机器学习与量化交易实战

第五讲

第一次作业

- 有效递交者：
- ruoang@qq.com
- **315135833@qq.com**
- **Zhangyiizhi**
- **273511010**
- **32969472国光**
- **1719466902宝宝**
- **vicky_violin**
- 郭高安
- 任远航
- Victorxu
- 冯斐
- Jared Qu
- Ross

作业格式的提交建议

- 检查入口为ipython notebook的demo
- 如果有自己的较复杂函数，可以单独将.py文件同notebook一同提交，demo中import相关文件即可

Where are we now

- Get the data
- Store the data
- Transform data into training set
- Building predictive models
- Building event driving back-test pipelines

WARNING

- Building Models vs Building （event driving）trading strategies

- 以上二者并不一样，这周关注第一件事（第一件事是第二件事的基础)

建立训练集

How to choose Features(predictors)

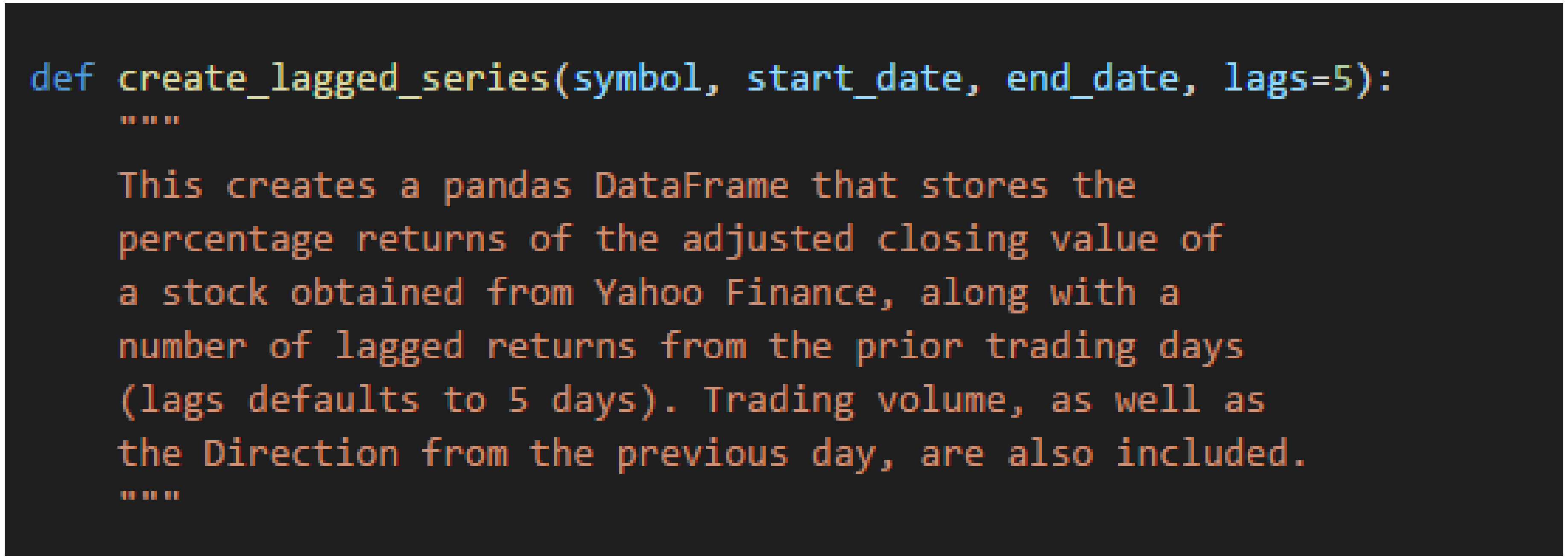
- 选择特征是quant对问题的一种理解
  - 不同的人有不同的信念
  - 不同的信念会造成不同的features选择
  - Pipeline相同
  - 这次课：
    - 选择features的主要流程和基本方法
    - 下次课（save your ass)：如何从选出来的一坨也许不是很好的特征中进行自动筛选和变换

基本方法和原则

- 因子的选择取决于你对问题的理解
  - 举例：你认为沪深三百股指和什么事情有关？

常见的features

- Time lags



```
def create_lagged_series(symbol, start_date, end_date, lags=5):
    """
    This creates a pandas DataFrame that stores the
    percentage returns of the adjusted closing value of
    a stock obtained from Yahoo Finance, along with a
    number of lagged returns from the prior trading days
    (lags defaults to 5 days). Trading volume, as well as
    the Direction from the previous day, are also included.
    """
```

