Introduction to Modeling in R (Notes)

Five College DataFest 2019
Evan Moore (MassMutual DSDP)

What is a model?

- A ***model*** is a mathematical function of form $y = f(X)$
  - y is some quantity of interest that we want to predict
  - X is a collection of individual observations and some information about them
  - f is our model
- Statistical models - derive insight about a real-world event while quantifying uncertainty about predictions
- Machine learning models - output the best prediction possible without regard for interpretability
- “All models are wrong, but some are useful”
  - A model is a simplified representation of a potentially complex real-world phenomenon

How does a model work?

- Models “learn” by associating a set of various observations of different feature inputs (e.g. weight, age) with a set of corresponding outputs (e.g. height) in a way such that an arbitrary ***cost function*** is minimized
  - This combination is known as the “***training set***”
- Modeling is typically one of the last steps in a data science pipeline
  - Make sure data is cleaned and relevant features are selected or processed beforehand
  - Typical feature engineering tasks include removal of missing values, normalization/standardization, or one-hot encoding depending on model type
- The process of modeling includes many potentially subjective steps that need to be justified by the modeler, so be careful!

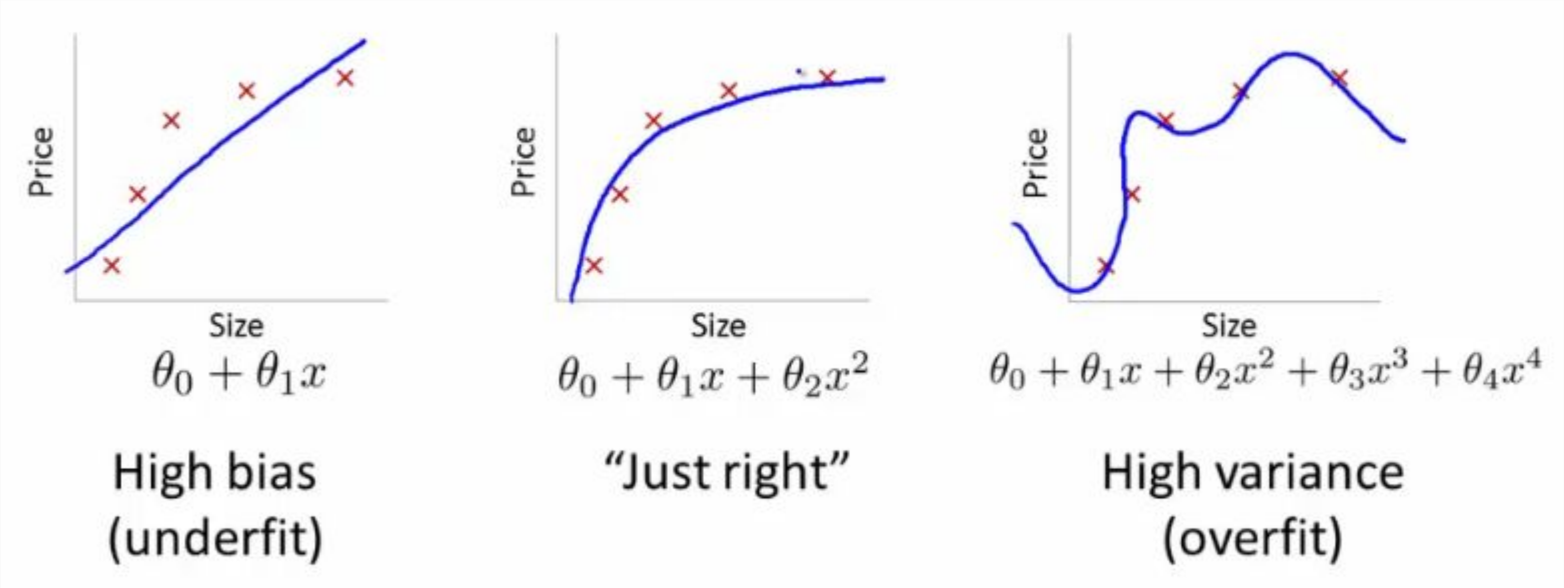
Model performance

- Models are ultimately judged on their ability to generalize to never-before seen data (frequently known as the "***test set***")
    - A train/test partition of 80%/20% is often used in practice
- A fundamental concept in modeling is the ***bias-variance*** tradeoff
    - ***Bias*** - error introduced in the specification of the form of the model
    - ***Variance*** - error introduced by inherent variability in the data itself
- Too much bias indicates the model has ***underfit*** to the data, while too much variance indicates the model is ***overfit*** to the set of observations it has seen
- The best models balance both!

Train/test code

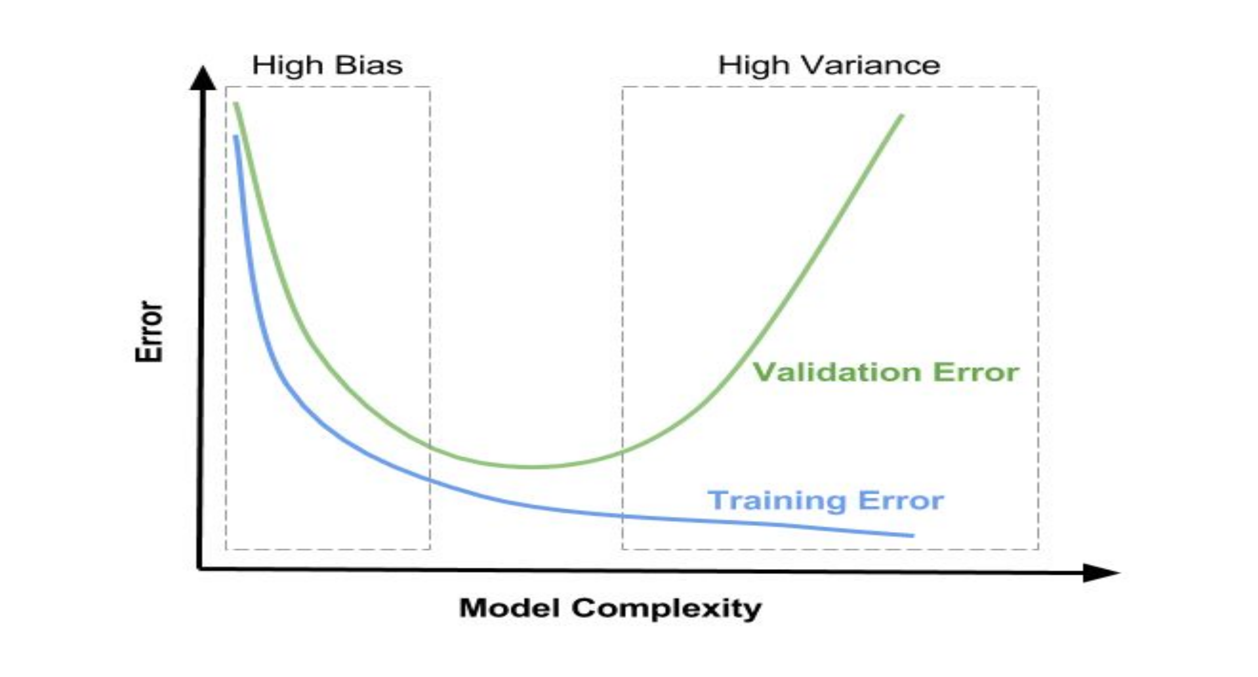
```
library(caret)
set.seed(1234)
trainIndex <- createDataPartition(df$targetVariable, p = .8,
                                  list = FALSE,
                                  times = 1)

train <- df[ trainIndex,]
test  <- df[-trainIndex,]
```

