Wednesday, due 7pm Friday
2. Structured much like the problem sets
    - Mix of conceptual and coding questions
3. Open note, open internet, BUT DO NOT COPY CODE FROM ANYWHERE FOUND ONLINE OR FROM YOUR PEERS
4. How to study?
    - Read slides and textbook and ensure you know all core concepts
    - Practice running and interpreting models, producing output
    - Prepare code snippets to do common tasks

Class 8: Outline

1. Random Forest Review
2. Why Model Interpretability Redux
3. Local and Global Model Interpretability
4. Interpreting Random Forests with Random Forest Explainer (on exam)
5. Interpreting Random Forests Lab
6. Automated Machine Learning With Caret (not on exam)
7. Exam Review Qs

Random Forests

- Random forests are a slight trick to bagging that highly improves predictive power
- **Many trees do poorly because the stepwise greedy algorithm doesn't fully explore variable and parameter space**
- Random forests is like bagging, only each time a split in a tree is considered, a random selection of m predictors is chosen as split candidate
- A fresh set of m predictors is taken at each split.



Random Forest Model: Many Decision Trees

B bootstrap sampes of data (~ 1000)

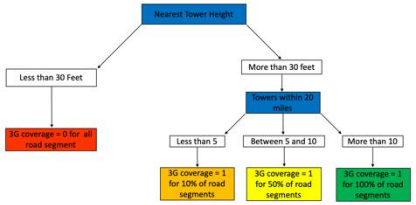
...

Bootstrap 1

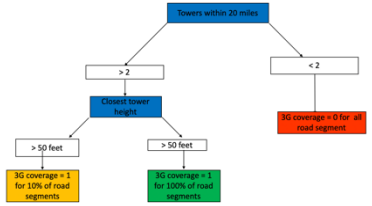
Bootstrap b

Bootstrap 1000

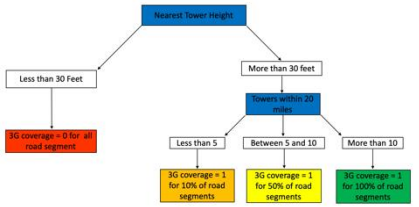
Tree 1



Tree b



Tree 1000

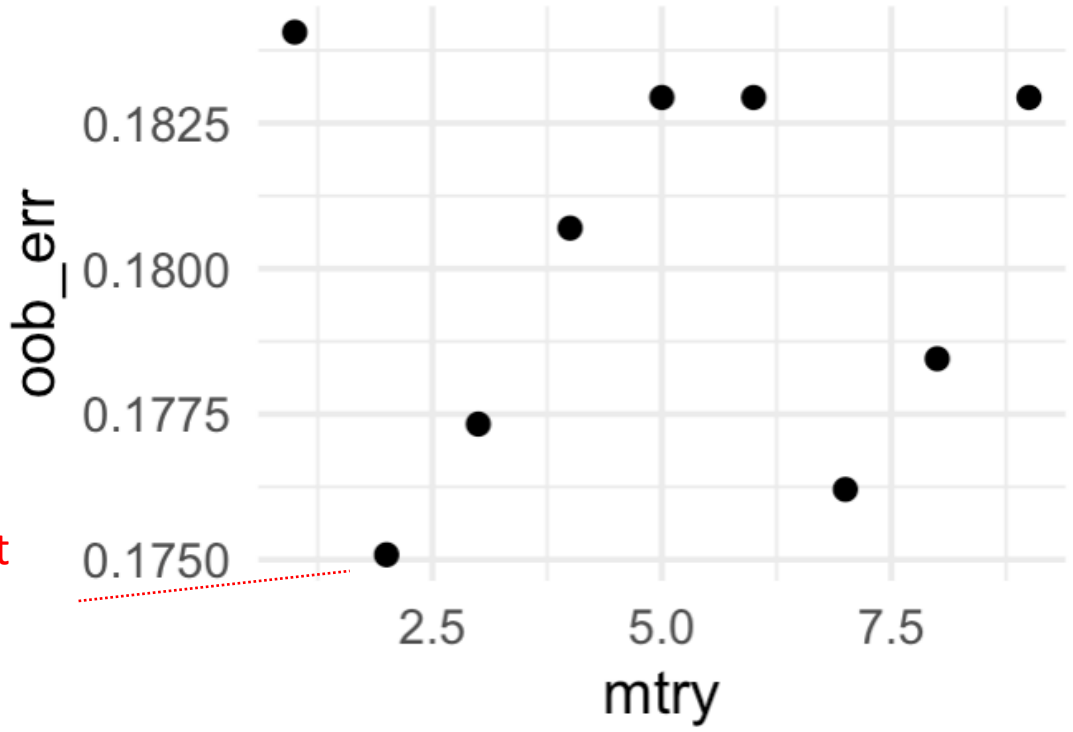


Vote road i has 3G access = 0 or 1

Vote road i has 3G access = 0 or 1

Vote road i has 3G access = 0 or 1

Cross-validate to select “mtry”

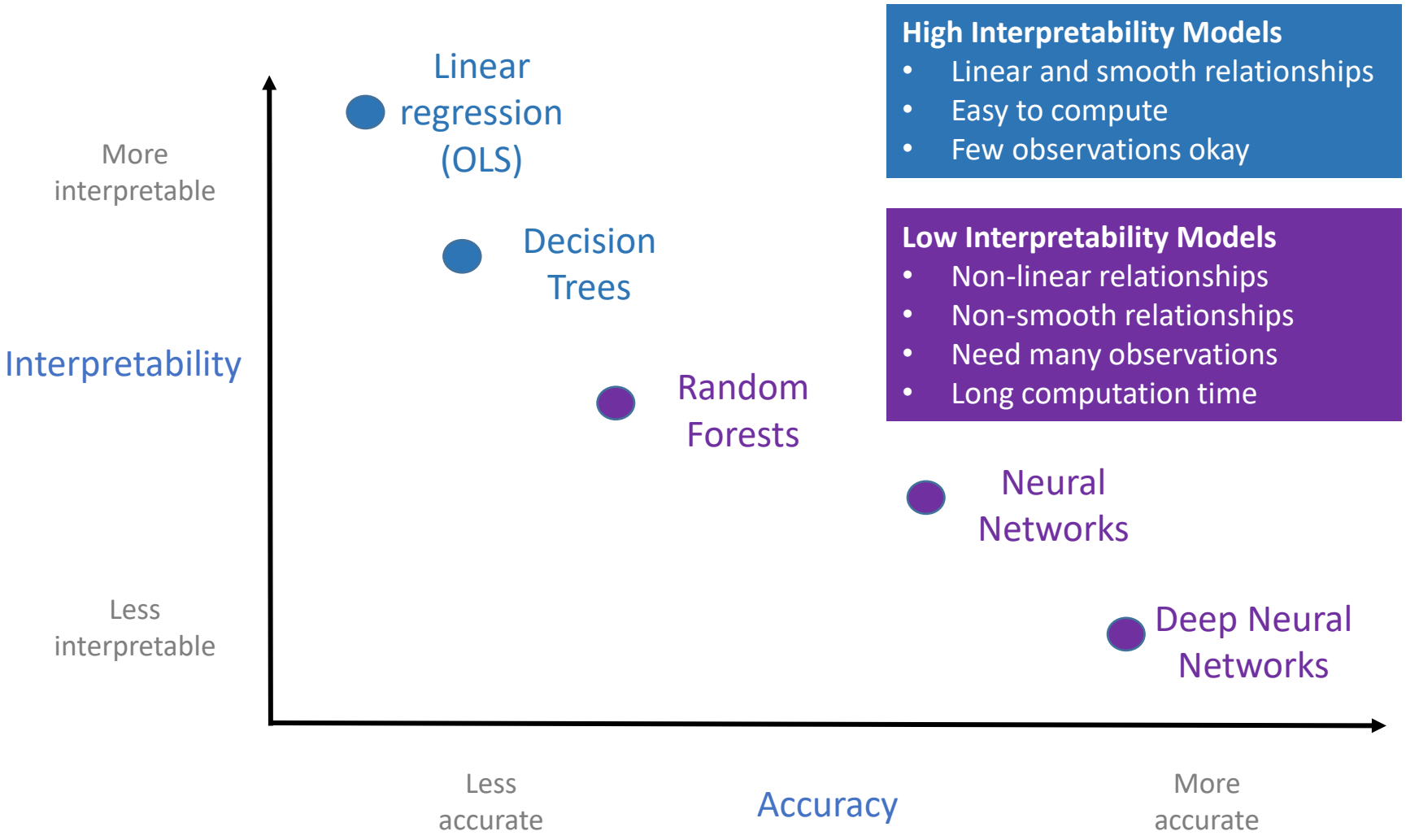


Mtry value With lowest error

- If we plot the OOB (out of bag) error against mtry, we find mtry = 2 gives us the lowest OOB error.
- Therefore the “tuned” model sets mtry = 2, and ntree = 200
- To see true out of sample fit, we use these tuned parameters to predict out of sample fit on the test set

```
results_DF <- data.frame(mtry = 1:9, oob_err)
ggplot(results_DF, aes(x = mtry, y = oob_err)) + geom_point() + theme_minimal()
```

What Is Model Interpretability?