常见的features

$$\left(P_{open}^{(t)} - P_{open}^{(t-1)}\right) / P_{open}^{(t-1)}$$

$$\left(P_{high}^{(t-1)} - P_{high}^{(t-2)}\right) / P_{high}^{(t-2)}$$

$$\left(P_{low}^{(t-1)} - P_{low}^{(t-2)}\right) / P_{low}^{(t-2)}$$

$$\left(V_{open}^{(t)} - V_{open}^{(t-1)}\right) / V_{open}^{(t-1)}$$

$$\left(V_{high}^{(t-1)} - V_{high}^{(t-2)}\right) / V_{high}^{(t-2}$$

$$\left(V_{low}^{(t-1)} - V_{low}^{(t-2)}\right) / V_{low}^{(t-2)}$$

常见的features

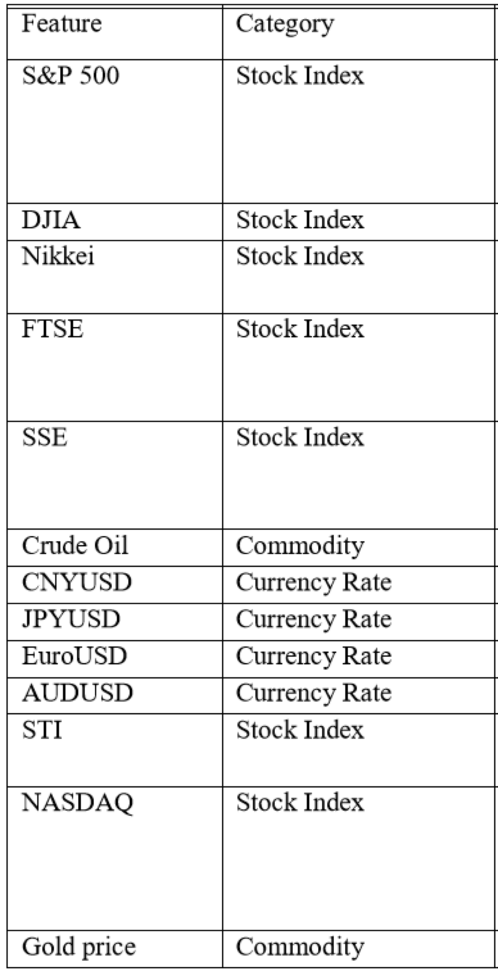
$$
\begin{aligned}
PH_\lambda^{(t)} &= \max_{t-\lambda\le i\le t-1} P_{high}^{(i)} \\
PL_\lambda^{(t)} &= \min_{t-\lambda\le i\le t-1} P_{low}^{(i)} \\
VH_\lambda^{(t)} &= \max_{t-\lambda\le i\le t-1} P_{high}^{(i)} \\
VL_\lambda^{(t)} &= \min_{t-\lambda\le i\le t-1} P_{low}^{(i)}
\end{aligned}
$$

常见的features

$$\frac{\left(P_{open}^{(t)} - P_{open}^{(t-1)}\right)}{PH_{\lambda}^{(t)} - PL_{\lambda}^{(t)}}$$

$$\frac{\left(V_{open}^{(t)} - V_{open}^{(t-1)}\right)}{VH_{\lambda}^{(t)} - VL_{\lambda}^{(t)}}$$

常见的features

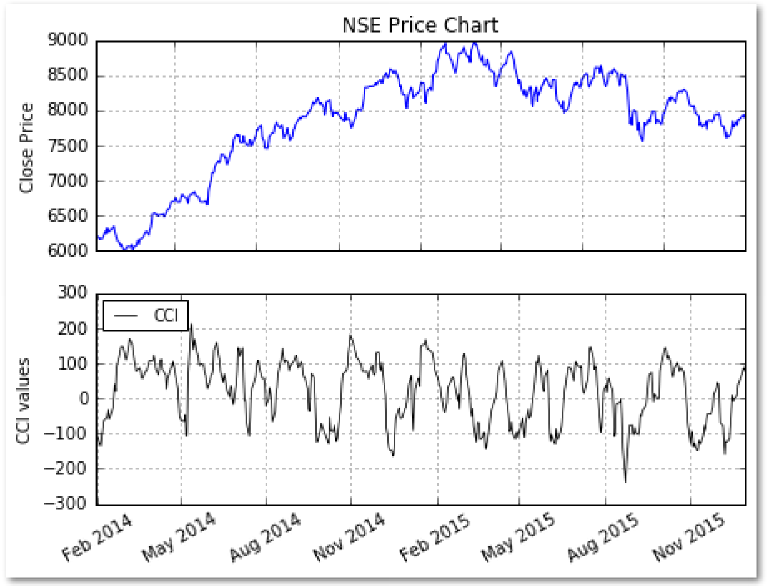


Feature	Category
S&P 500	Stock Index
DJIA	Stock Index
Nikkei	Stock Index
FTSE	Stock Index
SSE	Stock Index
Crude Oil	Commodity
CNYUSD	Currency Rate
JPYUSD	Currency Rate
EuroUSD	Currency Rate
AUDUSD	Currency Rate
STI	Stock Index
NASDAQ	Stock Index
Gold price	Commodity

常见的features

- CCI

- CCI = (Typical price – MA of Typical price) / (0.015 * Standard deviation of Typical price)



常见的features

- CCI can be used to determine overbought and oversold levels. Readings above +100 can imply an overbought condition, while readings below −100 can imply an oversold condition. However, one should be a careful because a security can continue moving higher after the CCI indicator becomes overbought. Likewise, securities can continue moving lower after the indicator becomes oversold.
- Whenever the security is in overbought/oversold levels as indicated by the CCI, there is a good chance that the price will see corrections. Hence a trader can use such overbought/oversold levels to enter in short/long positions.
- Traders can also look for divergence signals to take suitable positions using CCI. A bullish divergence occurs when the underlying security makes a lower low and the CCI forms a higher low, which shows less downside momentum. Similarly, a bearish divergence is formed when the security records a higher high and the CCI forms a lower high, which shows less upside momentum.

