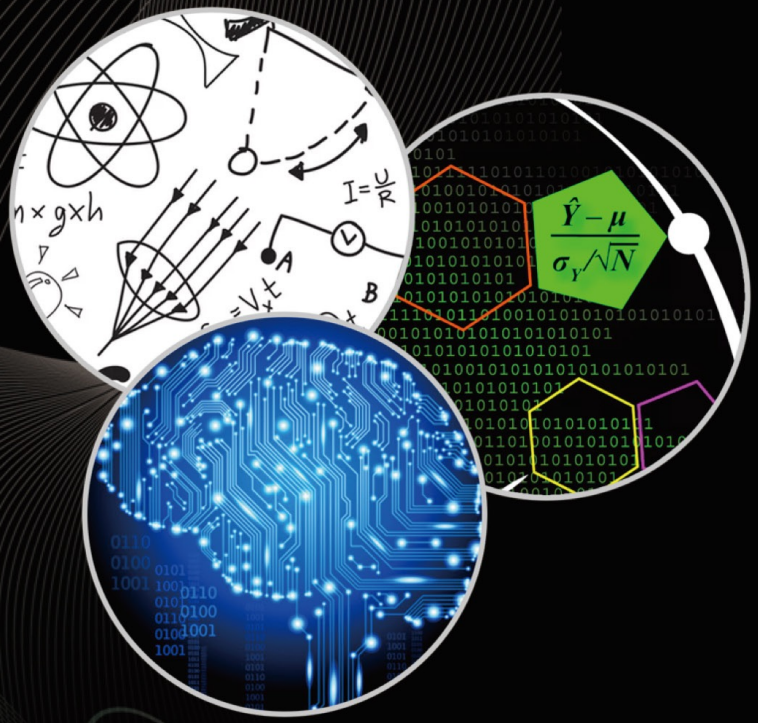


Tidymodels

Outline

- Introduction to Tidymodels
- Example: Prediction of diamonds price

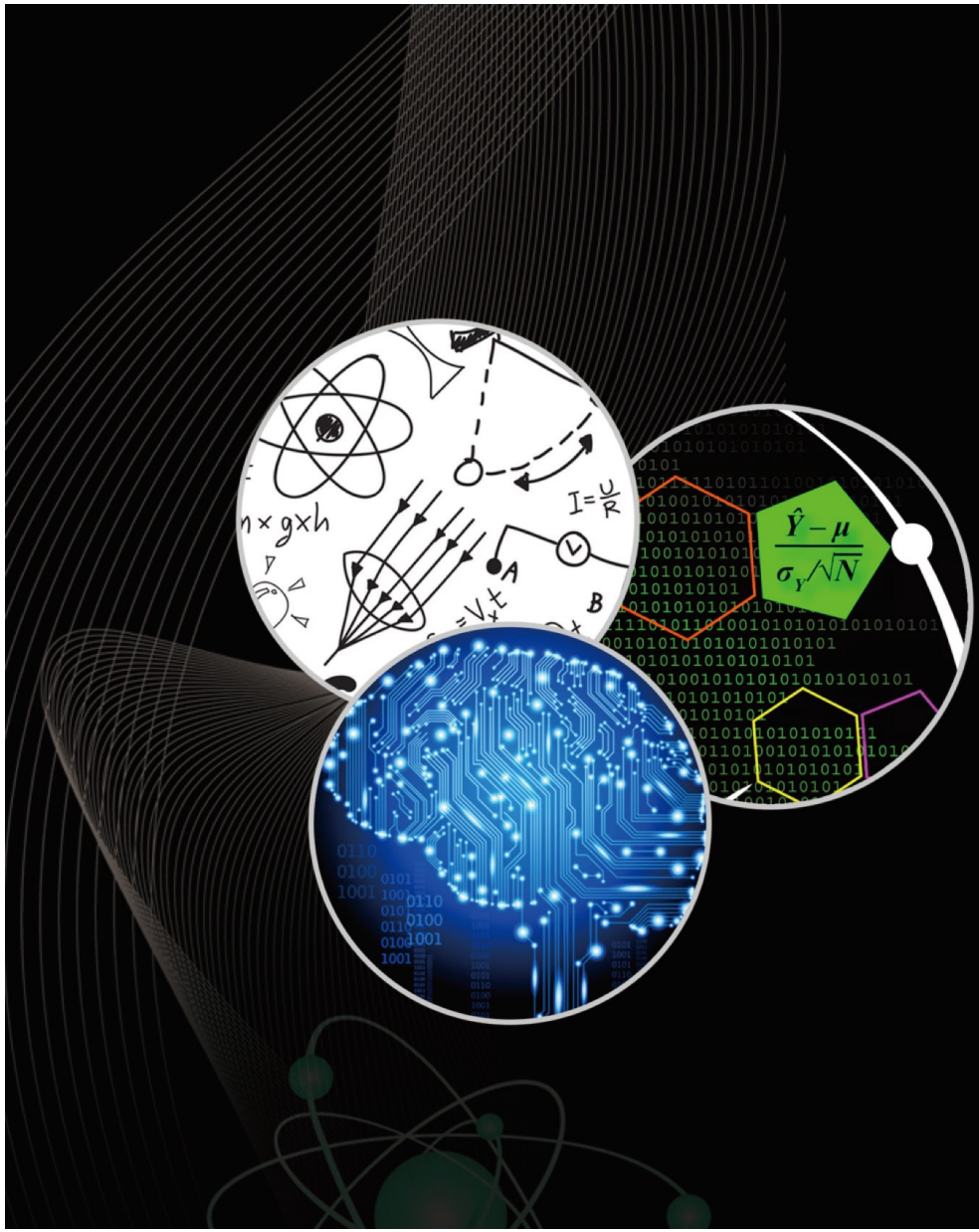
  - Dataset overview

  - Processes of predicting: core packages of Tidymodels

    (rsample/recipes/parsnip/broom/tune/dials/yardstick/workflows)

Introduction

- 什么是Tidymodels？
  - R语言机器学习建模的统一框架，整合了不同机器学习与统计建模的算法；由不同的包执行不同的建模步骤
- 机器学习的一般建模过程是什么？主要涉及Tidymodels的哪些包？
  - 数据准备（训练集/测试集划分）—— rsample
  - 数据预处理（特征工程）—— recipes
  - 模型的训练和调整 —— parsnip/tune/workflows/yardstick/dials
  - 模型验证 —— tune/workflow/parsnip