Bias-variance tradeoff



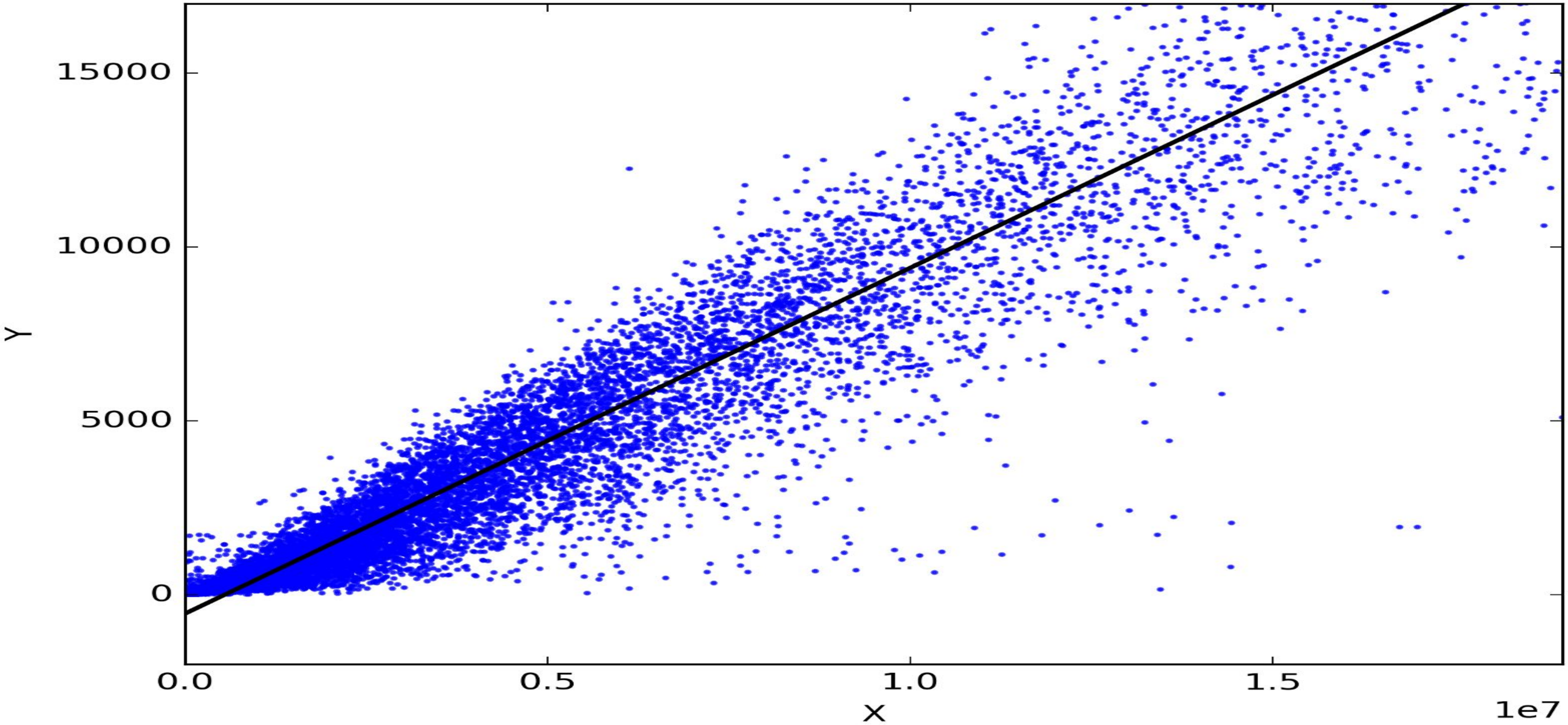
http://www.turingfinance.com/perils-optimization-in-investment-management/