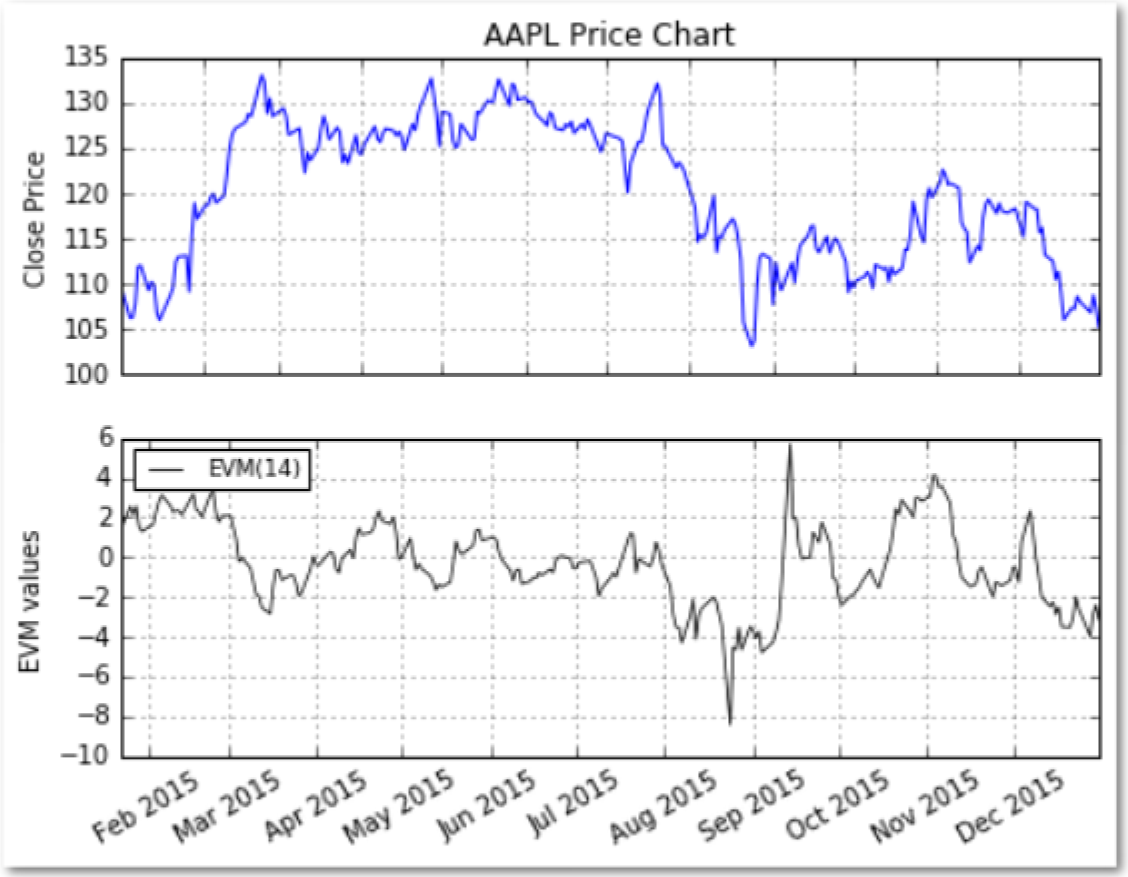
常见的features

- Ease of Movement (EVM)
- Distance moved = ((Current High + Current Low)/2 - (Prior High + Prior Low)/2)

- Ease of Movement (EMV) is a volume-based oscillator which was developed by Richard Arms. EVM indicates the ease with which the prices rise or fall taking into account the volume of the security. For example, a price rise on a low volume means prices advanced with relative ease, and there was little selling pressure. Positive EVM values imply that the market is moving higher with ease, while negative values indicate an easy decline.

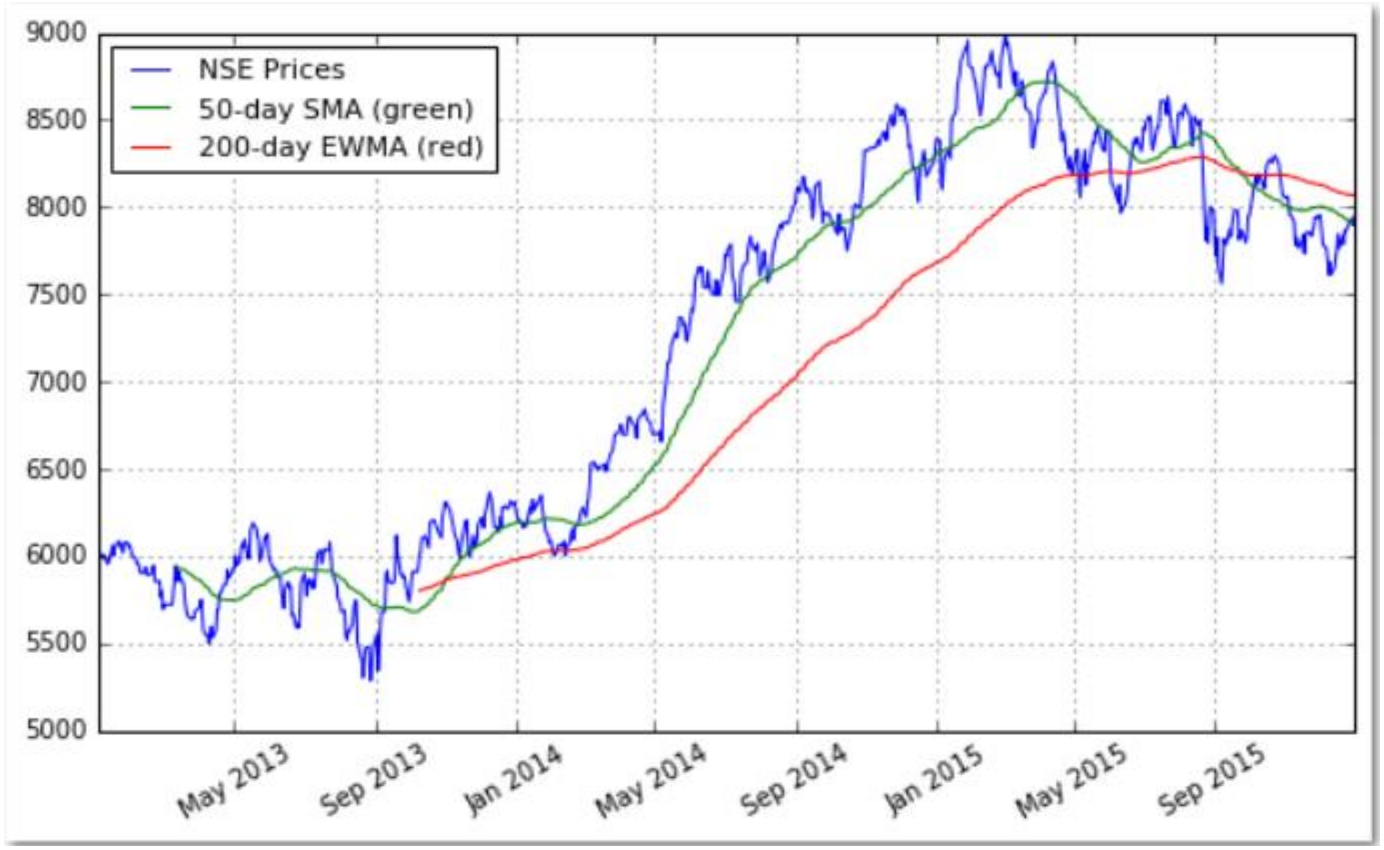
常见的features

- Ease of Movement (EVM)



常见的features

- ma



常见的features

- Rate of Change (ROC)
- The Rate of Change (ROC) is a technical indicator that measures the percentage change between the most recent price and the price “n” day’s ago. The indicator fluctuates around the zero line.
- If the ROC is rising, it gives a bullish signal, while a falling ROC gives a bearish signal. One can compute ROC based on different periods in order to gauge the short-term momentum or the long-term momentum.