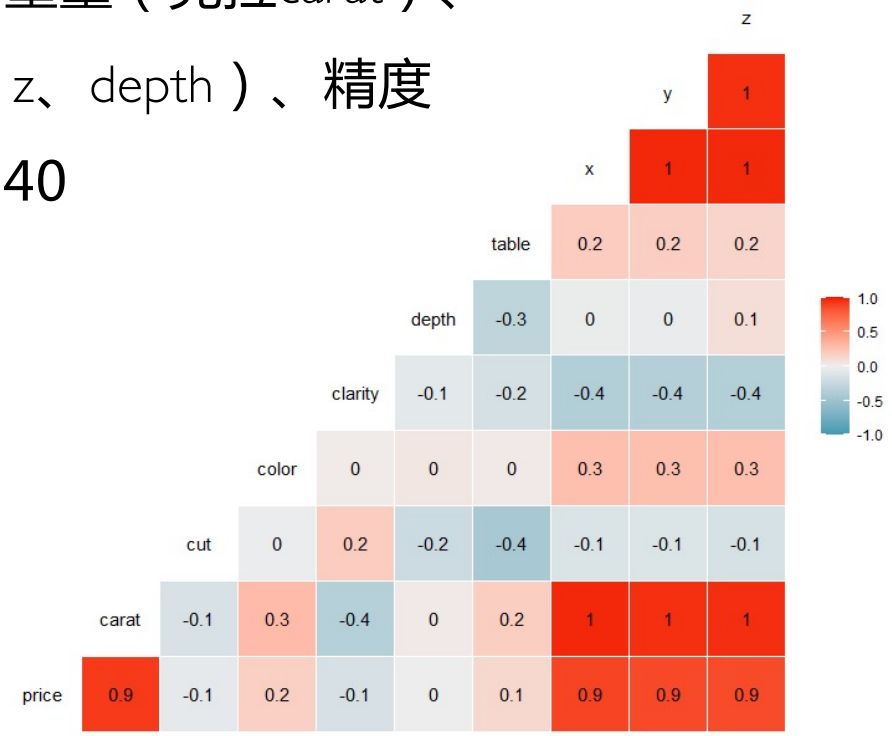


Example: Prediction of diamonds price

00 - Dataset Overview

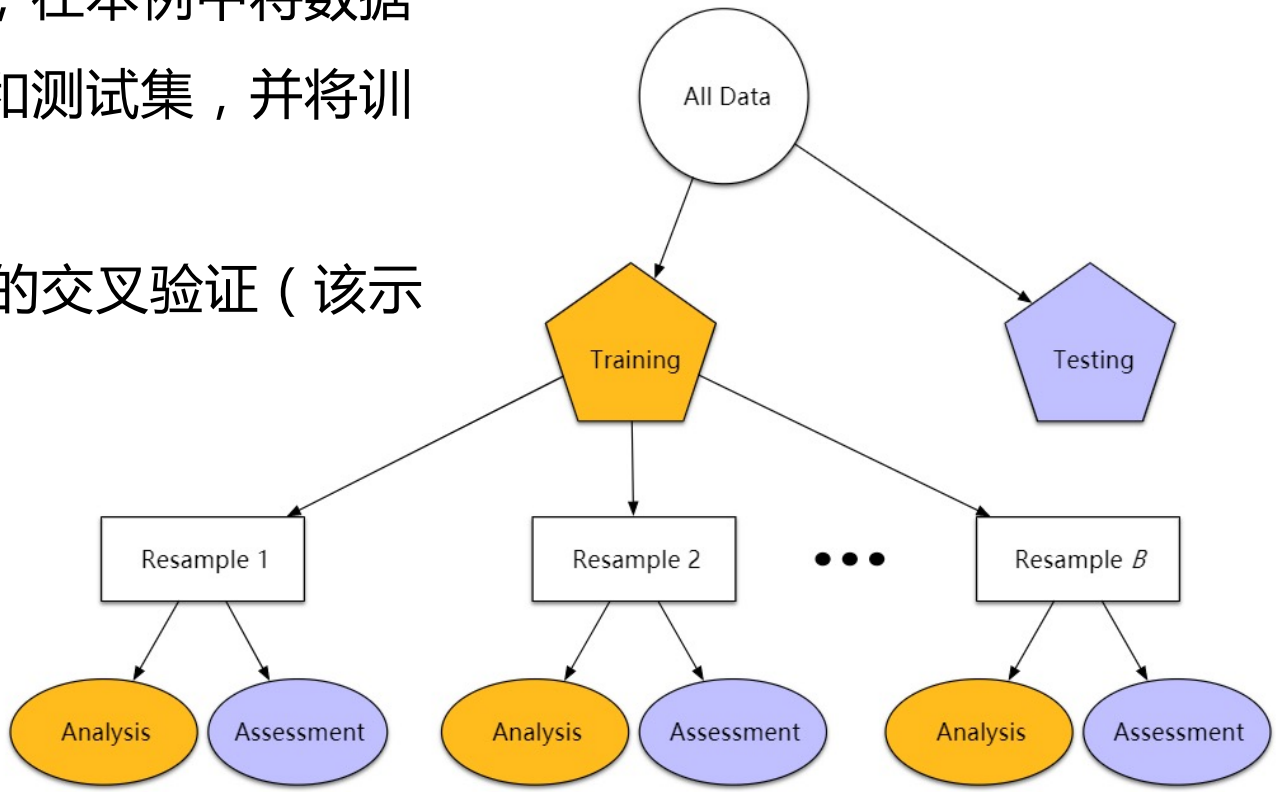
- Diamonds数据集已知信息：价格（price）、重量（克拉carat）、颜色（color）、切割（cut）、尺寸（x、y、z、depth）、精度（clarity）、台宽比（table），数据量为53940