Bias-variance tradeoff



https://www.learnopencv.com/bias-variance-tradeoff-in-machine-learning/

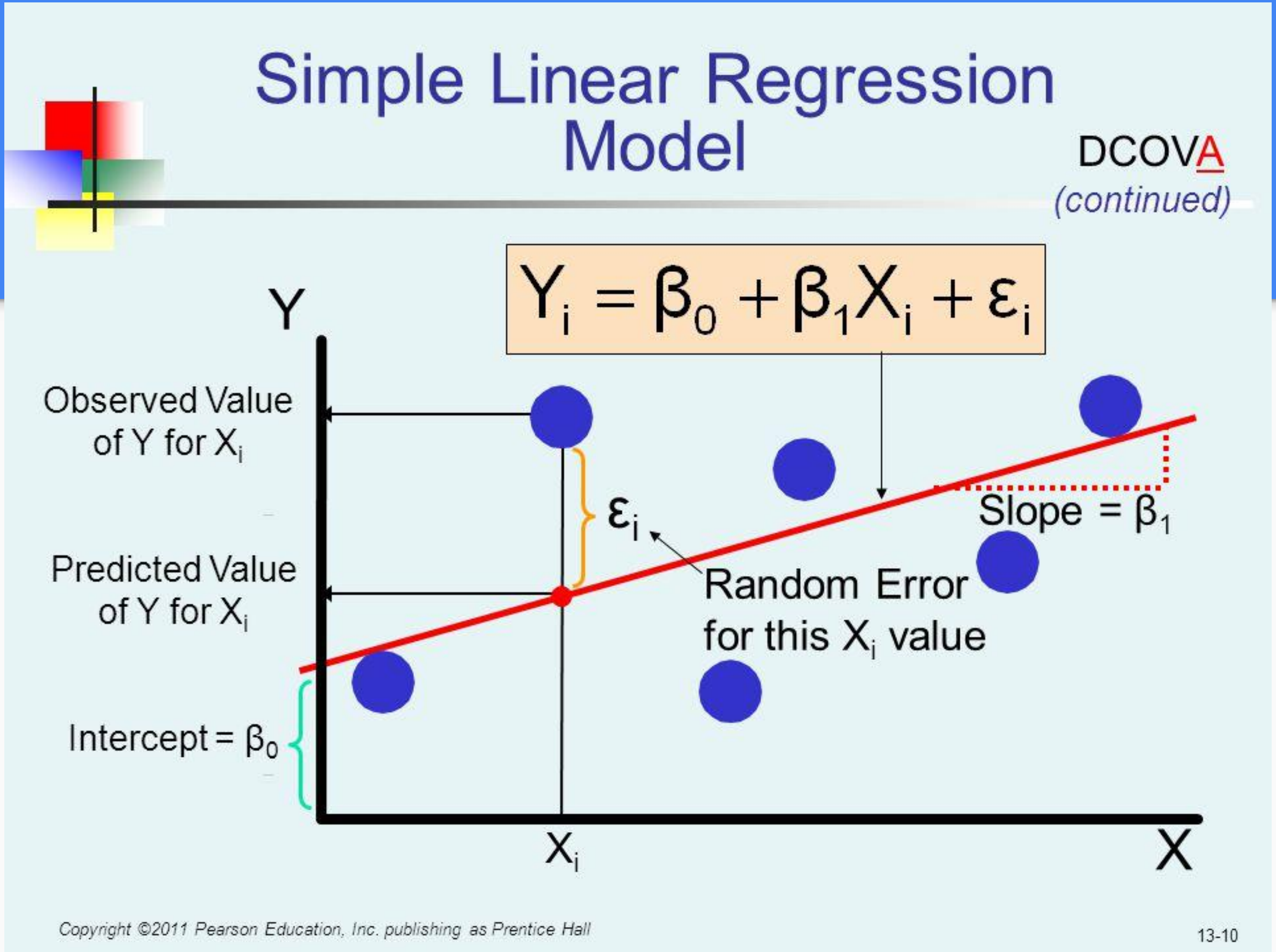
Regression



https://medium.com/@amarbudhiraja/ml-101-linear-regression-tutorial-1e40e29f1934

Regression background

- "***Regression***" is the act of predicting a continuous quantity (e.g. the price of something)
- An example is ***linear regression***, which fits an (N-1)-dimensional hyperplane to an MxN numeric matrix of observations such that distance between the hyperplane and all observations is minimized
  - A typically used cost function is ***mean-squared error*** (MSE) e.g. the average of the sum of (*y* - *y_pred*)^2 across all M observations
- Linear regression assumes normally distributed error terms, and is therefore a ***parametric*** model
  - Other assumptions include independence and even variance of errors, a true underlying linear relationship, and uncorrelated input variables



https://github.com/tirthajyoti/R-stats-machine-learning/blob/master/README.md

Linear regression code - fitting the model

```
lin_reg <- lm(fev ~ age, data = train)

summary(lin_reg)
```

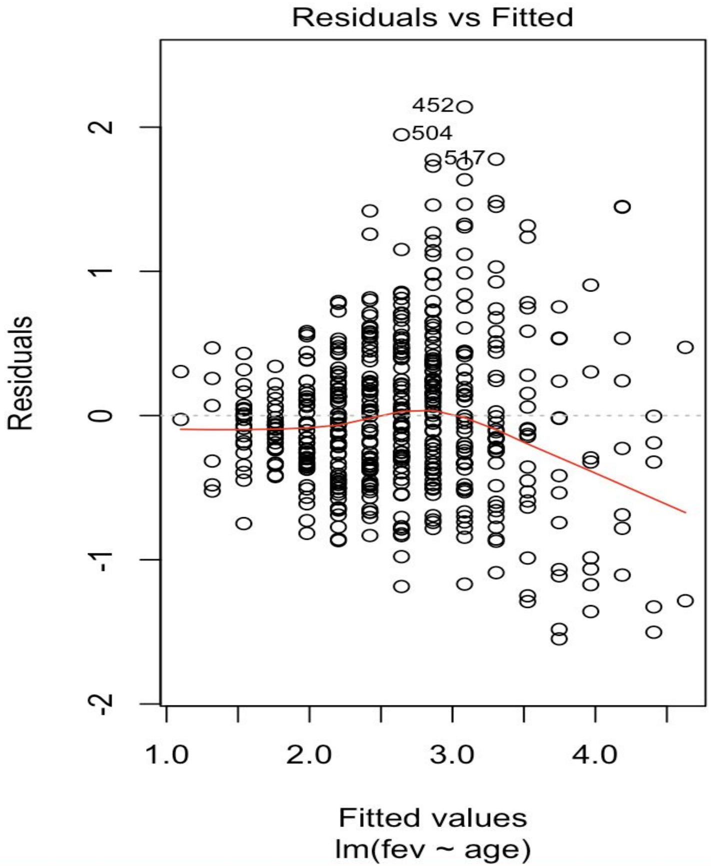
```
Call:
lm(formula = fev ~ age, data = train_reg)

Residuals:
     Min       1Q   Median       3Q      Max 
-1.54869 -0.34681 -0.05007  0.30434  2.13925 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 0.436999   0.088163   4.957 9.69e-07 ***
age         0.220646   0.008508  25.935  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.5672 on 524 degrees of freedom
Multiple R-squared:  0.5621,	Adjusted R-squared:  0.5613 
F-statistic: 672.6 on 1 and 524 DF,  p-value: < 2.2e-16
```