常见的features

- Bollinger Bands
- These bands comprise of an upper bollinger band and a lower bollinger band, and are placed two standard deviations above and below a moving average.
- Bollinger bands expand and contract based on the volatility. During a period of rising volatility, the bands widen, and they contract as the volatility decreases. Prices are considered to be relatively high when they move above the upper band and relatively low when they go below the lower band.

常见的features

- **Force Index**
- The force index was created by Alexander Elder. The force index takes into account the direction of the stock price, the extent of the stock price movement, and the volume. Using these three elements it forms an oscillator that measures the buying and the selling pressure.
- Each of these three factors plays an important role in determination of the force index. For example, a big advance in prices, which is given by the extent of the price movement, shows a strong buying pressure. A big decline on heavy volume indicates a strong selling pressure.

And More………

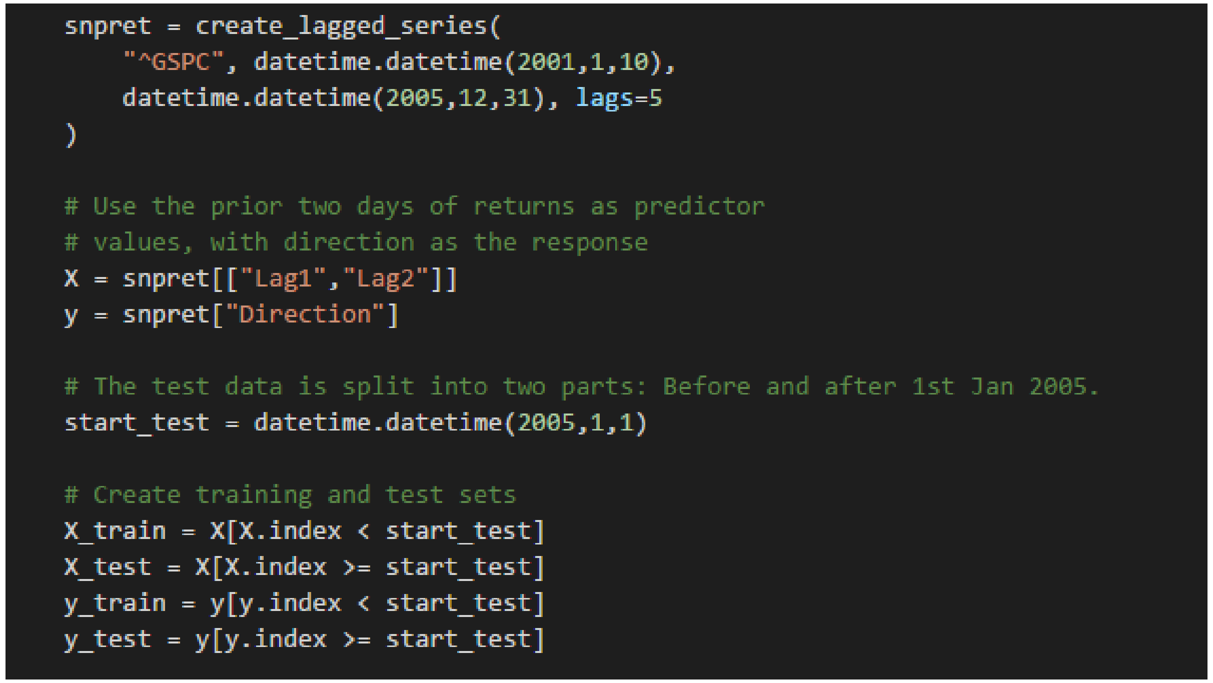
- See
- http://stockcharts.com/school/doku.php?id=chart_school:technical_indicators

作业0

- 建立你自己的feature extracting 库，并命名为FeatureUtils.py
- When in use, just

```
import FeatureUtils
```