- ggcorr——变量相关性图形
  - 图中可知，price与carat之间存在较高相关性



01 - rsample

- 用以创建数据集的变体，在本例中将数据集随机地划分为训练集和测试集，并将训练集分折
- 分折时，通常采用10折的交叉验证（该示例为3折）



initial_split()　　划分

```
dia_split<-initial_split(diamonds,prop=.5)
dia_train<-training(dia_split)
dia_test<-testing(dia_split)
```

vfold_cv()　　对训练集分折

```
dia_vfold<-vfold_cv(dia_train,v=3,repeats=1)
```

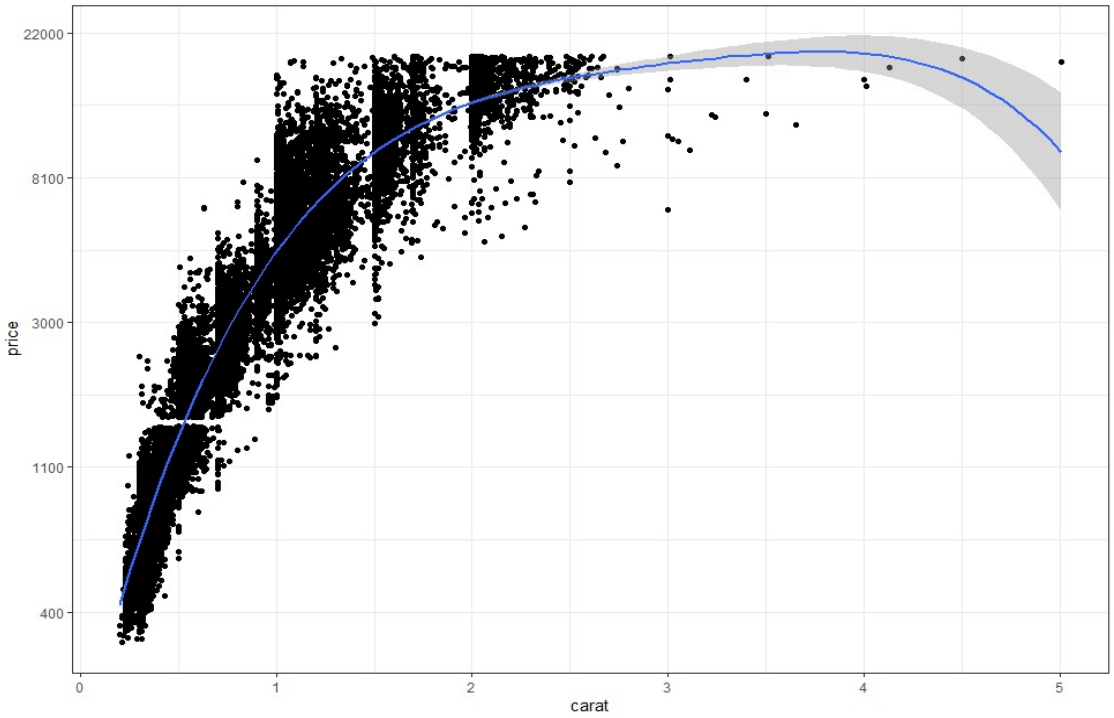
```
> str(dia_train)
tibble[,10] [26,970 x 10] (S3: tbl_df/tbl/data.frame)
 $ carat   : num [1:26970] 0.21 0.29 0.31 0.24 0.22 0.2 0.3 0.3 0.23 0.23 ...
 $ cut     : Ord.factor w/ 5 levels "Fair"<"Good"<..: 4 4 2 3 4 4 2 3 3 3 ...
 $ color   : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 2 6 7 7 3 2 7 7 2 5 ...
 $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 3 4 2 6 3 2 3 3 4 5 ...
 $ depth   : num [1:26970] 59.8 62.4 63.3 62.8 60.4 60.2 63.8 62.7 63.8 61 ...
 $ table   : num [1:26970] 61 58 58 57 61 62 56 59 55 57 ...
 $ price   : int [1:26970] 326 334 335 336 342 345 351 351 352 353 ...
 $ x       : num [1:26970] 3.89 4.2 4.34 3.94 3.88 3.79 4.23 4.21 3.85 3.94 ...
 $ y       : num [1:26970] 3.84 4.23 4.35 3.96 3.84 3.75 4.26 4.27 3.92 3.96 ...
 $ z       : num [1:26970] 2.31 2.63 2.75 2.48 2.33 2.27 2.71 2.66 2.48 2.41 ...
```

```
> dia_vfold[1]
# A tibble: 3 x 1
  splits
  <list>
1 <split [17980/8990]>
2 <split [17980/8990]>
3 <split [17980/8990]>
> dia_vfold[[1]]
[[1]]
<Analysis/Assess/Total>
<17980/8990/26970>

[[2]]
<Analysis/Assess/Total>
<17980/8990/26970>

[[3]]
<Analysis/Assess/Total>
<17980/8990/26970>

> dia_vfold[[2]]
[1] "Fold1" "Fold2" "Fold3"
```

02 – recipe

- 使用不同的“步骤”step_*()函数来对（用于建模的）数据集进行预处理；prep() 进行该处理， juice()将处理好的整洁数据框“榨汁”提取出来
- 为什么需要预处理？

  - 如图，价格price和克拉carat之间可能存在着非线性关系，需要引入高阶项来拟合两者之间的关系



```
dia_rec<- recipe(price~.,data=dia_train) %>%
step_log(all_outcomes())%>%
step_normalize(all_predictors(),-all_nominal())%>%
step_dummy(all_nominal())%>%
step_poly(carat,degree=4)
```