Linear regression code - evaluation

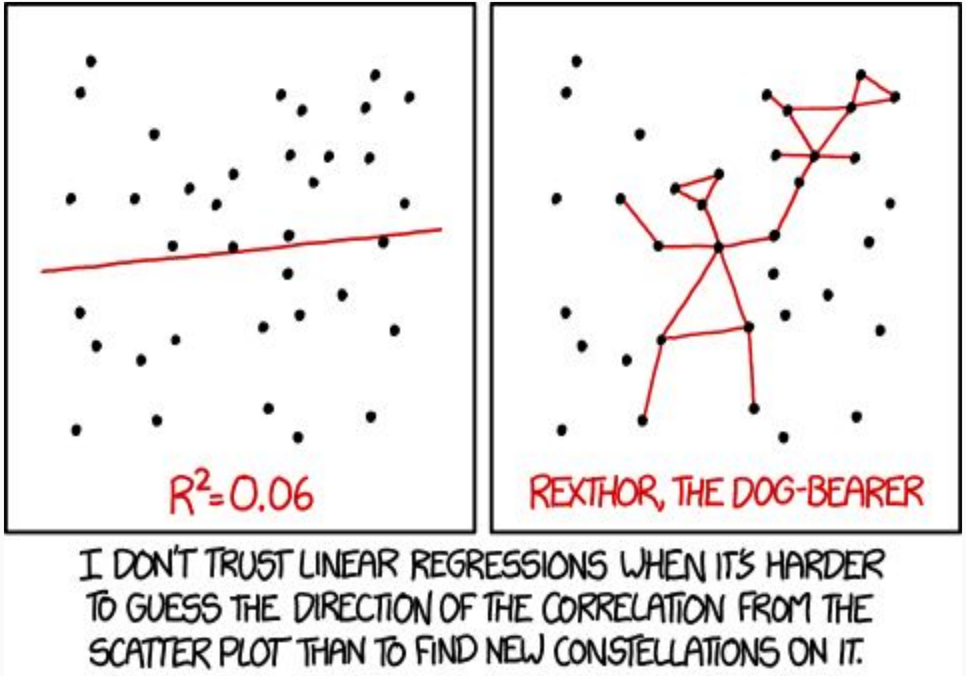
```
pred_lm <- predict(lin_reg, test)

mse_lm <- mean((test$fev - pred_lm)^2)
```

```
> mse_lm
[1] 0.324164
```

```
plot(lin_reg)
```