建立训练集

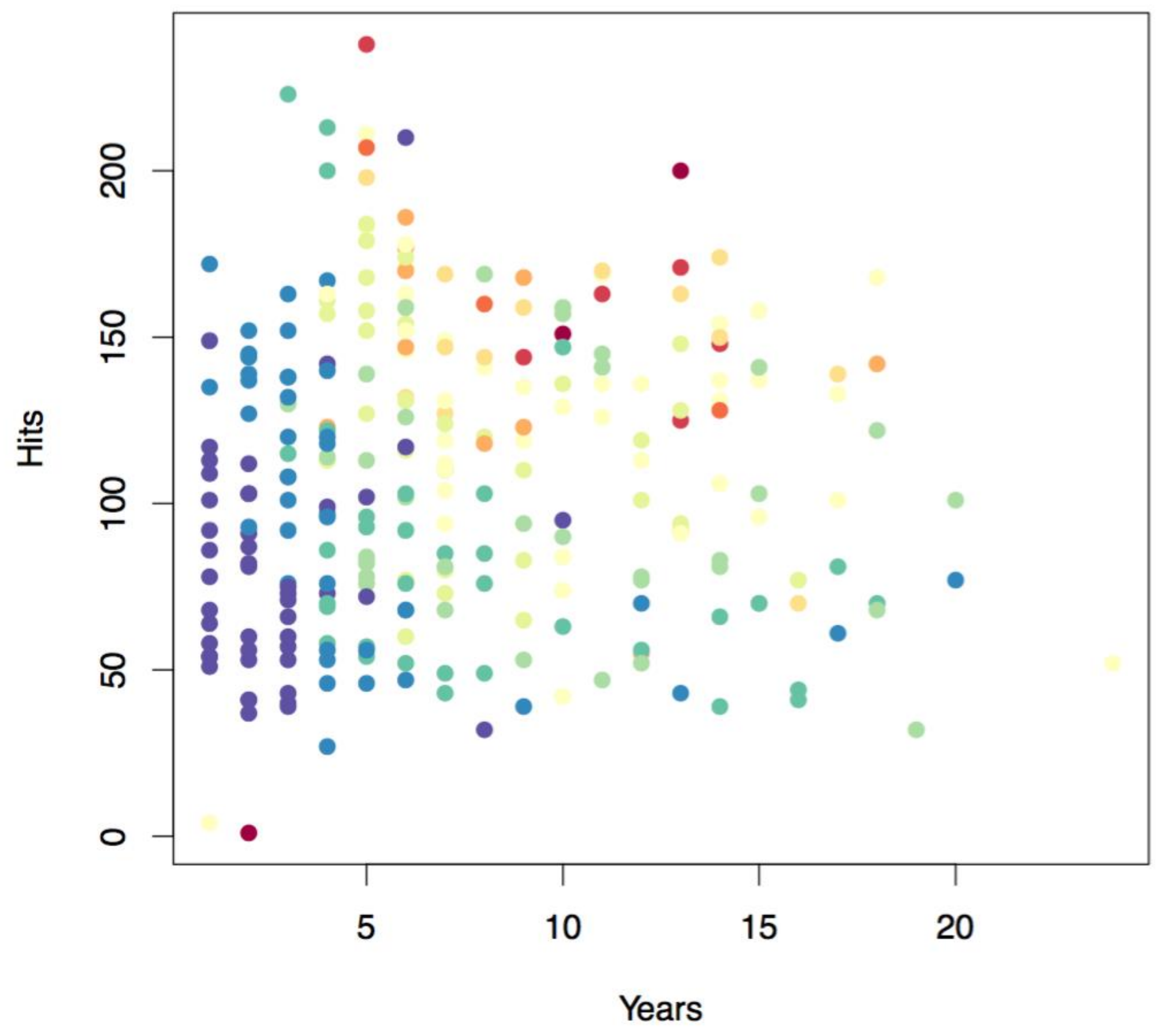


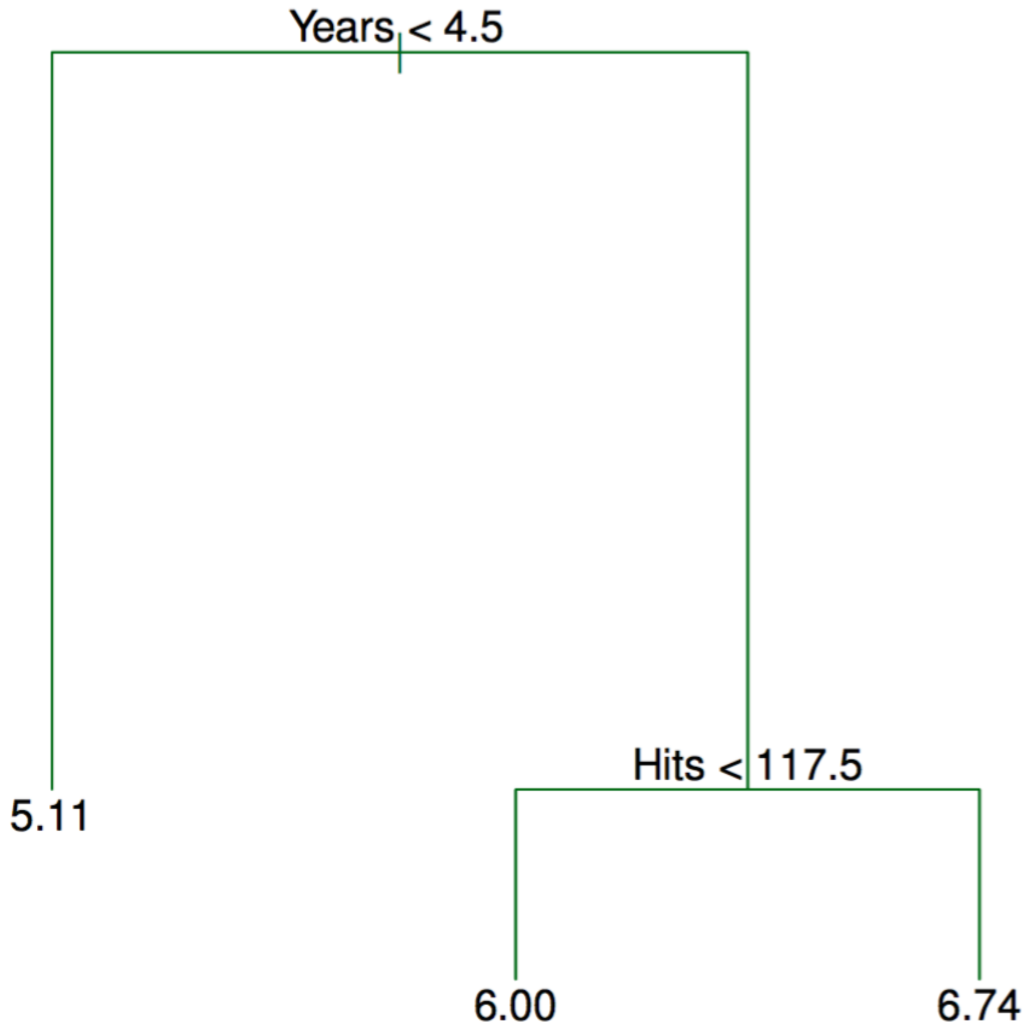
```
snpret = create_lagged_series(
    "^GSPC", datetime.datetime(2001,1,10),
    datetime.datetime(2005,12,31), lags=5
)

# Use the prior two days of returns as predictor
# values, with direction as the response
X = snpret[["Lag1","Lag2"]]
y = snpret["Direction"]

# The test data is split into two parts: Before and after 1st Jan 2005.
start_test = datetime.datetime(2005,1,1)

# Create training and test sets
X_train = X[X.index < start_test]
X_test = X[X.index >= start_test]
y_train = y[y.index < start_test]
y_test = y[y.index >= start_test]
```