对数变换

中心化和标准化

哑编码

四次正交多项处理

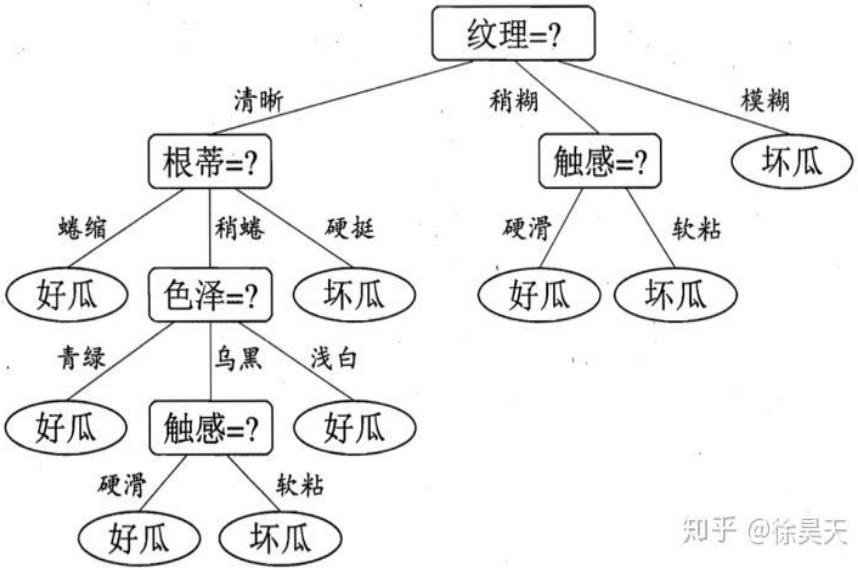
- 预处理后的数据

```
> names(dia_juiced)
 [1] "depth"        "table"        "x"            "y"            "z"            "price"
 [7] "cut_1"        "cut_2"        "cut_3"        "cut_4"        "color_1"      "color_2"
[13] "color_3"      "color_4"      "color_5"      "color_6"      "clarity_1"    "clarity_2"
[19] "clarity_3"    "clarity_4"    "clarity_5"    "clarity_6"    "clarity_7"    "carat_poly_1"
[25] "carat_poly_2" "carat_poly_3" "carat_poly_4"
```

03 – parsnip

决策树通过将数据划分为具有相似值的子集来构建出一个完整的树，通过一系列非叶节点上特征属性的测试，产生多个分支，由此生成的每个叶子节点就是表达输出结果的连续或离散的数据。

随机森林是以决策树为基础的一种更高级的算法。像决策树一样，随机森林既可以用于回归，也可以用于分类。这个森林是由很多互不相关的决策树组成，通过多个决策树结果投票来决定最后的结果，因此表现一般要优于单一的决策树。



随机森林模型的具体工作原理如下：

- 从数据集（表）中随机选择k个特征（列），共m个特征（其中k小于等于m），m值由mtry确定。然后根据这k个特征建立决策树。
- 重复n次，这k个特性经过不同随机组合建立起来n棵决策树（或者是数据的不同随机样本，称为自助法样本），n值由ntree确定。
- 对每个决策树都传递随机变量来预测结果。存储所有预测的结果（目标），你就可以从n棵决策树中得到n种结果。
- 计算每个预测目标的得票数，再选择模式（最常见的目标变量）。换句话说，将得到高票数的预测目标作为随机森林算法的最终预测。

Parsnip(定义并拟合模型)

- Parsnip包的目标是为模型提供一个整洁、统一的接口，可以用来操作一系列模型和算法，而不陷入每个底层包的语法细节中。对同一种模型，不同的R包会采用不同的语法、参数和数据结构（参见下面的例子，随机森林在ranger包、randomForest包、sparklyr包中的实现方式相当不同）。

模型中主要有以下重要参数:

formula指定模型的公式形式，类似于y~x1+x2+x3...;

data指定分析的数据集；

ntree指定随机森林所包含的决策树数目；

mtry指定节点中用于二叉树的变量个数；

importance用于指定是否计算各个变量在模型中的重要性；

```
# randomForest包
rf_1 <- randomForest(
  y ~ .,
  data = .,
  mtry = 10,
  ntree = 2000,
  importance = TRUE
)
```

```
# ranger包
rf_2 <- ranger(
  y ~ .,
  data = dat,
  mtry = 10,
  num.trees = 2000,
  importance = "impurity"
)
```

```
# sparklyr包
rf_3 <- ml_random_forest(
  dat,
  intercept = FALSE,
  response = "y",
  features =
names(dat)[names(dat) != "y"],
  col.sample.rate = 10,
  num.trees = 2000
)
```

- 通过parsnip包，基于不同底层包的随机森林实现都可如此完成：

```
rand_forest(mtry=3,trees=500,min_n=5) %>%
    set_engine("ranger", importance = "impurity") %>%
    set_mode("regression") %>%
    fit(price~ ., dia_juiced)
```

- parsnip的优点：
  - 模型定义和评价分离
  - 模型指定与执行分离
  - 统一了不同包中的变量名称（如n.trees, ntrees, trees），仅需记住一个变量
  - 无论具体使用哪个底层包，我们都可以以相同的方式进行拟合
- 若希望改变调用的底层包，仅需通过set_engine() 进行

parsnip目前可以支持的模型类型如下表所示

函数	模型/算法	模式	底层包（引擎）
boost_tree()	Boost树	classification, regression	xgboost、c5.0、spark
decision_tree()	决策树	classification, regression	rpart,C5.0,spark
linear_reg()	线性回归	regression	lm,glmnet,stan,spark,keras
logistic_reg()	逻辑回归	classification	glm,glmnet,stan,spark,keras
mars()	多元适应性回归样条	classification, regression	earth
mlp()	单层神经网络	classification, regression	nnet,keras
multinom_reg()	多项式回归	classification	glmnet,nnet,stan,keras
nearest_neighbor()	L近邻聚类	classification, regression	kknn

04 – broom

许多模型都提供了用于输出模型拟合结果的函数，如summary()或coef()，然而，不同包的输出格式基本各不相同。

broom包可以使建模结果变量更加整洁，使许多各不相同的模型输出转化成整洁的tibble形式，其主要有三个常用函数：

- tidy():查看模型截距、估计等
- glance():查看模型总体情况
- augment():查看构成模型的每个样本的情况

- tidy:查看模型截距、估计等

它给出模型成分（各个预测变量的系数）的信息，每个成分给出一行。

```
broom::tidy(lm_fit)%>%
    arrange(desc(abs(statistic)))
```

```
> broom::tidy(lm_fit)%>%
+   arrange(desc(abs(statistic)))
# A tibble: 27 x 5
   term          estimate std.error statistic   p.value
   <chr>            <dbl>     <dbl>     <dbl>     <dbl>
 1 (Intercept)     7.72     0.00155    4980.   0.
 2 clarity_1       0.903    0.00482     188.   0.
 3 carat_poly_2  -48.7      0.269      -181.   0.
 4 color_1        -0.441    0.00274    -161.   0.
 5 carat_poly_1  169.       1.25        135.   0.
 6 carat_poly_3   15.2      0.147       103.   0.
 7 clarity_2      -0.241    0.00447     -54.1  0.
 8 carat_poly_4   -7.00     0.134       -52.3  0.
 9 color_2        -0.0877   0.00250     -35.1  3.72e-264
10 clarity_3       0.133    0.00382      34.7  1.05e-258
# ... with 17 more rows
```