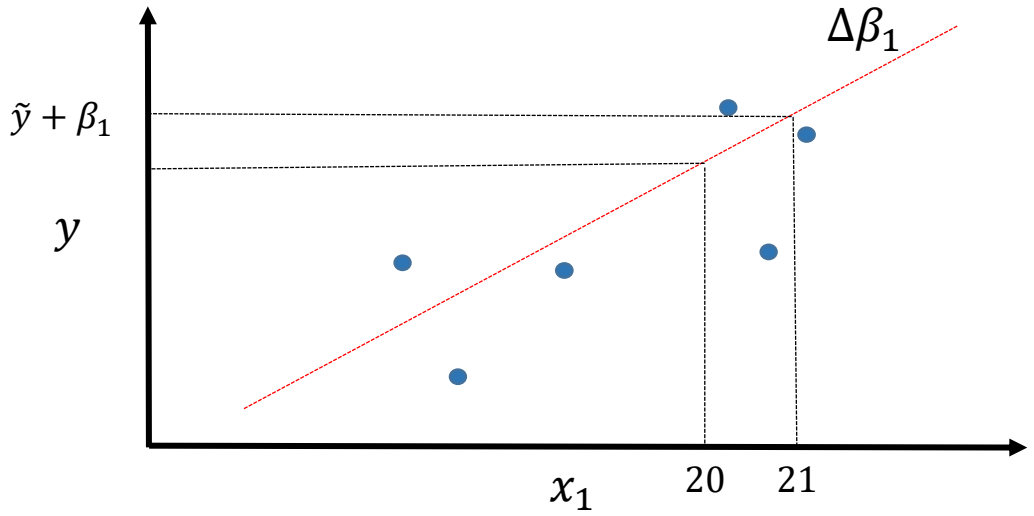
- **Model interpretability:**
  - "the degree to which a human can understand the cause of a decision" (Miller, 2017)
- The higher the interpretability, the easier it is for someone to comprehend why a decision has been made

Interpreting Linear Model Coefficients

- Linear model are easy to interpret because the impact of any predictor is the same, regardless of the range of X that we consider
- This is not the case for non-linear models.
- For ensemble models the problem is all the more difficult because we have many models to consider, each of which may be different
- So how do we interpret non-linear ensemble models?

$$y = \beta_0 + \boldsymbol{\beta_1} \cdot x_1 \ + \cdots + \beta_k \cdot x_k$$

$$?= \beta_0 + \boldsymbol{\beta_1} \cdot (x_1 + 1) \ + \cdots + \beta_k \cdot x_k$$



Local and Global Interpretability

- **Global Interpretability** refers to methods that help understand why a model or algorithm makes certain decisions on average. Global interpretability is based on a holistic view of variables, parameters and structure of the model.

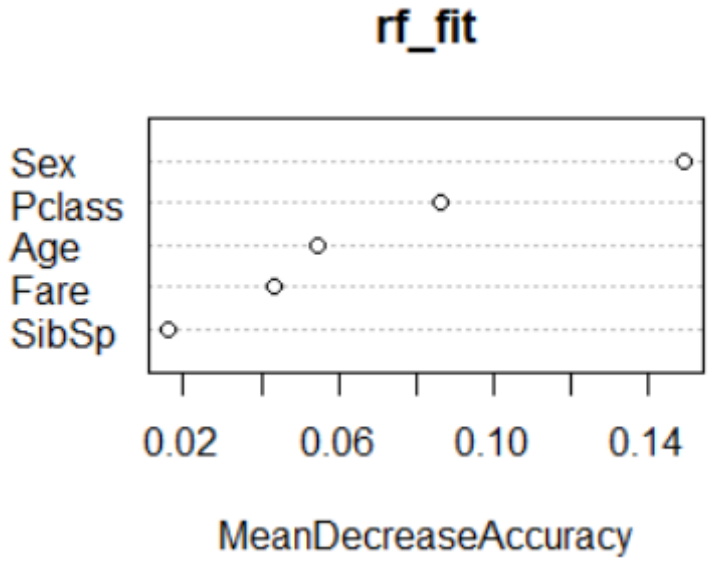
Local Interpretability refers to methods that zoom in on a particular observation and explain why the algorithm chose a result for that specific observations.

Global Interpretability: Permutation Feature Importance

- **Permutation feature importance**: take each variable, and successively replace it with random noise. How much does the prediction accuracy decline?
- The function importance() in the randomForest package calculates feature importance. MeanDecreaseAccuracy = % of observations mis-classified with this variable removed
- An additional 14% of observations would be mis-classified without the variable sex
- varImpPlot plots these variable importance measures

```
> importance(rf_fit, type = 1, scale = FALSE)
       MeanDecreaseAccuracy
Pclass           0.08607396
Sex              0.14881430
Age              0.05459818
SibSp            0.01619358
Fare             0.04353016
```

```
varImpPlot(rf_fit, type = 1, scale = FALSE)
```

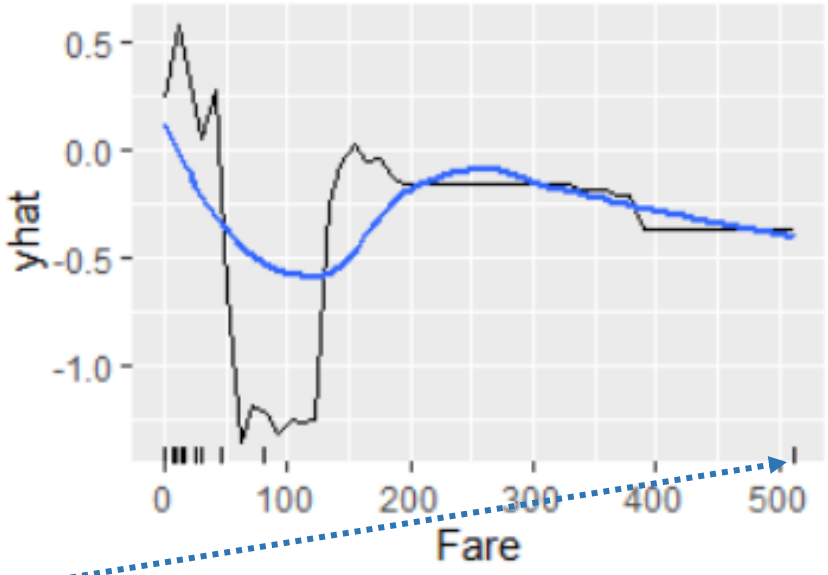


Very good (free!) book on interpretable machine learning with R
https://christophm.github.io/interpretable-ml-book/

Global Interpretability: Partial Dependence Plots (PDPs)