modeling

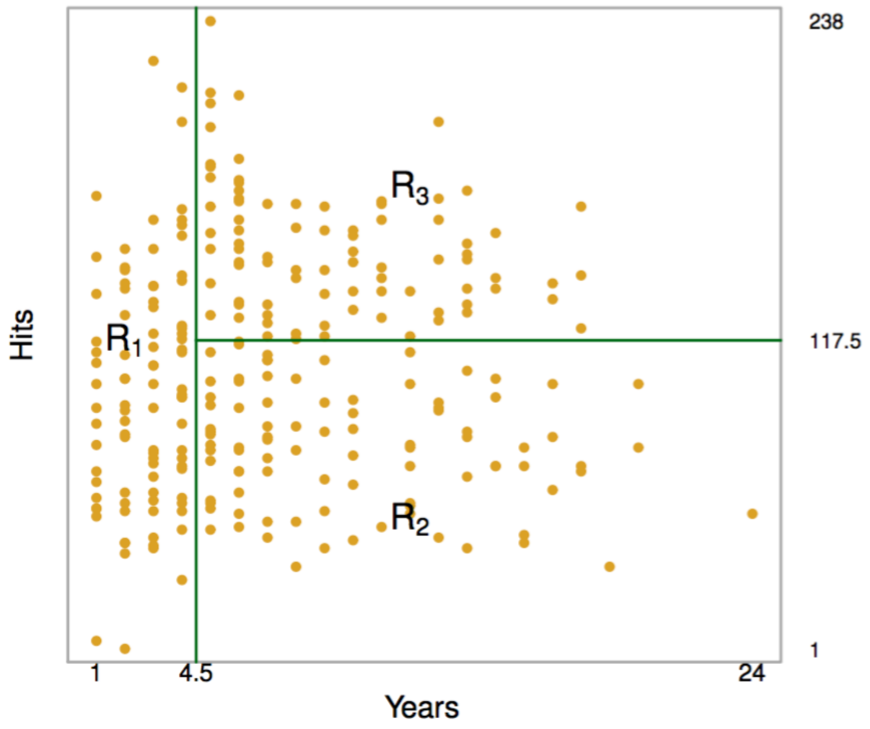
- Random Forests

- Here we describe *tree-based* methods for regression and classification.
- These involve *stratifying* or *segmenting* the predictor space into a number of simple regions.
- Since the set of splitting rules used to segment the predictor space can be summarized in a tree, these types of approaches are known as *decision-tree* methods.


Hits
Years
0
50
100
150
200
5
10
15
20


Years < 4.5
Hits < 117.5
5.11
6.00
6.74

- Overall, the tree stratifies or segments the players into three regions of predictor space: R_1 ={X | `Years`< 4.5}, R_2 ={X | `Years`>=4.5, `Hits`<117.5}, and R_3 ={X | `Years`>=4.5, `Hits`>=117.5}.

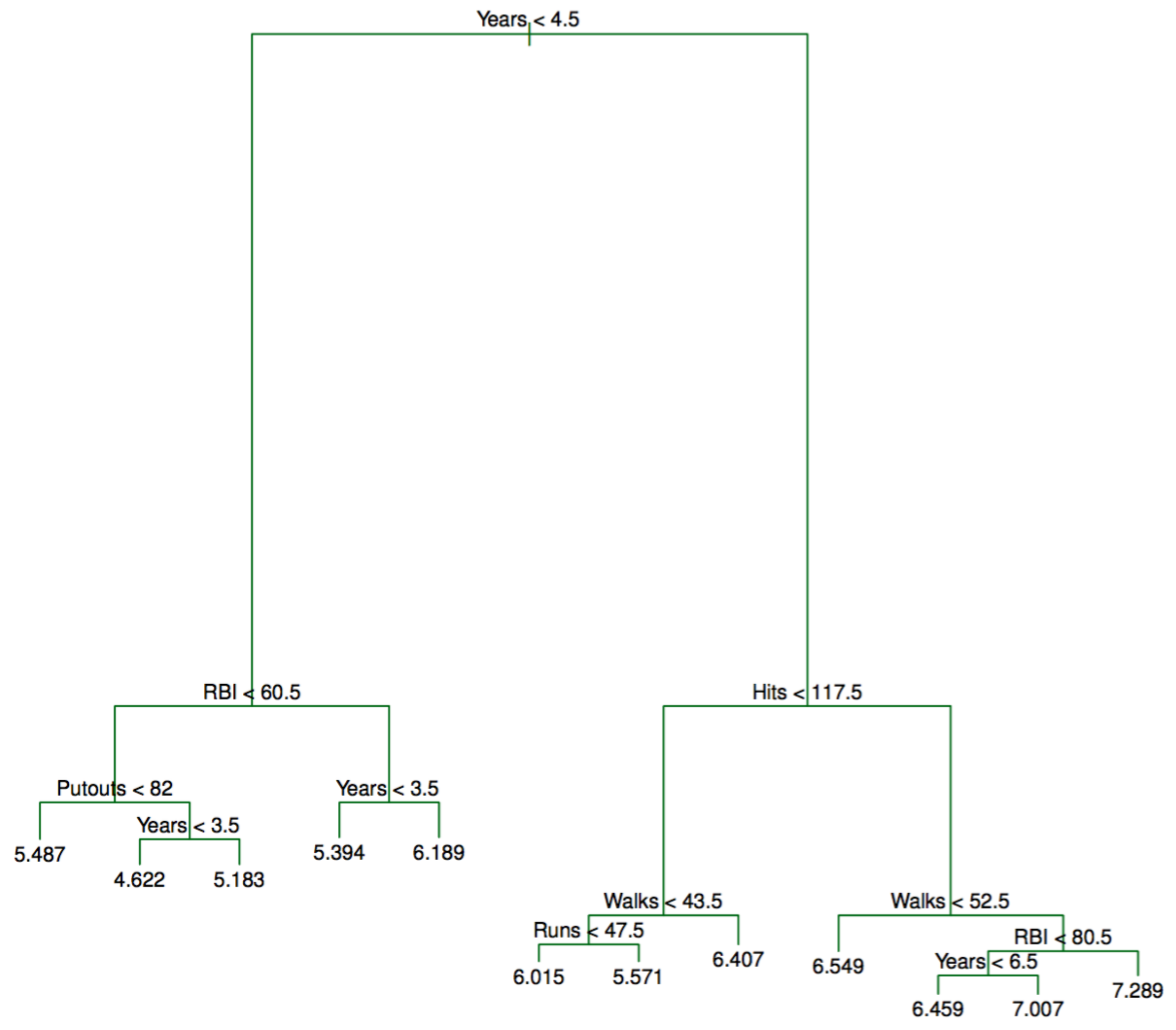


- In theory, the regions could have any shape. However, we choose to divide the predictor space into high-dimensional rectangles, or *boxes*, for simplicity and for ease of interpretation of the resulting predictive model.
- The goal is to find boxes $R_1, \ldots, R_J$ that minimize the RSS, given by

$$\sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2,$$

where $\hat{y}_{R_j}$ is the mean response for the training observations within the jth box.