- glance:查看模型总体情况

glance()给出的是整个模型的拟合优度及相关统计量，每个模型输出一行：

```
glance(lm_fit$fit)
```

```
> glance(lm_fit$fit)
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value    df logLik     AIC     BIC deviance
      <dbl>         <dbl> <dbl>     <dbl>   <dbl> <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
1     0.985         0.985 0.126    66029.       0    26 17602. -35148. -34919.     428.
# ... with 2 more variables: df.residual <int>, nobs <int>
```

- augment:查看构成模型的每个样本的情况

augment()将模型拟合结果（如拟合值/预测值，残差等）添加到进行训练的数据集上

```
lm_pred<-augment(lm_fit$fit,data=dia_juiced) %>%
+     rowid_to_column()
```

```
> lm_pred<-augment(lm_fit$fit,data=dia_juiced) %>%
+     rowid_to_column()
> lm_pred
# A tibble: 26,970 x 34
   rowid  depth  table     x     y     z price  cut_1  cut_2     cut_3  cut_4 color_1   color_2
   <int>  <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>     <dbl>  <dbl>   <dbl>     <dbl>
 1     1 -3.34   3.36 -1.50 -1.50 -1.72  5.79 -0.316 -0.267  6.32e- 1 -0.478  -0.378  9.69e-17
 2     2  0.449  0.241 -1.37 -1.35 -1.27  5.81  0.316 -0.267 -6.32e- 1 -0.478   0.378  0.
 3     3  0.724 -0.204 -1.60 -1.60 -1.48  5.82  0     -0.535 -4.10e-16  0.717   0.567  5.46e- 1
 4     4  0.380 -0.204 -1.59 -1.58 -1.50  5.82  0     -0.535 -4.10e-16  0.717   0.378  0.
 5     5 -1.62   1.58 -1.55 -1.52 -1.61  5.82  0     -0.535 -4.10e-16  0.717   0.189 -3.27e- 1
 6     6  0.724 -0.649 -1.61 -1.65 -1.51  5.83  0.632  0.535  3.16e- 1  0.120   0.567  5.46e- 1
 7     7  1.41  -0.649 -1.34 -1.33 -1.16  5.86 -0.316 -0.267  6.32e- 1 -0.478   0.567  5.46e- 1
 8     8  0.655  0.686 -1.36 -1.32 -1.23  5.86  0     -0.535 -4.10e-16  0.717   0.567  5.46e- 1
 9     9  1.41  -1.09  -1.68 -1.63 -1.48  5.86  0     -0.535 -4.10e-16  0.717  -0.378  9.69e-17
10    10 -1.62   2.02 -1.20 -1.17 -1.29  5.87  0     -0.535 -4.10e-16  0.717   0.567  5.46e- 1
# ... with 26,960 more rows, and 21 more variables: color_3 <dbl>, color_4 <dbl>,
#   color_5 <dbl>, color_6 <dbl>, clarity_1 <dbl>, clarity_2 <dbl>, clarity_3 <dbl>,
#   clarity_4 <dbl>, clarity_5 <dbl>, clarity_6 <dbl>, clarity_7 <dbl>, carat_poly_1 <dbl>,
#   carat_poly_2 <dbl>, carat_poly_3 <dbl>, carat_poly_4 <dbl>, .fitted <dbl>, .resid <dbl>,
#   .hat <dbl>, .sigma <dbl>, .cooksd <dbl>, .std.resid <dbl>
```

05 – yardstick

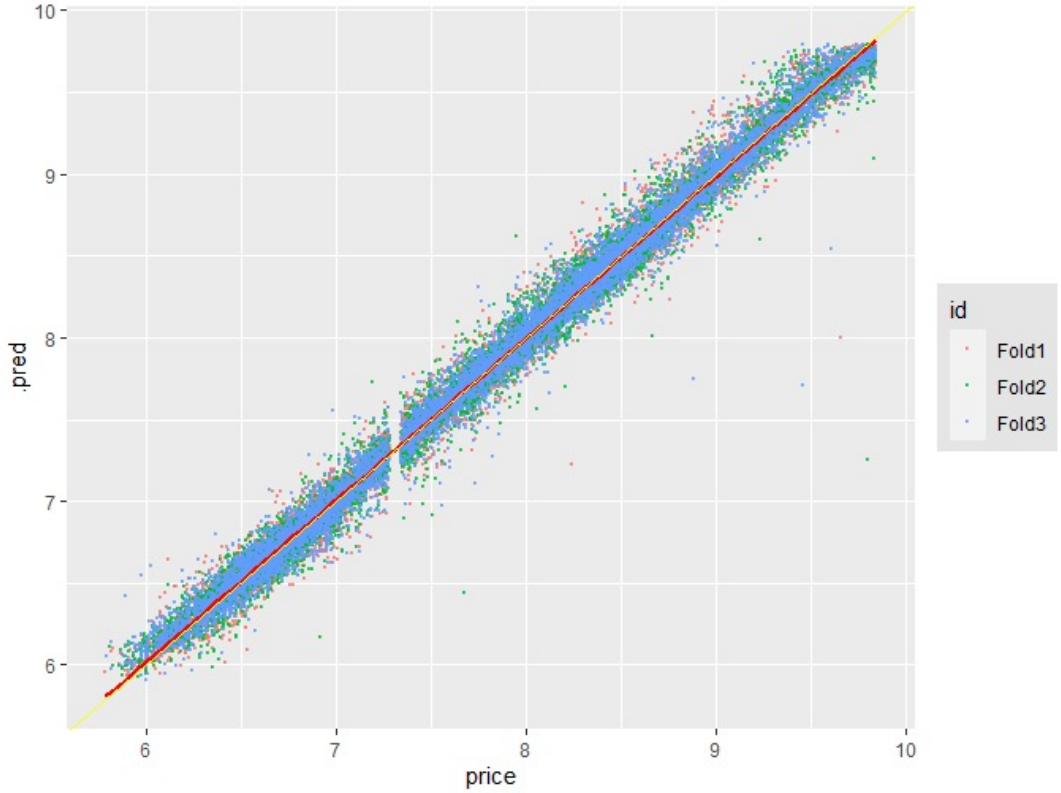
- 提供模型性能指标——包括分类指标、类别概率指标和回归指标
- 通用的性能估计函数：metrics()
  - metrics()中主要参数：
    -truth: 对应data中包含真实结果的列（数值或因子）
    -estimate: 对应data中包含预测结果的列
  - 根据truth参数的情况，metrics的输出结果不同，基本情形是：
    -truth为数值时，输出指标为rmse()、rsq()和mae()
    -truth为因子时，输出的指标为accuracy()和Kappa统计量kap()