- How do model predictions change as we vary certain X predictors, holding everything else fixed?
- Note this is not a causal interpretation, but merely explaining how the model makes decisions locally
- We pass the fitted model to the function partial that creates the plot showing how the ticket fare paid changes the predicted probability of survival (holding everything else fixed
- Black: actual predicted probability. Blue: smoothed line. The plot shows the predicted probability decreases sharply with fare paid, then increases.
- "rug" plot shows values of actual data in the "prediction support". Note only one observation for very high fare values = unstable prediction

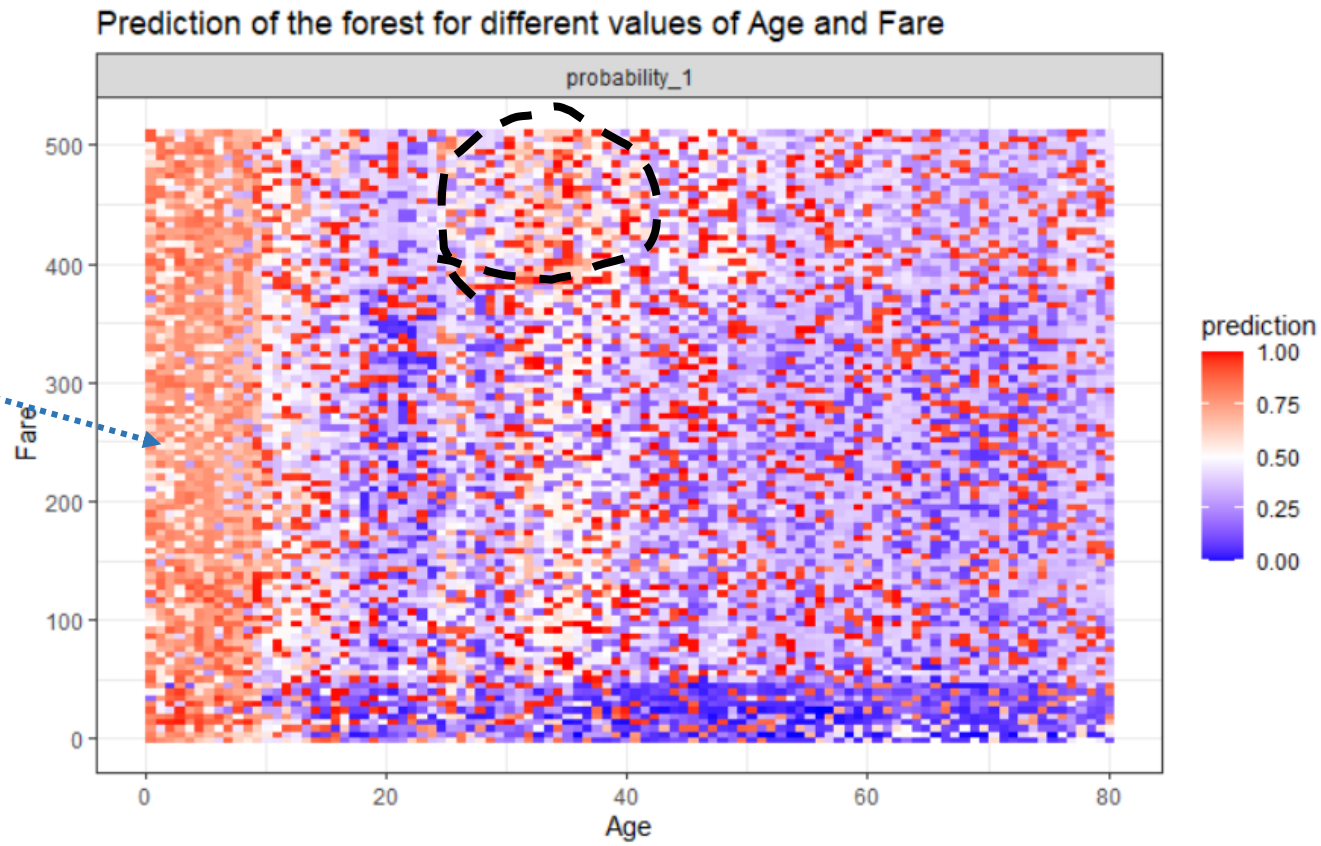
```
library('pdp')
partial(rf_fit,
        pred.var = "Fare",
        plot = TRUE,
        rug = TRUE,
        plot.engine = "ggplot2",
        smooth = TRUE)
```



Global Interpretability: Interaction Dependence Plots

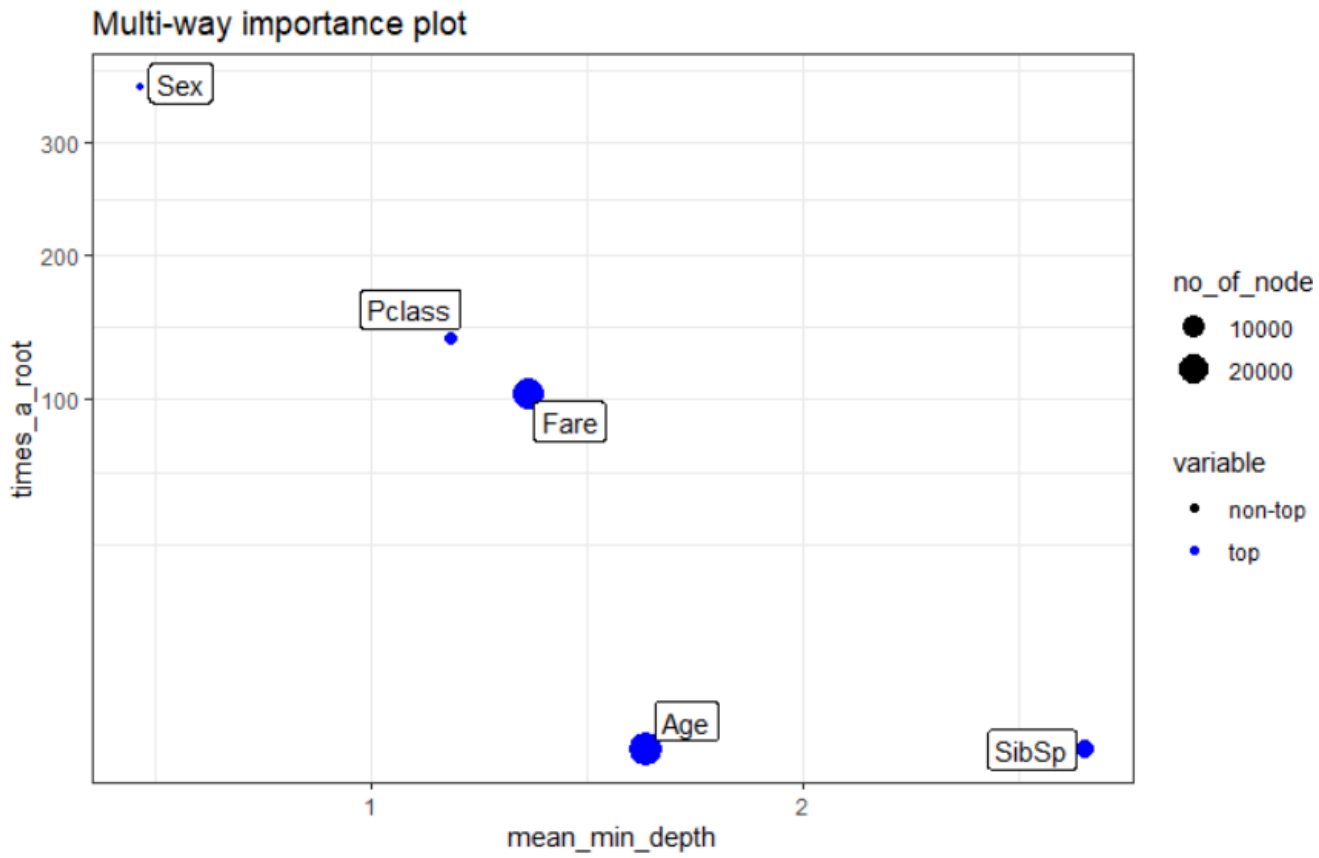
- How do variable interactions or combinations impact the predicted probability?
- The function plot_predict_interaction() will show how the predictions of the model vary for two continuous variables
- Here we see that for ages below 10 survival rate is high
- But high fare adults of middle age had high probability of survival

```
# PDP prediction interaction
library('randomForestExplainer')
plot_predict_interaction(rf_fit,
                         titanic_df,
                         "Age",
                         "Fare")
```