- We first select the predictor X_j and the cutpoint s such that splitting the predictor space into the regions $\{X|X_j < s\}$ and $\{X|X_j \geq s\}$ leads to the greatest possible reduction in RSS.
- Next, we repeat the process, looking for the best predictor and best cutpoint in order to split the data further so as to minimize the RSS within each of the resulting regions.
- However, this time, instead of splitting the entire predictor space, we split one of the two previously identified regions. We now have three regions.
- Again, we look to split one of these three regions further, so as to minimize the RSS. The process continues until a stopping criterion is reached; for instance, we may continue until no region contains more than five observations.


Years < 4.5
RBI < 60.5
Hits < 117.5
Putouts < 82
Years < 3.5
Years < 3.5
5.487
4.622
5.183
5.394
6.189
Walks < 43.5
Walks < 52.5
Runs < 47.5
6.015
5.571
6.407
6.549
RBI < 80.5
Years < 6.5
6.459
7.007
7.289

- ▲ Trees are very easy to explain to people. In fact, they are even easier to explain than linear regression!
- ▲ Some people believe that decision trees more closely mirror human decision-making than do the regression and classification approaches seen in previous chapters.
- ▲ Trees can be displayed graphically, and are easily interpreted even by a non-expert (especially if they are small).
- ▲ Trees can easily handle qualitative predictors without the need to create dummy variables.
- ▼ Unfortunately, trees generally do not have the same level of predictive accuracy as some of the other regression and classification approaches seen in this book.

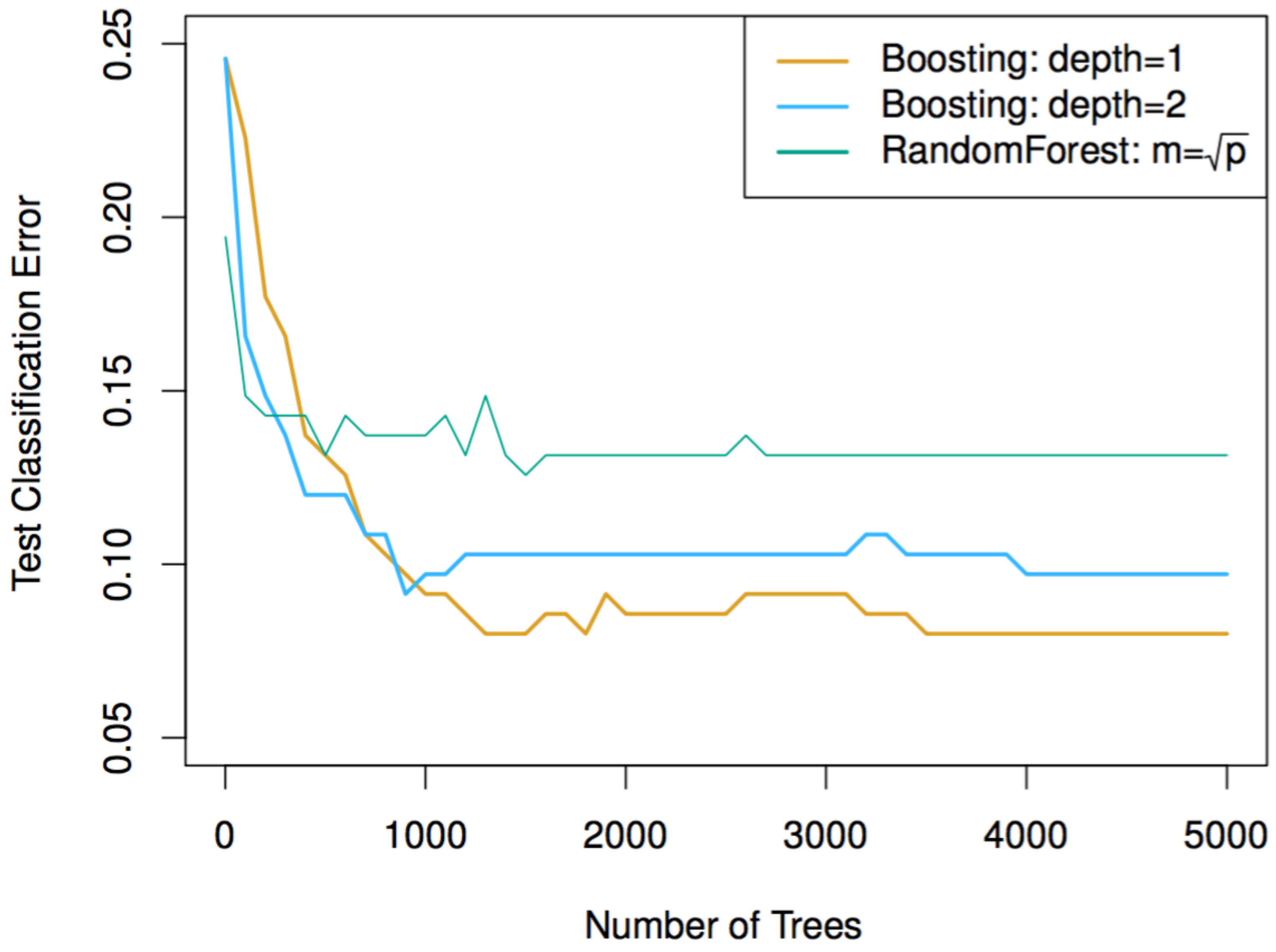
However, by aggregating many decision trees, the predictive performance of trees can be substantially improved. We introduce these concepts next.