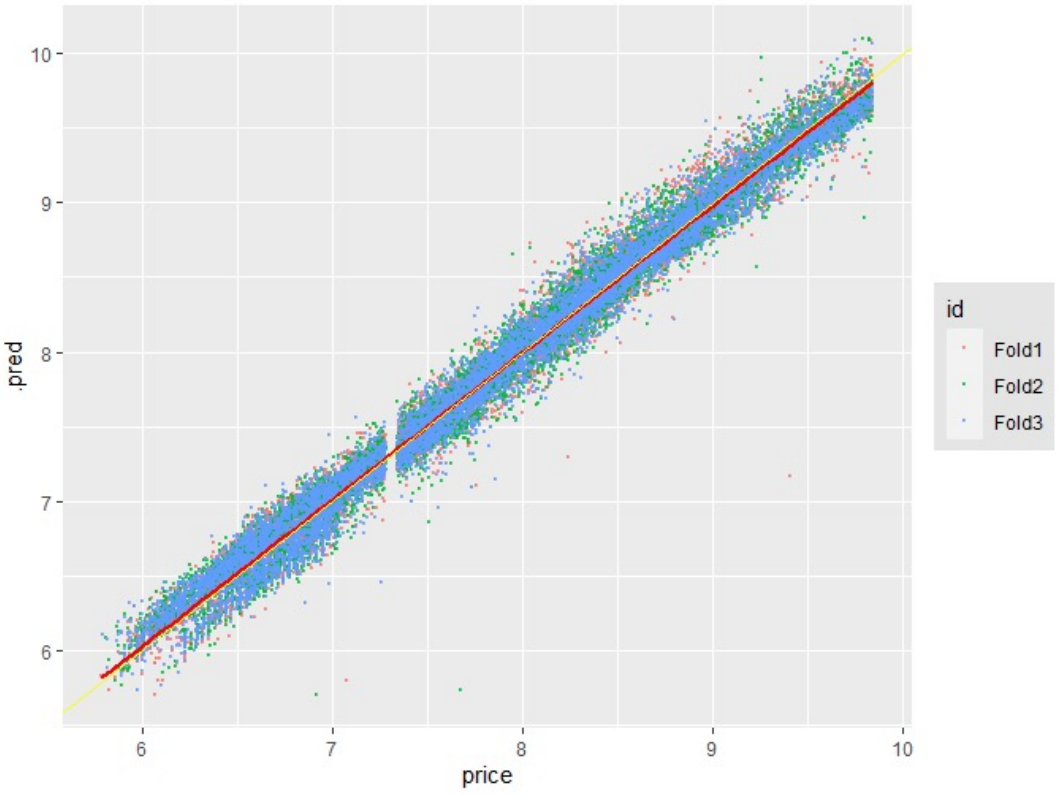
```
#数据框lm_pred与rf_pred包括钻石价格的真实值和预测值
metrics(lm_pred,truth=price,estimate=.pred) ##左边结果
metrics(rf_pred,truth=price,estimate=.pred) ##右边结果
```

```
# A tibble: 9 x 4
  id    .metric .estimator .estimate
  <chr> <chr>   <chr>          <dbl>
1 Fold1 rmse    standard       0.127
2 Fold2 rmse    standard       0.127
3 Fold3 rmse    standard       0.126
4 Fold1 rsq     standard       0.984
5 Fold2 rsq     standard       0.984
6 Fold3 rsq     standard       0.985
7 Fold1 mae     standard       0.0979
8 Fold2 mae     standard       0.0982
9 Fold3 mae     standard       0.0987
```

```
# A tibble: 9 x 4
  id    .metric .estimator .estimate
  <chr> <chr>   <chr>          <dbl>
1 Fold1 rmse    standard       0.0946
2 Fold2 rmse    standard       0.0963
3 Fold3 rmse    standard       0.0955
4 Fold1 rsq     standard       0.991
5 Fold2 rsq     standard       0.991
6 Fold3 rsq     standard       0.991
7 Fold1 mae     standard       0.0675
8 Fold2 mae     standard       0.0675
9 Fold3 mae     standard       0.0679
```

Lm与RF模型结果可视化





- 可见两者的预测结果与实际结果较为符合。

06 – tune/dials

- 一些统计和机器学习模型包含调节参数（也称为超参数），无法由模型直接估计。例如K-最近邻模型中的邻居数。为了确定这些参数恰当的取值，常采用一些间接的方法，如格点搜索、贝叶斯搜索等