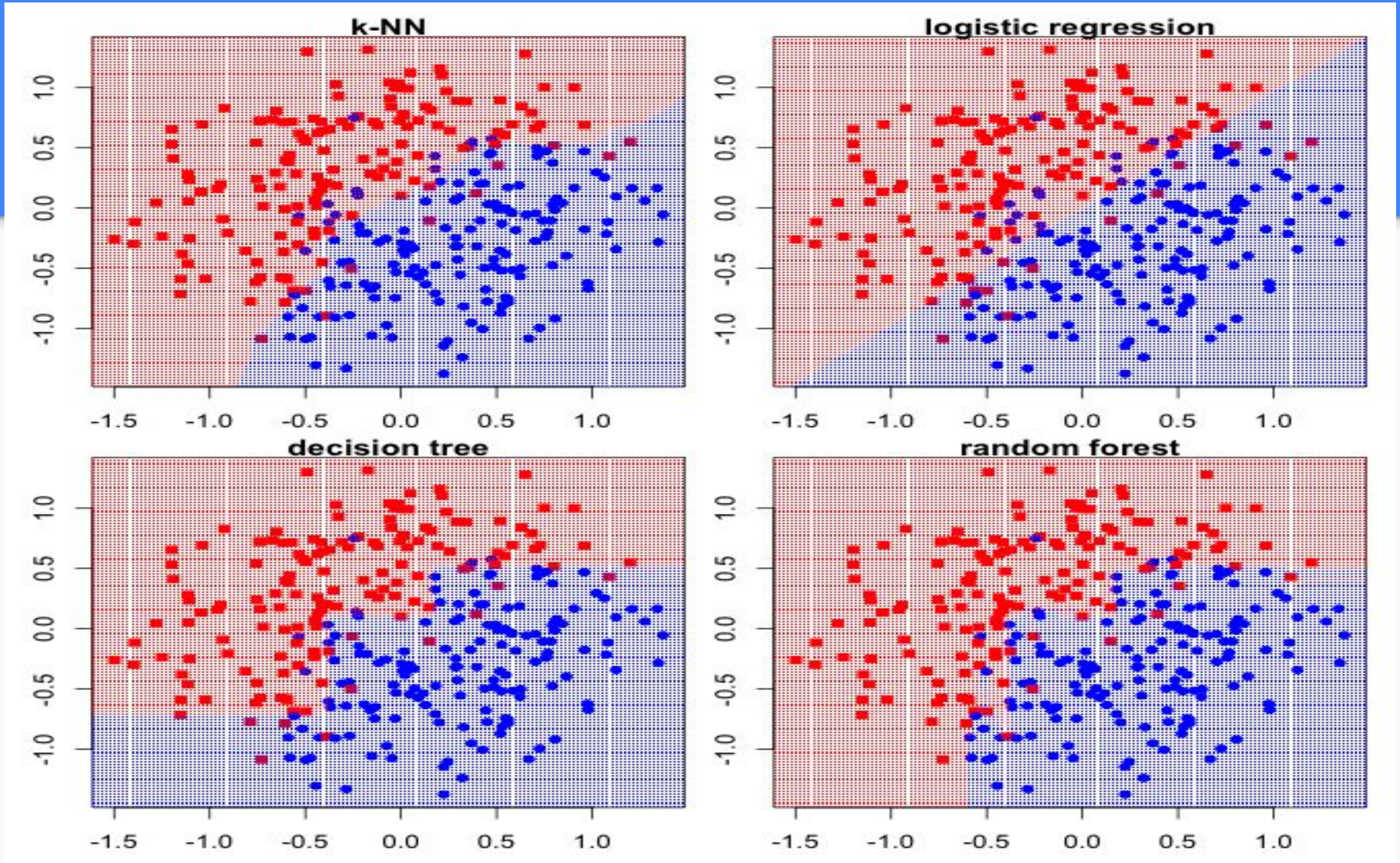


Relevant XKCD



https://www.explainxkcd.com/wiki/index.php/1725:_Linear_Regression

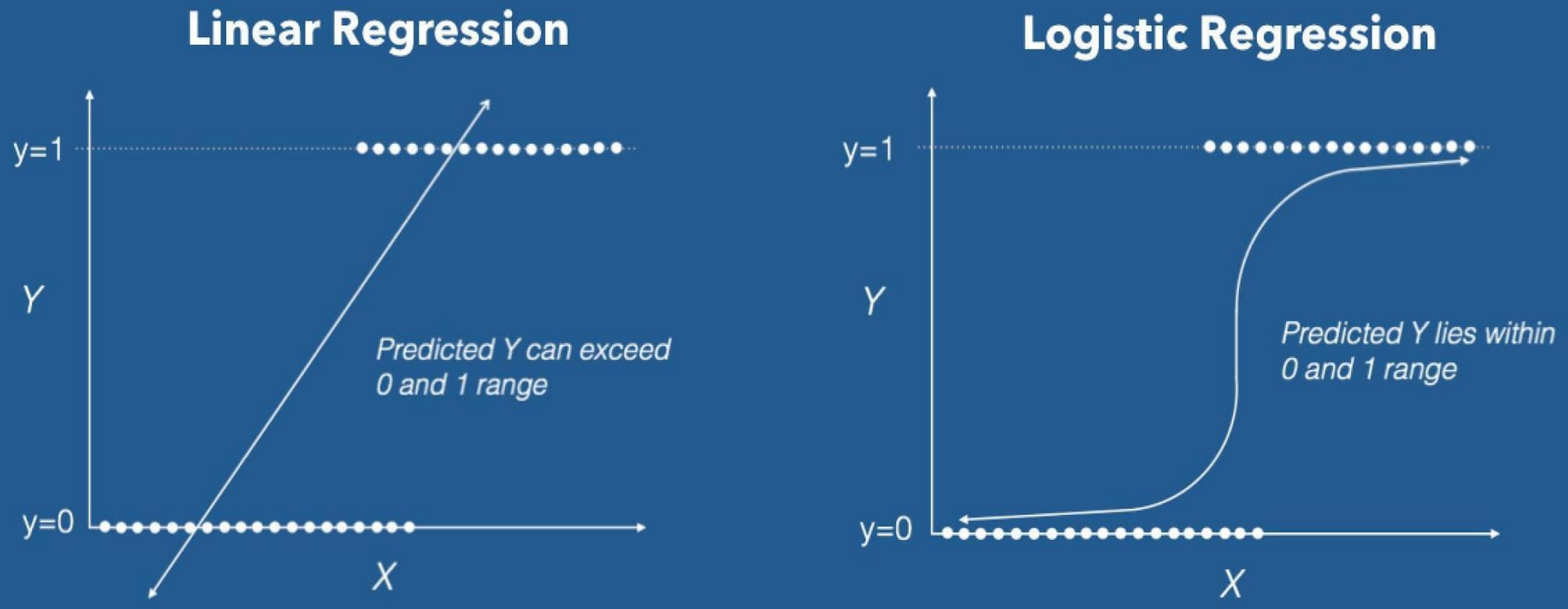
Classification



https://www.oreilly.com/library/view/data-analysis-with/9781788393720/7ef9ff0a-000c-44e1-ad6a-d352e0744c74.xhtml

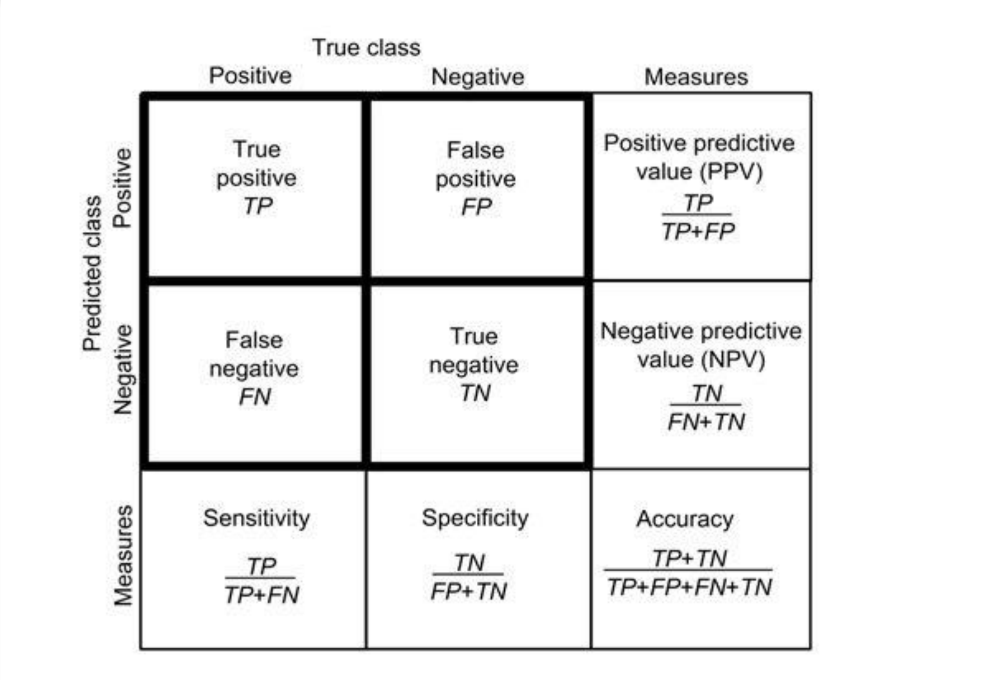
Classification background

- In ***classification***, our interest is a discrete class rather than a continuous quantity (e.g. a label indicating whether a sale was made or not), and our goal is to find a ***decision boundary*** that accurately separates classes
- A common classification model is ***logistic regression***, which outputs a probability of each observation belonging to a class
  - Like linear regression, it consists of a linear set of coefficients which are fed into a link function ensuring they output a real number between 0 and 1
- Classifiers are generally evaluated through metrics like ***accuracy***, ***precision***, and ***recall***
  - The right metric to optimize will depend on your modeling situation