Meta Model Information: Root Splits and Mean Depth

```
#
plot_multi_way_importance(rf_fit,
                          size_measure = "no_of_nodes")
```



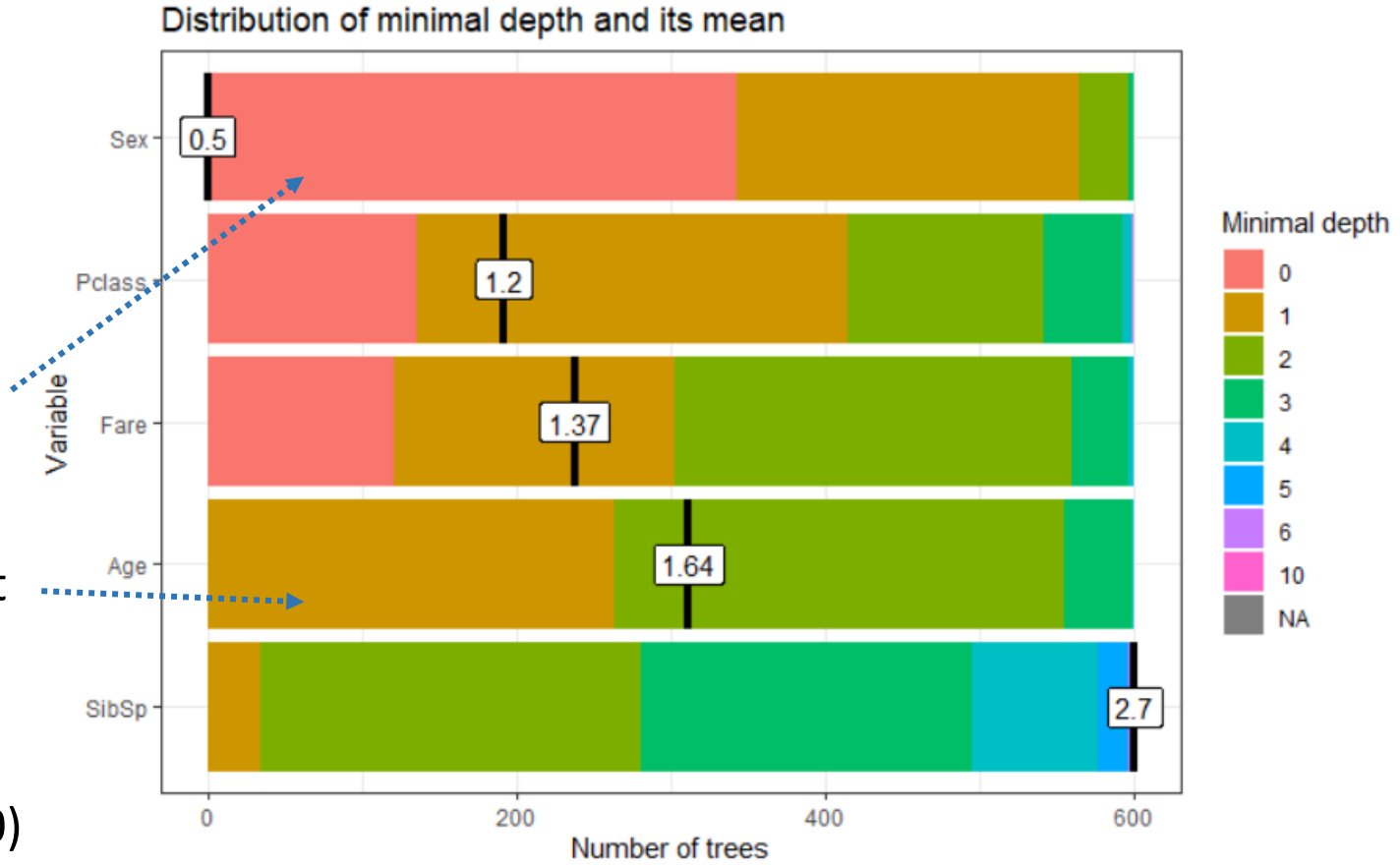
- We can see information on how many times each variable is a root against the mean depth for each variable using this plot_multi_way_importance() function

- This plot confirms sex is by far the most important variable, following by pclass and fare, then Age and sibSp

Global Interpretability: Meta Information About Ensemble Model

- Many times we want to know meta information about the model itself and its structure
- For random forest, we can explore all the individual models that comprise the random forest using the mid depth distribution plots
- Here we see sex is almost always used in the first or second split out of all of the trees
- Age is useful as the second and third split
- “Sex” has the highest average position in the trees, at an average of 0.5 (meaning half of the time it’s the first split at split 0)

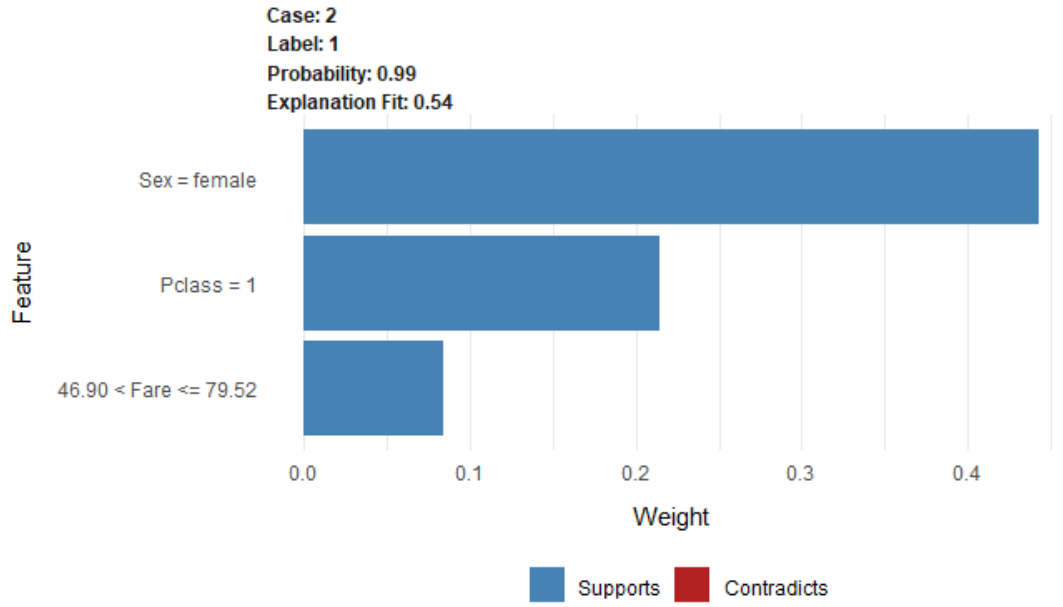
```
# min depth distribution
plot_min_depth_distribution(rf_fit)
```



Local Methods: LIME (Local Interpretable Model Agnostic Explanations)

- Can we build a simple, interpretable model around a data point and use that to explain the complicated model's decisions?
- The package lime allows us to build these, and we specify the features and complexity (permutations) of the simple model
- For one observation our "surrogate" model explains 54% of the variation in survival. The true value is 1 (survived) and the probability the model assigns is 99%.
- Feature weights show the contributions of each variable to the final prediction outcome

```
library(lime)
rf_fit<-as_classifier(rf_fit, labels = NULL)
explainer <- lime(titanic_df,
                  rf_fit,
                  quantile_bins = TRUE,
                  n_bins = 10)
explanation <- explain(titanic_df,
                       explainer,
                       n_labels = 1,
                       n_features = 5,
                       n_permutations = 100,
                       feature_select = "highest_weights")
```