- tune在调参中执行tidymodels的超参数调整（例如，网格搜索）
- dials包含用于创建和管理调节参数的工具

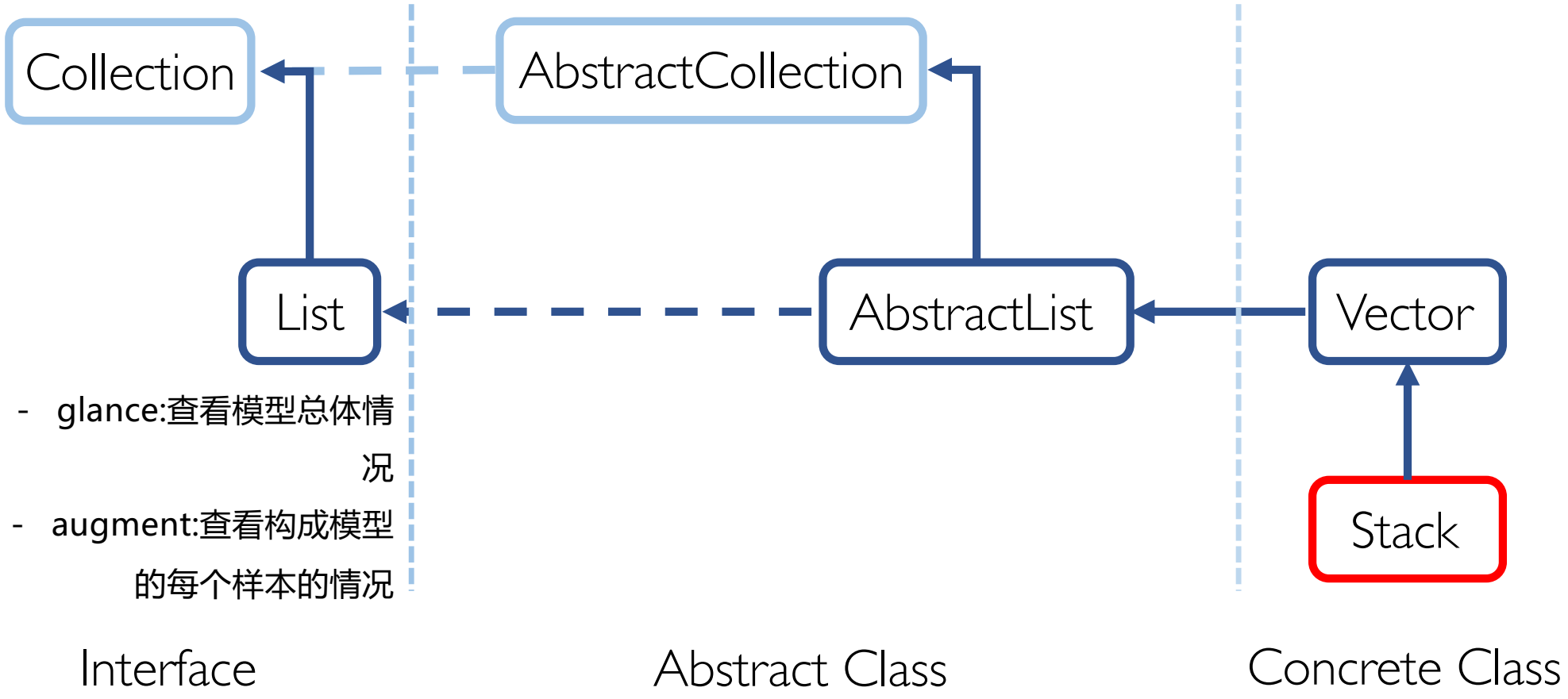
- 在数据准备阶段引入调节参数，调整变量carat阶数；定义了一个新的recipe，并在step_poly里将degree表示为调节参数tune()：

```
dia_rec2 <-
     recipe(price ~ ., data = dia_train) %>%
     step_log(all_outcomes()) %>%
     step_normalize(all_predictors(), -all_nominal()) %>%
     step_dummy(all_nominal()) %>%
     step_poly(carat, degree = tune())
 dia_rec2 %>%
     parameters() %>%
     pull("object")
[[1]]
Polynomial Degree (quantitative)
Range: [1, 3] ##结果可知，carat的阶数取值的合理范围是1到3。最初将整个值设置为5，是会导致过拟合的。
```

07 – workflow

- 将预处理、建模和后处理结合在一起
- 目前，workflow实现的操作：
  - 预处理阶段
    - -通过add_formula()处理一个标准的公式
    - -通过add_recipe()处理一个recipe
  - 模型拟合
    - -结合parsnip的模型指定
  - 后处理（尚待实现的功能）
    - -对类别问题添加概率阈值、校准概率估计、截断可能的预测范围等等

Inheritance



- 回到随机森林的例子。首先，创建一个初始workflow，然后添加数据预处理的recipe和随机森林模型，两者都包含要调整的参数。然后更新参数。

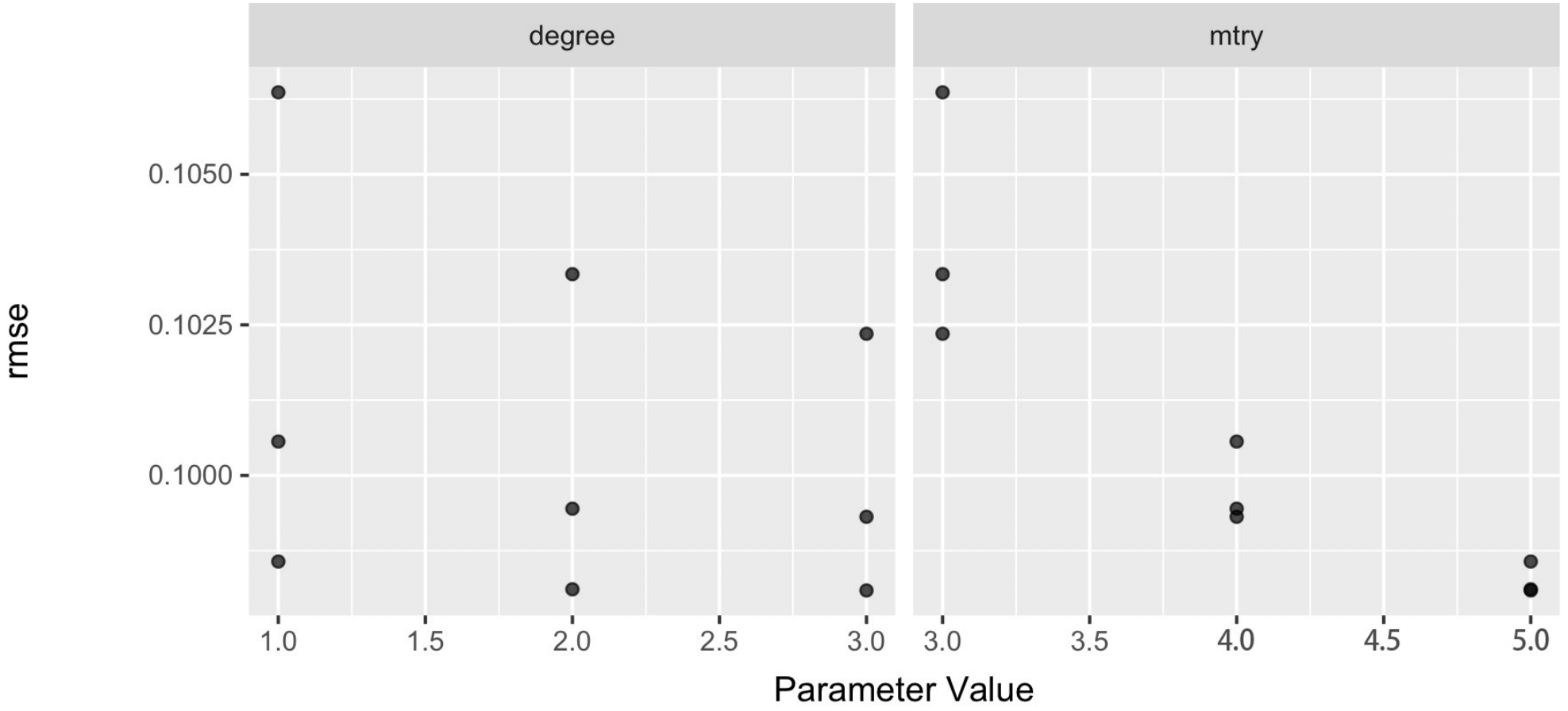
```
rf_wflow <-
    workflow() %>%
    add_model(rf_model1) %>%
    add_recipe(dia_rec2) #创建初始workflow

rf_param <-
    rf_wflow %>%
    parameters() %>%
    update(mtry = mtry(range = c(3L, 5L)),
           degree = degree_int(range = c(1L, 3L)))#更新参数
```