- *Random forests* provide an improvement over bagged trees by way of a small tweak that *decorrelates* the trees. This reduces the variance when we average the trees.
- As in bagging, we build a number of decision trees on bootstrapped training samples.
- But when building these decision trees, each time a split in a tree is considered, *a random selection of m predictors* is chosen as split candidates from the full set of p predictors. The split is allowed to use only one of those m predictors.
- A fresh selection of m predictors is taken at each split, and typically we choose $m \approx \sqrt{p}$ — that is, the number of predictors considered at each split is approximately equal to the square root of the total number of predictors (4 out of the 13 for the Heart data).

Boosting algorithm for regression trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:
   $$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$
   - 2.3 Update the residuals,
   $$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$
3. Output the boosted model,
$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

- Unlike fitting a single large decision tree to the data, which amounts to *fitting the data hard* and potentially overfitting, the boosting approach instead *learns slowly*.
- Given the current model, we fit a decision tree to the residuals from the model. We then add this new decision tree into the fitted function in order to update the residuals.
- Each of these trees can be rather small, with just a few terminal nodes, determined by the parameter d in the algorithm.
- By fitting small trees to the residuals, we slowly improve $\hat{f}$ in areas where it does not perform well. The shrinkage parameter λ slows the process down even further, allowing more and different shaped trees to attack the residuals.


Boosting: depth=1
Boosting: depth=2
RandomForest: $m=\sqrt{p}$
Test Classification Error
0.05
0.10
0.15
0.20
0.25
0
1000
2000
3000
4000
5000
Number of Trees

Tuning parameters for boosting

1. The *number of trees* B. Unlike bagging and random forests, boosting can overfit if B is too large, although this overfitting tends to occur slowly if at all. We use cross-validation to select B.
2. The *shrinkage parameter* λ, a small positive number. This controls the rate at which boosting learns. Typical values are 0.01 or 0.001, and the right choice can depend on the problem. Very small λ can require using a very large value of B in order to achieve good performance.
3. The *number of splits* d in each tree, which controls the complexity of the boosted ensemble. Often $d = 1$ works well, in which case each tree is a *stump*, consisting of a single split and resulting in an additive model. More generally d is the *interaction depth*, and controls the interaction order of the boosted model, since d splits can involve at most d variables.

modeling

- Logistic Regression

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

modeling

- LASSO

modeling

- SVMs

modeling

- ANN
- Model architecture and activation functions

modeling

- ANN

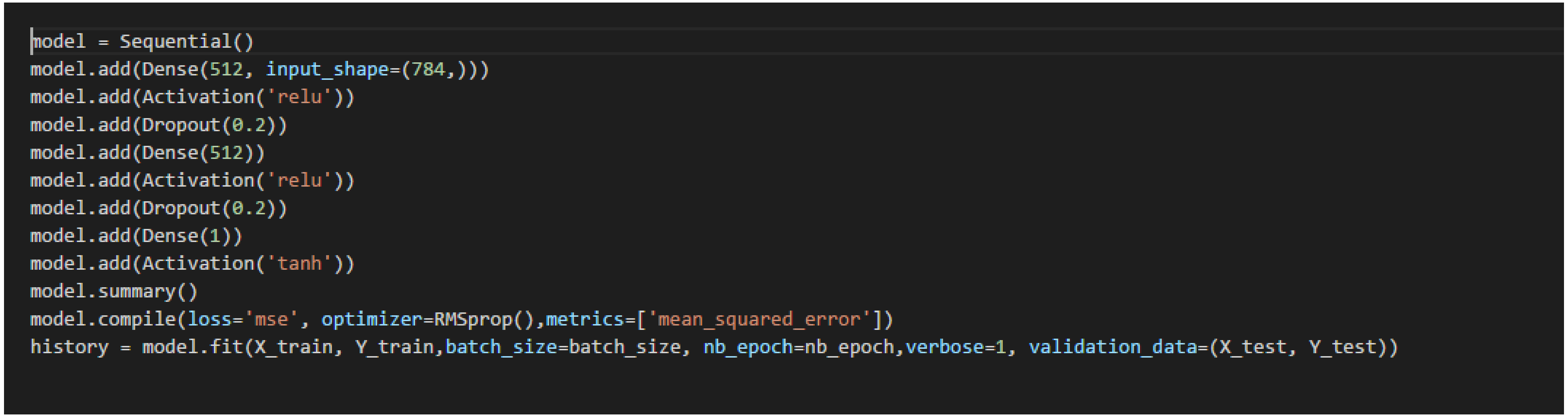
- Loss function
- Cross entropy for classification
- MSE for regression

modeling

- ANN
- Hyper-parameters

modeling

- ANNs



```
model = Sequential()
model.add(Dense(512, input_shape=(784,)))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.2))
model.add(Dense(1))
model.add(Activation('tanh'))
model.summary()
model.compile(loss='mse', optimizer=RMSprop(),metrics=['mean_squared_error'])
history = model.fit(X_train, Y_train,batch_size=batch_size, nb_epoch=nb_epoch,verbose=1, validation_data=(X_test, Y_test))
```

Cross validation

Homework 1

- 1）沪深三百股指的预测分类/回归模型

- 要求：至少10个features，并plot出feature importances
- （http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html#sphx-glr-auto-examples-ensemble-plot-forest-importances-py）
- 模型：需要进行cross validation
- 汇报结果：plot the results with confusion matrix(for classification problems)

回归任务的要求

- Plot出每一个time step的真实值和预测值，真实值为蓝色，预测值为红色

Homework 2

- 1）股票价格的预测分类/回归模型

- 1. price between 10-50
- 2. 沪深三百内的
- 3. average daily volume (ADV) in the middle 33 percentile

Homework 2

根据给定数据，建立回归模型，并汇报r^2值

链接: http://pan.baidu.com/s/1nvDBqlj 密码: 5tc3