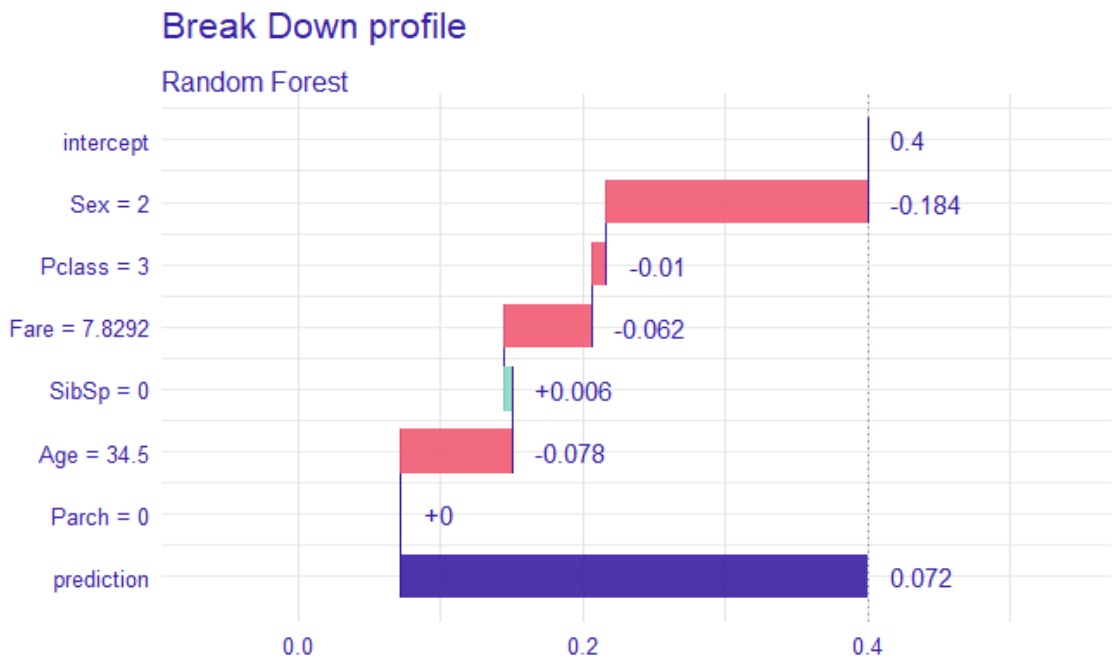
Local Methods: Shapley Values

- Shapely values are the most common local interpretability methods.
- Shapely value comes from a game theory. Each feature is a player in a “game” where the players have to determine how to split the proceeds from some communal winnings.
- The details aren’t necessary to understand, but know, for one observation, we can see the successive contribution of features to the final prediction.
- Here for the first observation in the test, the final prediction is 7.2%, and the contributions to the total are shown in the plot to the right. Age adds

```
library('DALEX')
explain_rf <- DALEX::explain(model = rf_fit,
                             data = titanic_df[, -1],
                             y = titanic_df$Survived == 1,
                             label = "Random Forest")

predict(explain_rf, titanic_test[1,])

shap_test1 <- predict_parts(explainer = explain_rf,
                            new_observation = titanic_test[1,],
                            type = "break_down",
                            B = 25)
plot(shap_test1, show_boxplots = TRUE)
```

```
> titanic_test[1,]
# A tibble: 1 x 6
  Pclass Sex     Age SibSp Parch  Fare
  <fct>  <fct> <dbl> <int> <int> <dbl>
1 3      male   34.5     0     0  7.83
```

```
> shap_test1
                                  contribution
Random Forest: intercept             0.400
Random Forest: Sex = 2              -0.184
Random Forest: Pclass = 3           -0.010
Random Forest: Fare = 7.8292        -0.062
Random Forest: SibSp = 0             0.006
Random Forest: Age = 34.5           -0.078
Random Forest: Parch = 0             0.000
Random Forest: prediction            0.072
```



https://christophm.github.io/interpretable-ml-book/shapley.html

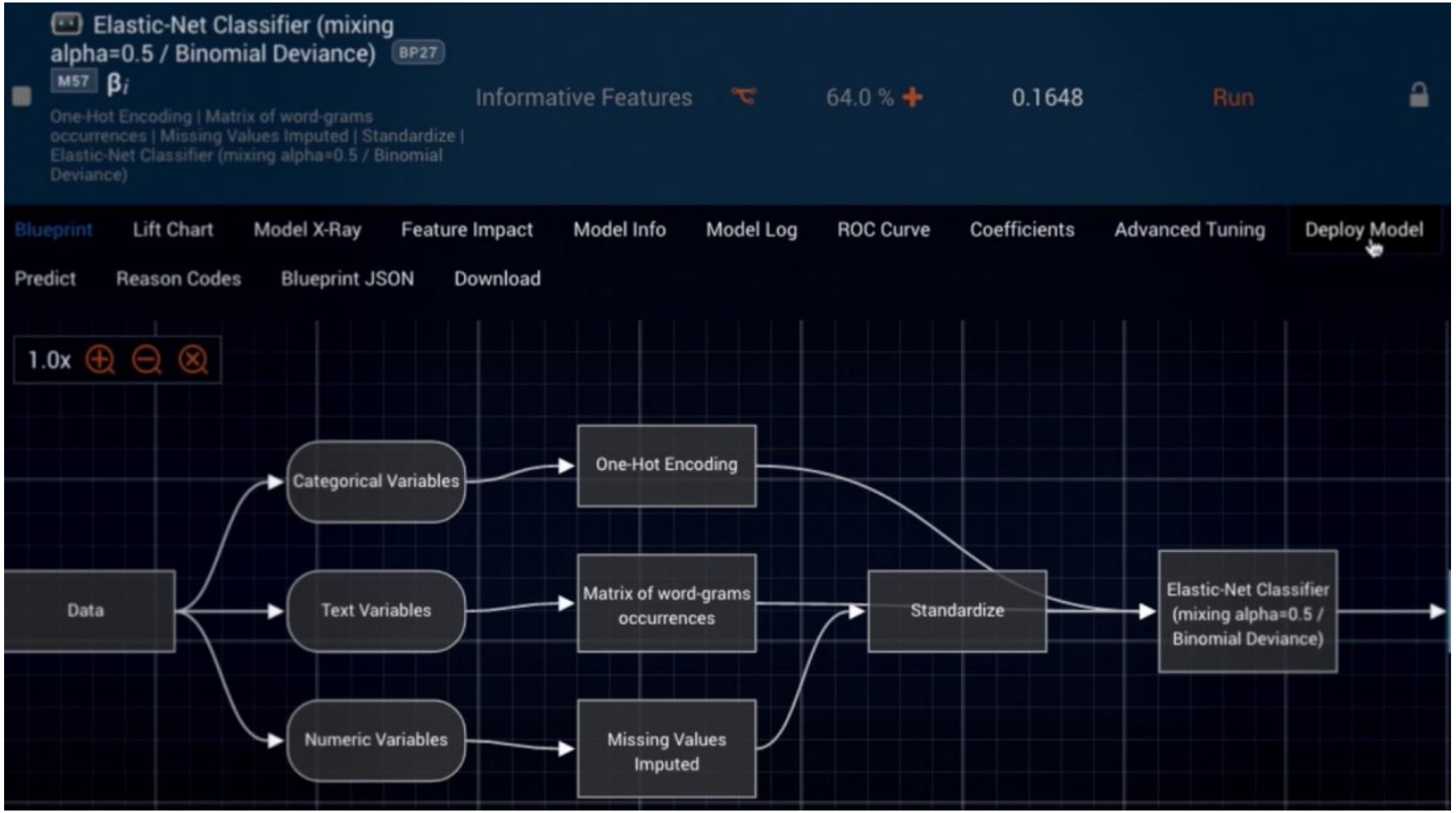
Lab

```r
# 1. Estimate the random forest model below

rf_fit2 <- randomForest(Survived ~
                          Sex + Age + SibSp + Fare,
                        data = titanic_df %>% drop_na(),
                        type = classification,
                        mtry = 2,
                        ntree = 500,
                        importance = TRUE,
                        keep.inbag = TRUE)

# 2. Calculate the variable importance using the function importance
#    What does the output indicate is the most important variable is?

# 3. Produce a variable importance plot

# 4. Produce a partial dependence plot for "Age" using parital.
#    Explain in words how Age affects the probability of survival

# 5. Interact Age with SibSp and explain the plot.
#    For what age and number of siblings is the probability of survival quite high?

# 6. Plot the mininimum depth distributions for rf_fit2 and explain the plot

# 7. Run the full explain_forest function and examine the outputed rmarkdown file
```