- 使用格点搜索寻找这两个调节参数的最优值（使用dials包的grid_regular()创建格点），通过交叉验证完成这个步骤。
- 两个调节参数根据取值范围形成9个组合，在3折交叉验证上，共有27个模型需要拟合。tune包支持以并行的方式对模型进行拟合。
- 最后，检查结果。tune包的autoplot()函数可以绘制格点搜索的结果。show_best()展示了各个参数组合下估计的模型性能，按照rmse进行排列。select_best()取出其中性能最优的一个（具有最小的rmse）。select_by_one_std_err()根据“一个标准误”准则（one-standard error rule），选择在性能指标优化结果的一个标准误范围内最简单的模型。

- 可知，具有最小rmse估计的结果是(mtry=5,degree=3)，一个标准误内最简单的模型对应的参数是(mtry=4,degree=2)。

```
select_by_one_std_err(rf_search, mtry, degree, metric = "rmse")
## A tibble: 1 x 9
#   mtry degree .metric .estimator   mean     n std_err  .best .bound
#  <int>  <int> <chr>   <chr>       <dbl>  <int>   <dbl>   <dbl>
<dbl>
#1     4      2      rmse   standard  0.0994    3    0.00164 0.0981 0.0995

 select_best(rf_search,metric="rmse")
## A tibble: 1 x 2
#   mtry degree
#  <int>  <int>
#1     5      3
```

- 如下显示了网格检索的可视化结果。

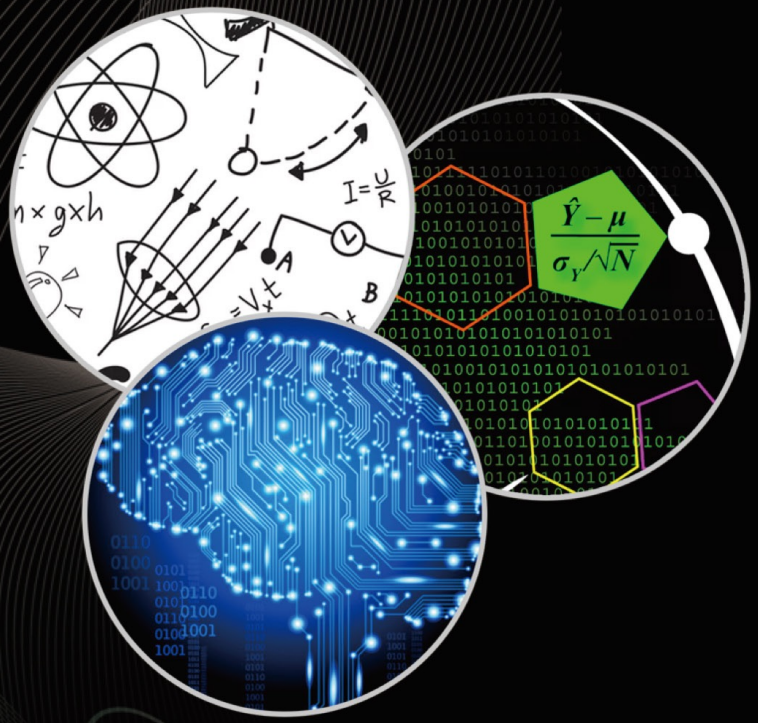


- 我们选择(mtry=4,degree=2)作为最佳的调节参数取值，然后使用fit()在整个训练集上拟合对应的模型。
- 最后，在测试集上执行预测predict()。在预测之前，使用workflows包的pull_prepped_recipe()将拟合过程中训练好的预处理recipe提取出来，通过bake()将它运用到测试数据集dia_test上。

```
metrics(truth=logprice,estimate=.pred)
# A tibble: 3 x 3
#   .metric .estimator .estimate
#   <chr>   <chr>          <dbl>
#1 rmse    standard       0.0978
#2 rsq     standard       0.991
#3 mae     standard       0.0722   ##在测试集上得到的rmse=0.0978，略低于训练集上的效果。
```

08 – Summary

- 在tidyverse系统的基础上，tidymodels尝试将建模（机器学习和统计建模）过程整合成一个标准的、统一的工作流程。从设计目的的角度，tidymodels确实可以让建模过程大大简化，从而让R用户可以从大量不同语法不同数据结构不同实现细节的R包中解脱出来，用30行左右的代码完成一个一般性的建模任务。当然，tidymodels目前依然是高度开发的状态，上述目标还不能完全实现。



THANK YOU!