https://medium.com/greyatom/logistic-regression-89e496433063

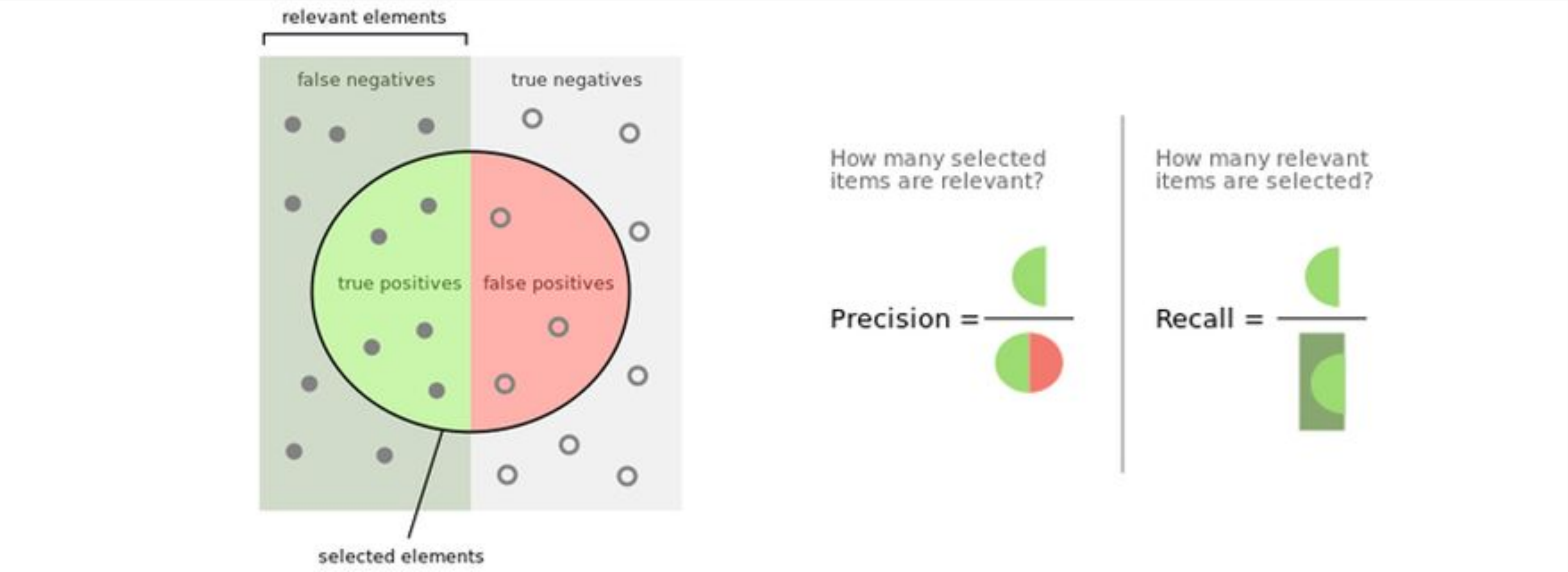
Classification evaluation - confusion matrix



		True class: Positive	True class: Negative	Measures
Predicted class	Positive	True positive TP	False positive FP	Positive predictive value (PPV) $\frac{TP}{TP+FP}$
Predicted class	Negative	False negative FN	True negative TN	Negative predictive value (NPV) $\frac{TN}{FN+TN}$
Measures		Sensitivity $\frac{TP}{TP+FN}$	Specificity $\frac{TN}{FP+TN}$	Accuracy $\frac{TP+TN}{TP+FP+FN+TN}$

https://www.researchgate.net/figure/Contingency-matrix-and-measures-calculated-based-on-it-2x2-contigency-table-for_fig4_230614354

Classification evaluation - precision/recall



https://www.digital-mr.com/blog/view/precision-and-recall-in-social-listening

Classification code - fitting the model

```
log_reg <- glm(smoke ~ fev +
age, data = train,
family=binomial(link="logit"))

summary(log_reg)
```

```
> exp(log_reg$coefficients)
 (Intercept)          fev          age
1893.0724028    1.3276088    0.5853436
```

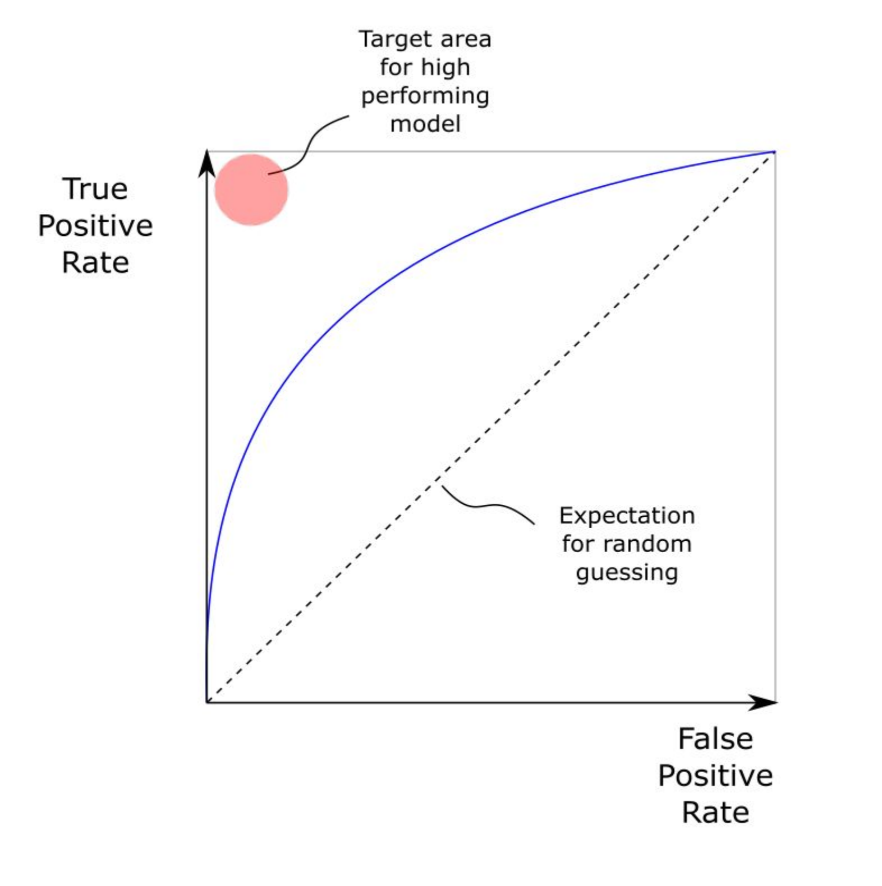
```
Call:
glm(formula = smoke ~ fev + age, family = binomial(link = "logit"),
    data = train_class)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.5754   0.1608   0.2533   0.3979   1.7491

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  7.54596    0.80404   9.385  < 2e-16 ***
fev          0.28338    0.23835   1.189    0.234
age         -0.53556    0.07505  -7.136 9.62e-13 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 338.93  on 523  degrees of freedom
Residual deviance: 249.04  on 521  degrees of freedom
AIC: 255.04

Number of Fisher Scoring iterations: 6
```

Classification code - evaluation

```
pred_glm <- predict(log_reg,
test_class, type = "response")

thresh <- .5

pred_glm <- ifelse(pred_glm > thresh,
1, 0)

confusionMatrix(factor(test$smoke),
factor(pred_glm), positive = "1")
```