Class 8: Outline

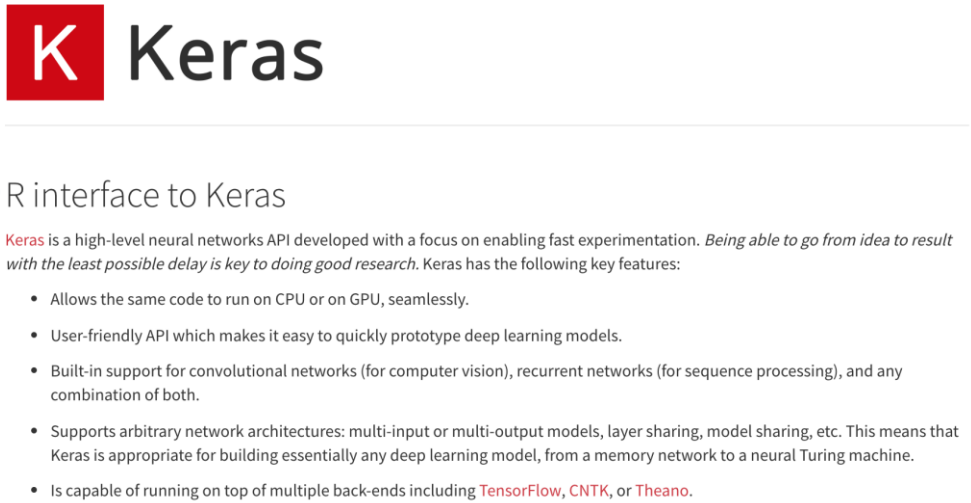
1. Random Forest Review
2. Why Model Interpretability Redux
3. Local and Global Model Interpretability
4. Interpreting Random Forests with Random Forest Explainer (on exam)
5. Interpreting Random Forests Lab
6. **Automated Machine Learning With Caret (not on exam)**
7. Exam Review Qs

Automated machine learning software: DataRobot

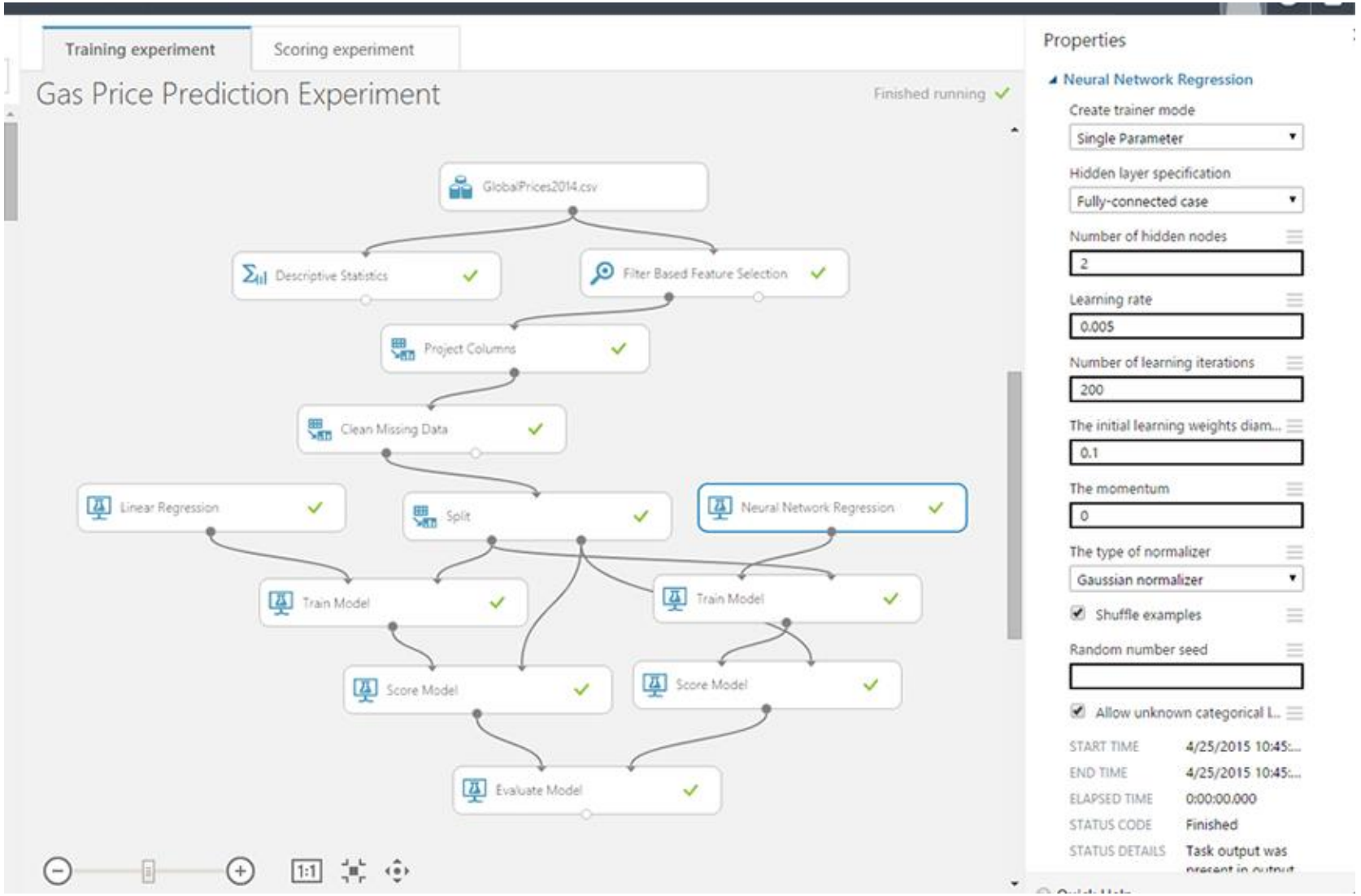


AutoML: TensorFlow & Keras

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```



Auto ML: Azure Studio



A Short Introduction to the caret Package

The **caret** package (short for Classification And REgression Training) contains functions to streamline the model training process for complex regression and classification problems. The package utilizes a number of R packages but tries not to load them all at package start-up (by removing formal package dependencies, the package startup time can be greatly decreased). The package "suggests" field includes 30 packages. **caret** loads packages as needed and assumes that they are installed. If a modeling package is missing, there is a prompt to install it.

Install **caret** using

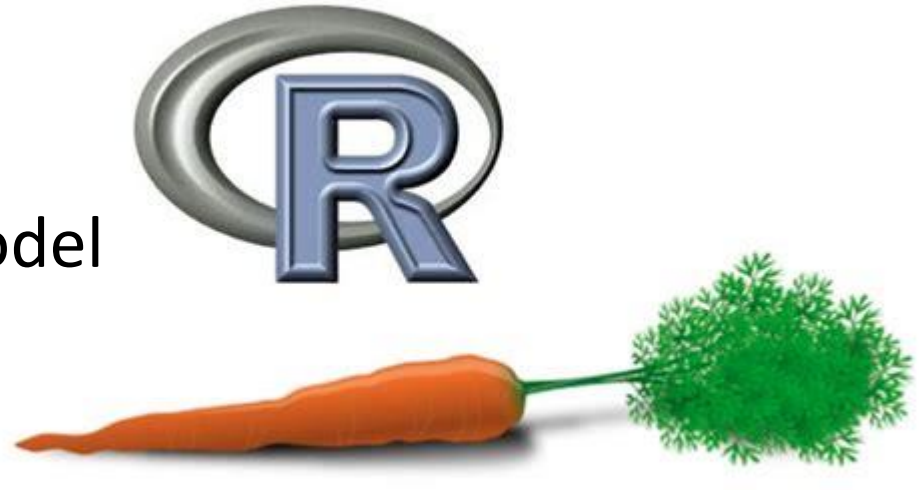
```
install.packages("caret", dependencies = c("Depends", "Suggests"))
```

to ensure that all the needed packages are installed.