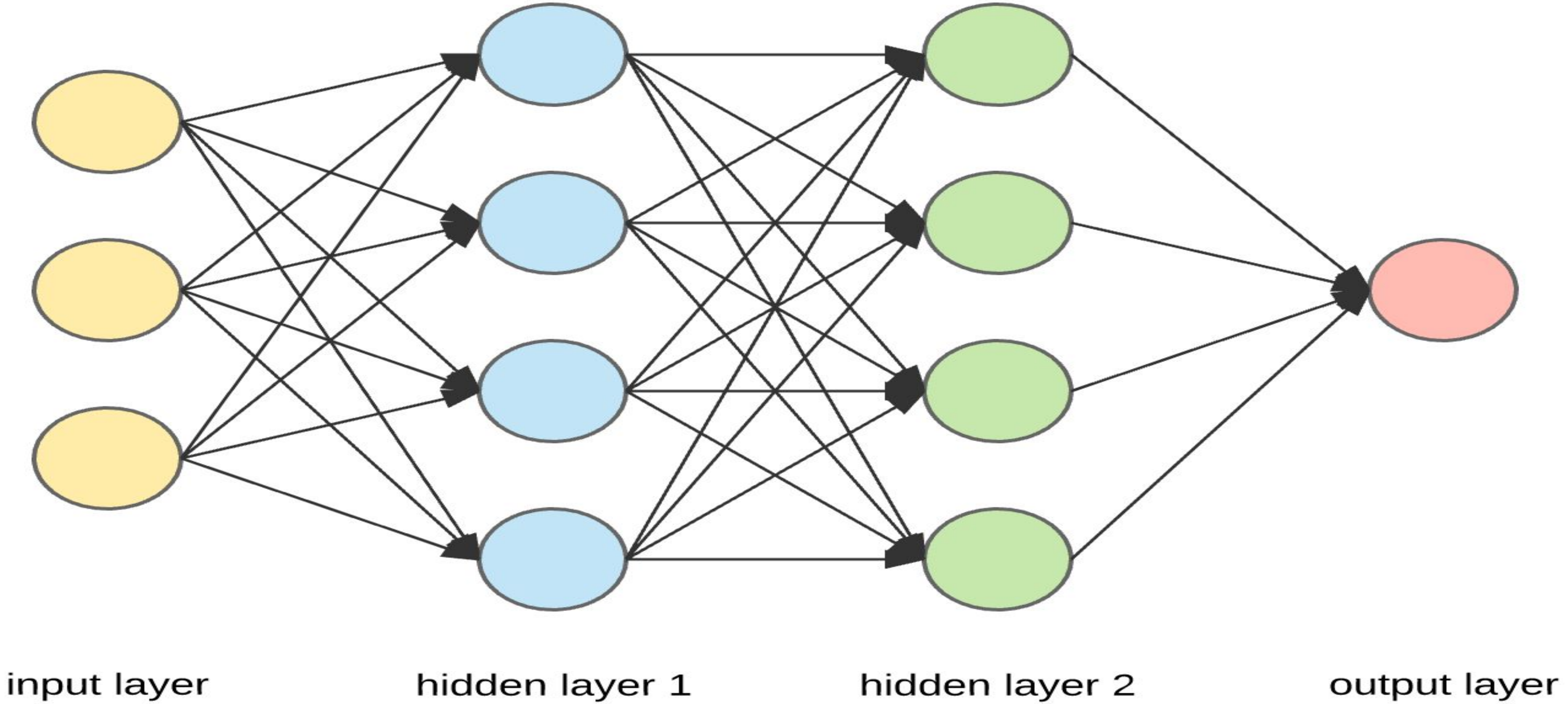
```
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0   4   9
         1   3 114

               Accuracy : 0.9077
                 95% CI : (0.8443, 0.9514)
    No Information Rate : 0.9462
    P-Value [Acc > NIR] : 0.9765

                  Kappa : 0.3548
 Mcnemar's Test P-Value : 0.1489

            Sensitivity : 0.9268
            Specificity : 0.5714
         Pos Pred Value : 0.9744
         Neg Pred Value : 0.3077
             Prevalence : 0.9462
         Detection Rate : 0.8769
   Detection Prevalence : 0.9000
      Balanced Accuracy : 0.7491
```

ROC curve



https://deparkes.co.uk/2018/02/16/the-roc-curve/

Advanced Methods

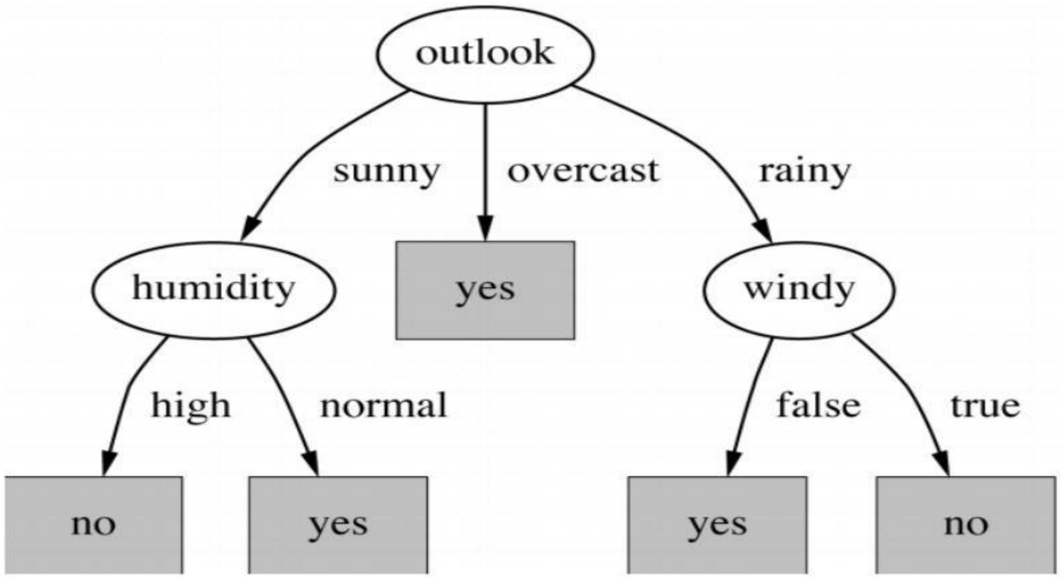


https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Decision tree background

- Some advanced methods can perform both regression and classification
- ***Decision trees*** attempt to split observations into distinct groupings based on feature values such that bin purity is maximized, ultimately creating a binary tree in the process
    - The predicted class will then be the maximally-represented class in the leaf nodes for classification, or an average of continuous values for a regression
- Decision trees are prone to overfitting, but the number of splits (e.g. tree depth) can be specified beforehand by the modeler
- ***Non-parametric***, meaning it makes no assumption of the underlying form of the data

Decision tree (playing tennis)