The **main help pages** for the package are at https://topepo.github.io/caret/ Here, there are extended examples and a large amount of information that previously found in the package vignettes.

Why caret?

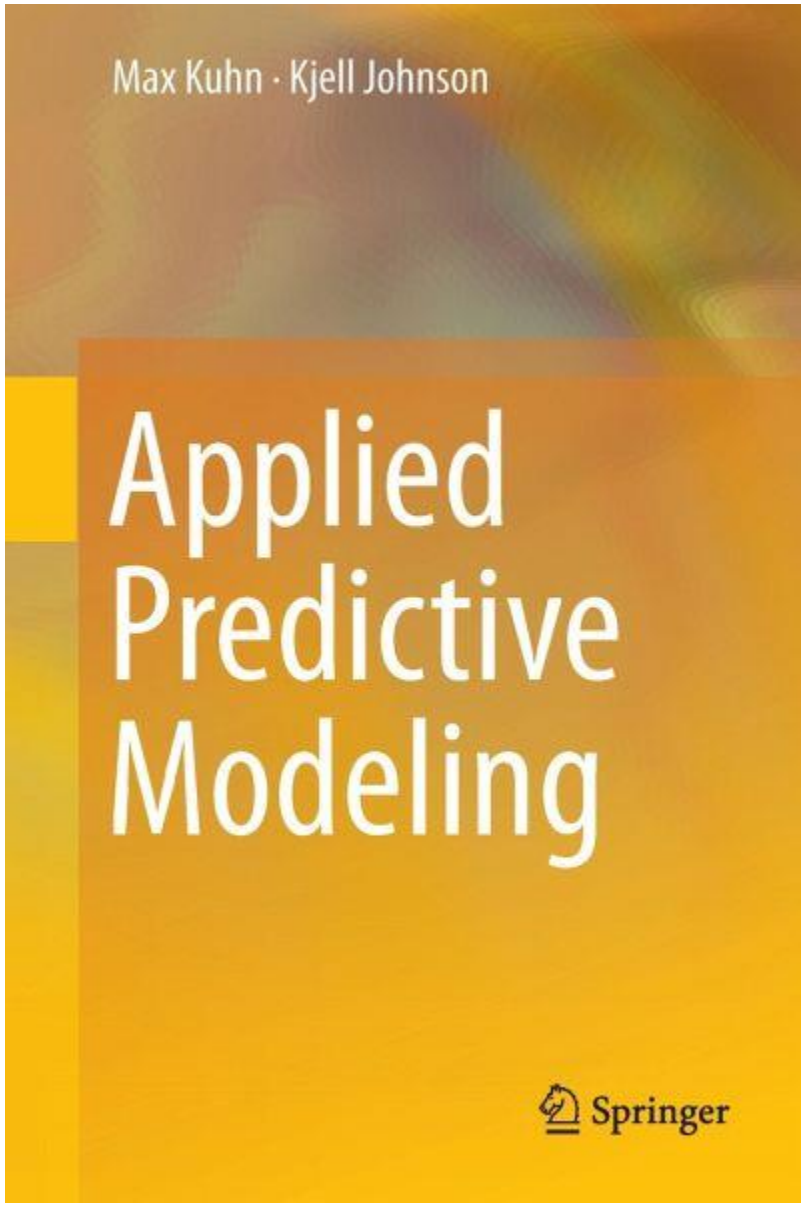
- In short, caret streamlines the model building process and
- Provides a unified formula interface
- Evaluates, using resampling, the effect of model parameters on validation performance
- Chooses optimal parameters on OOS performance
- HUGE list of models available



Caret Has Tools For

- Data Splitting
- Data Pre-Processing (center, scaling)
- Feature selection
- Model tuning and resampling
- Variable importance and other post-estimation diagnostics

Comprehensive guide at:
https://topepo.github.io/caret/index.html



Caret like Python scikit-learn

scikit learn Install User Guide API Examples More Go

scikit-learn

Machine Learning in Python

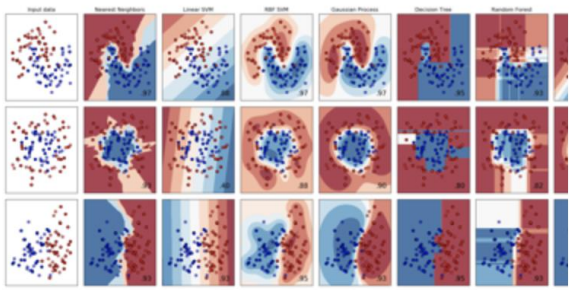
Getting Started | What's New in 0.22.1 | GitHub

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying which category an object belongs to.

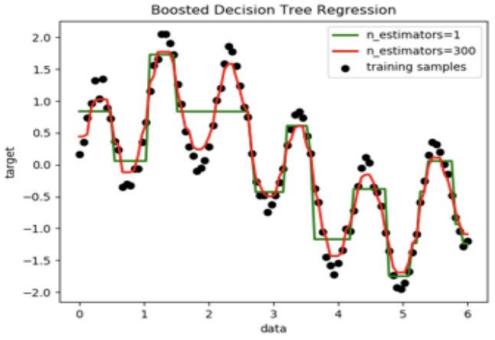
Applications: Spam detection, image recognition.
Algorithms: SVM, nearest neighbors, random forest, and more...



Regression

Predicting a continuous-valued attribute associated with an object.

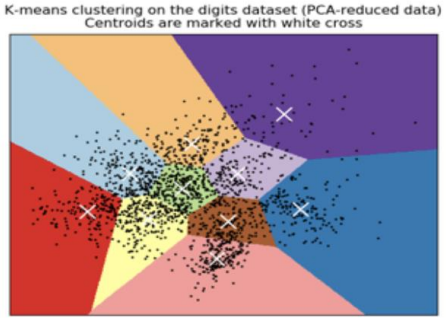
Applications: Drug response, Stock prices.
Algorithms: SVR, nearest neighbors, random forest, and more...



Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, and more...



Partial Model List

- Glmnet
- K-nearest neighbors
- Logistic regression
- Neural networks
- Random forests
- Parallel random forests
- Partial Least Squares
- Ridge Regression
- SVMs
- Weighted Least Squares
- Extreme Gradient Boosted Trees
- Bayesian Additive Regression Trees (BART)
- Bayesian GLM
- Multivariate Adaptive Regression Splines (MARS)
- Partial Least Squares
- Least Squares Support Vector Machine with Radial Basis Function Kernel
- Fuzzy Rules Using the Structural Learning Algorithm on Vague Environment

Key functions: train() and trainControl()

trainControl()

- Specifies range of parameters over which you want to estimate your model, nuances of training process

Train()

- Function that estimates your model against your data given a formula using options specified by trainControl()

Example Wage Data

Griliches {Ecdat}

Wage Datas

Description

a cross-section from 1980

number of observations : 758

observation : individuals

country : United States

Usage

```
data(Griliches)
```

rns
: residency in the southern states (first observation) ?

rns80
: same variable for 1980

mrt
: married (first observation) ?

mrt80
: same variable for 1980

smsa
: residency in metropolitan areas (first observation) ?