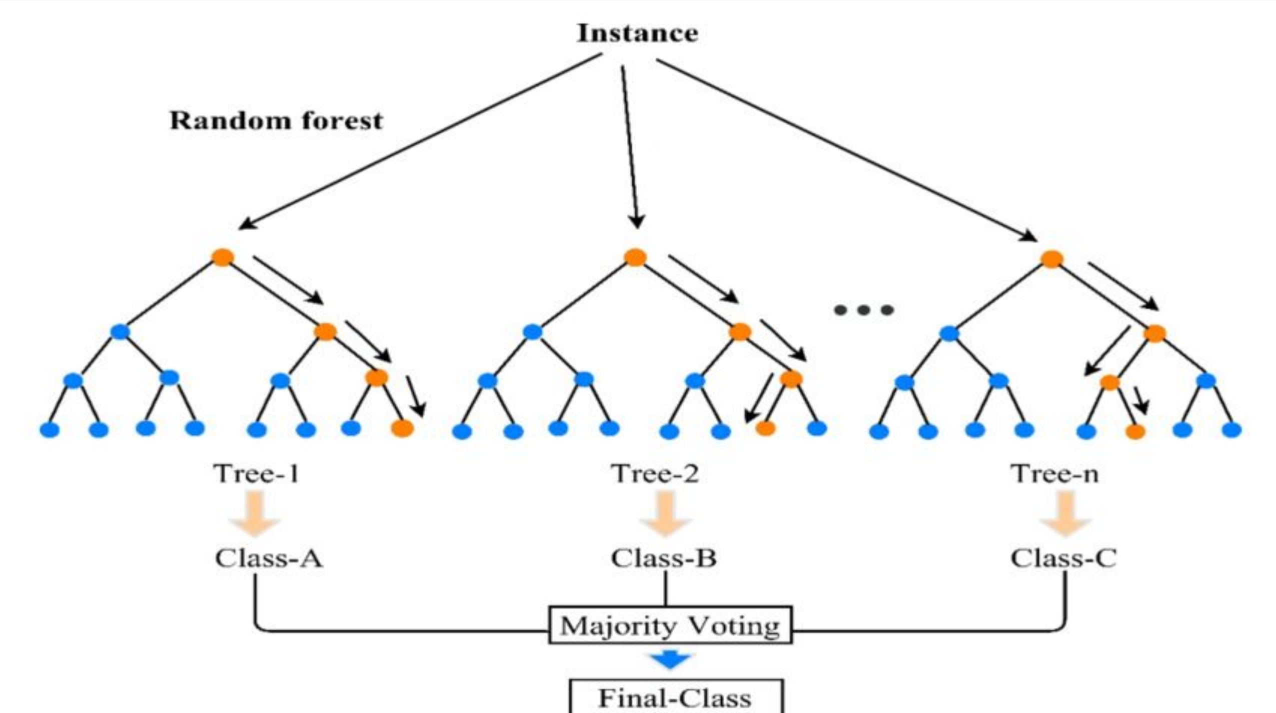
https://medium.com/deep-math-machine-learning-ai/chapter-4-decision-trees-algorithms-b93975f7a1f1

Random forest background

- In order to reduce variance and chance of overfitting, a ***random forest*** is an average of the results of many decision trees splitting on random subsets of features
- Random forests are typically a ***robust*** model working well in many situations (including high-dimensional data), but they may lack interpretability as a result
- Other forms of tree-based learning include ***boosting*** and ***bagging*** (beyond the scope of this presentation, but worth looking into for those interested!)

Random forest example

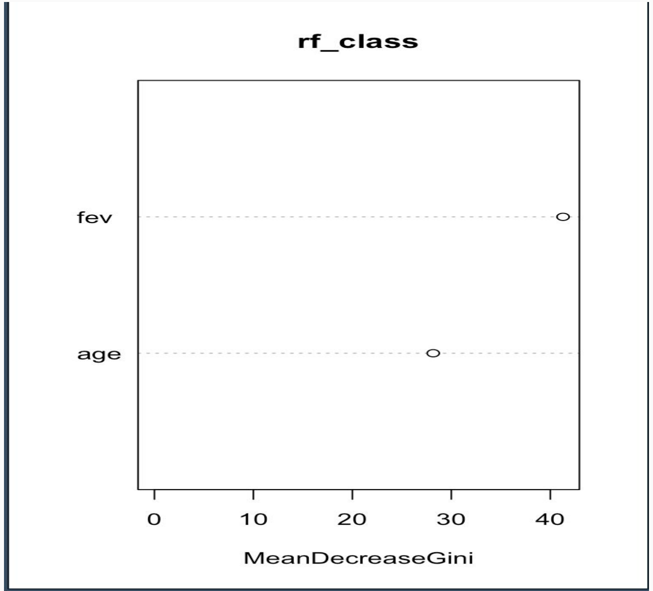
https://www.researchgate.net/figure/Random-Forests-Naive-Bayes-NB-NB-approaches-are-a-family-of-simple-probabilistic_fig2_326722598



Random forest code example

```
library(randomForest)

rf_reg <- randomForest(fev ~ age, data = train)

rf_class <- randomForest(factor(smoke) ~ age + fev, data = train)
```

```
varImpPlot(rf_class)
```



Additional types of models

- We have only covered a small subset of modeling approaches that exist!
- Regression
  - Regularized linear regression (LASSO, Ridge, Elastic Net), polynomial regression/splines
- Classification
  - Naive Bayes, linear/quadratic discriminant analysis
- Regression/classification
  - Support vector machines, K-nearest neighbors, XGBoost
- Other approaches
  - Survival analysis, time series, structural equation modeling

Other types of modeling

- Unsupervised learning - infer *y* from the data (no explicit optimization criteria)
    - E.g. *k*-means clustering (split data in *k* distinct groupings, useful for market segmentation)
    - Also encompasses dimensionality reduction techniques like principal component analysis
- Deep learning - send input through a nested series of “hidden” activation functions (or layers) before reaching the output
    - E.g. neural networks (require lots of data, often used for language or vision related tasks)
- Reinforcement learning - choose the best actions in a system where many are possible in order to optimize some reward function over time
    - E.g. q-learning (useful for robotics and training AI to play video games)

Additional resources

- Linear regression: http://r-statistics.co/Linear-Regression.html
- Logistic regression: http://uc-r.github.io/logistic_regression
- Advanced methods: https://lgatto.github.io/IntroMachineLearningWithR/an-introduction-to-machine-learning-with-r.html

Exercises

https://www.kaggle.com/ermoore/modeling-in-r-exercises-5-college-datafest-2019