smsa80
: same variable for 1980

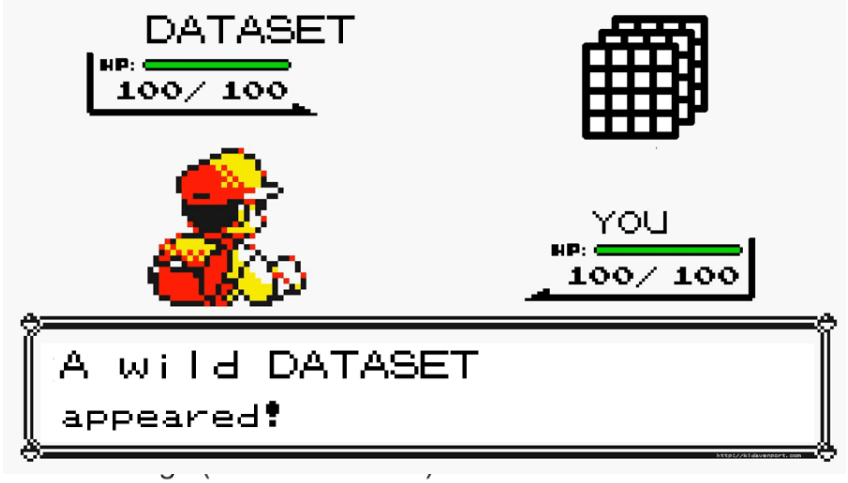
med
: mother's education in years

iq
: IQ score

kww
: score on the “knowledge of the world of work” test

year
: year of the observation

age
: age (first observation)



age80
: same variable for 1980

school
: completed years of schooling (first observation)

school80
: same variable for 1980

expr
: experience in years (first observation)

expr80
: same variable for 1980

tenure
: tenure in years (first observation)

tenure80
: same variable for 1980

lw
: log wage (first observation)

lw80
: same variable for 1980

Download Wage DF and create a formula to predict log wage in 1980

```
install.packages('Ecdat')
data(Griliches, package = "Ecdat")
wages <- Griliches
wageFormula <- lw80 ~ age80 + school80 + expr80 + iq + rns80 + mrt80 +
  smsa80 + tenure80 + med + kww
```

Example trainControl()

```
# example trainControl function
ctrl1 <- trianControl(method = "repeatedcv", repeats = 5,
                      allowParallel = TRUE)
```

train() function – parameters needed? – modelLookup('model')

```
> modelLookup('rf')
  model parameter                         label forReg forClass probModel
1    rf      mtry #Randomly Selected Predictors   TRUE     TRUE      TRUE
```

```
> modelLookup('xgbTree')
    model         parameter                          label forReg forClass probModel
1 xgbTree           nrounds         # Boosting Iterations   TRUE     TRUE      TRUE
2 xgbTree         max_depth                Max Tree Depth   TRUE     TRUE      TRUE
3 xgbTree               eta                     Shrinkage   TRUE     TRUE      TRUE
4 xgbTree             gamma       Minimum Loss Reduction   TRUE     TRUE      TRUE
5 xgbTree colsample_bytree    Subsample Ratio of Columns   TRUE     TRUE      TRUE
6 xgbTree min_child_weight Minimum Sum of Instance Weight   TRUE     TRUE      TRUE
7 xgbTree         subsample           Subsample Percentage   TRUE     TRUE      TRUE
```

See list of models
https://topepo.github.io/caret/available-models.html

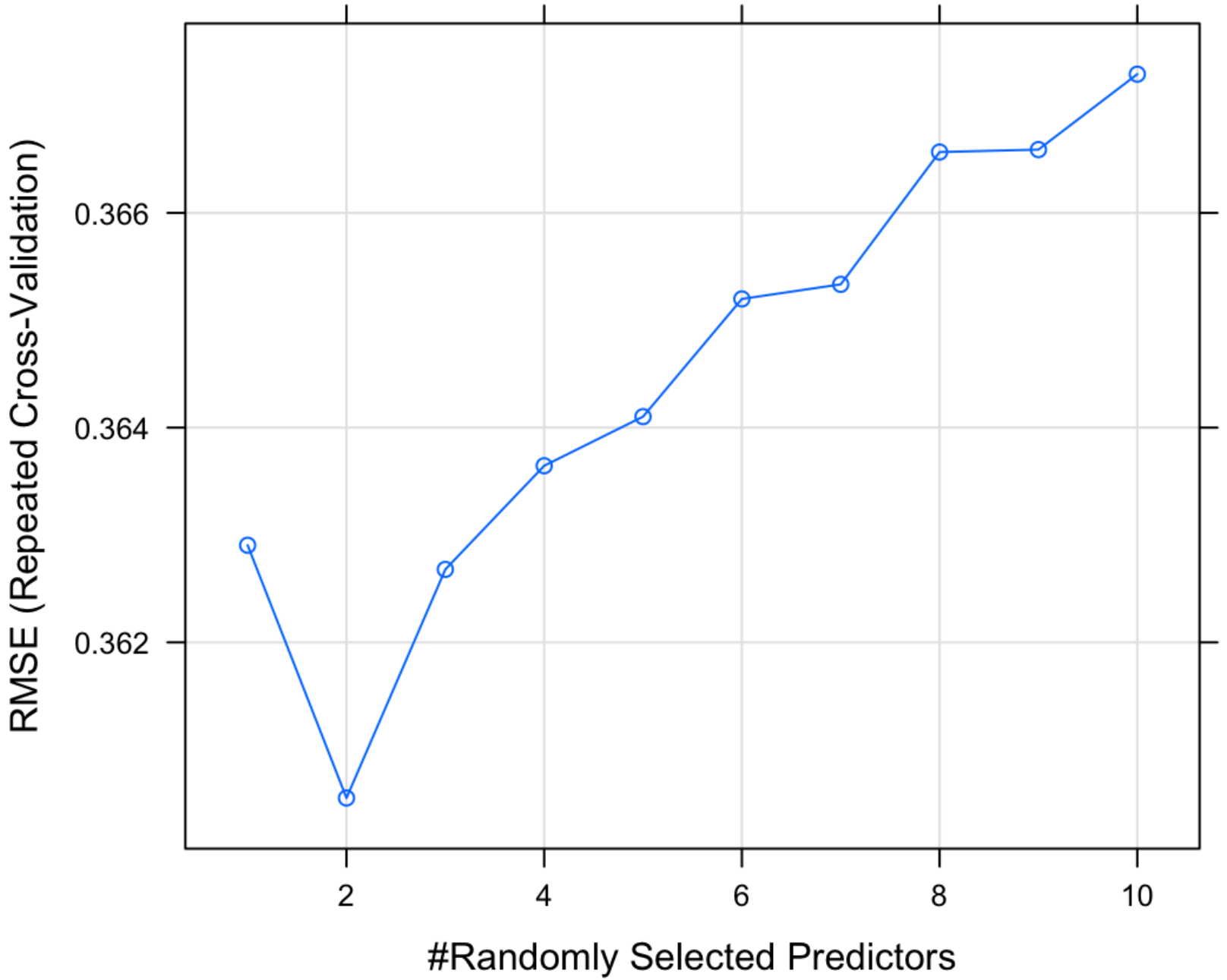
Training Grid

```r
# grid of mtry values to try
rfGrid <- expand.grid(mtry = seq(1, 10, 1))
```

```
> rfGrid
   mtry
1     1
2     2
3     3
4     4
5     5
6     6
7     7
8     8
9     9
10   10
```

```
> rfTrain
Random Forest

758 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 682, 682, 682, 683, 682, 682, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE        Cost
   1    0.3629045  0.2403263  0.2787721  1
   2    0.3605500  0.2282054  0.2751021  1
   3    0.3626793  0.2191935  0.2763876  1
   4    0.3636449  0.2163870  0.2772056  1
   5    0.3641026  0.2149631  0.2775443  1
   6    0.3651989  0.2112161  0.2780479  1
   7    0.3653347  0.2110051  0.2784213  1
   8    0.3665673  0.2071727  0.2793231  1
   9    0.3665899  0.2075322  0.2793810  1
  10    0.3672926  0.2052027  0.2796850  1

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 2.
```

```
> plot(rfTrain)
>
```



Class 8 Summary

- Global interpretability refers to methods that explain a model overall and why it makes certain predictions or decisions
- Local interpretability are methods that explain a prediction for a particular observation
- Variable importance captures the contribution each feature to the overall prediction accuracy
- Partial dependence plots create partial derivates of features and trace how these features affect the model predictions
- Meta model information can be useful such as minimum depth distribution for the ensemble of models.
- LIME and Shapley are two methods for generating local interpretations for particular observations