# Feature Selection

Machine Learning

Fall 2018

Kasthuri Kannan

- Interpretability vs. Prediction
- Types of feature selection
- Subset selection/Forward/Backward
- Shrinkage (Lasso/Ridge)
- Best model (CV)
- Feature selection vs. Dimension reduction

# Model

- Statistical model:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \varepsilon$$

- How many features to have in the model?

- Prediction accuracy vs. model interpretability

| Less number of features | More number of features |
|---|---|
| Easy to interpret | Difficult to interpret |
| Less likely to over fit | More likely to over fit |
| Low prediction accuracy | High prediction accuracy |

# Feature selection

- Performance of machine learning/statistical models depends on the below:
  - Choice of algorithm
  - Feature selection
  - Feature creation
  - Model selection

- Feature selection is also known as variable selection

# Feature selection

- Basically of three types
  - Filter methods
  - Wrapper methods
  - Embedded methods

# Filter methods

- Some calls this as single factor analysis
- The predictive power of **each individual variable is evaluated**:
  - Correlation with the target variable
  - Information value
  - Chi Square Test (categorical variable)

# Filter methods

- High correlation with target variable
- Low correlation between predictors
- Higher the information value, better is the variable

# Correlation

| Predictor | Y |
|---|---|
| X1 | 0.86 |
| X2 | 0.81 |
| X3 | 0.72 |

| | X1 | X2 | X3 |
|---|---|---|---|
| X1 | 1 | 0.5 | 0.2 |
| X2 | 0.5 | 1 | 0.7 |
| X3 | 0.2 | 0.7 | 1 |

# Information value

Information value is a very useful concept for variable selection during model building.

Similarity information value is a widely used concept of entropy in information theory

# Information value

$$IV = \sum(DistributionGood_i - DistributionBad_i) \times ln(\frac{DistributionGood_i}{DistributionBad_i})$$

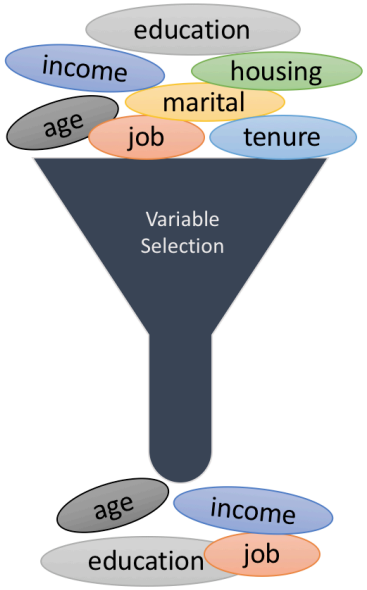
$$Weight of Evidence = ln(\frac{DistributionGood_i}{DistributionBad_i})$$

$$IV = \sum(DistributionGood_i - DistributionBad_i) \times WOE_i$$

| Age Group | Total Number of loans | Number of Bad loans | Number of Good Loans | % Bad loans | Name of Coarse Groups | Distribution of loans | Distribution Bad (DB) | Distribution Good (DG) | WOE | DG - DB | (DG -DB) *WOE |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 21-30 | 4821 | 206 | 4615 | 4.3% | G1 | 0.079 | 0.135 | 0.078 | -0.553 | -0.057 | 0.0318 |
| 30-36 | 10266 | 357 | 9909 | 3.5% | G2 | 0.169 | 0.235 | 0.167 | -0.339 | -0.067 | 0.0228 |
| 36-48 | 32926 | 776 | 32150 | 2.4% | G3 | 0.542 | 0.510 | 0.542 | 0.062 | 0.032 | 0.0020 |
| 48-60 | 12788 | 183 | 12605 | 1.4% | G4 | 0.210 | 0.120 | 0.213 | 0.570 | 0.092 | 0.0527 |
| Total | 60801 | 1522 | 59279 | | | | | | | Information Value --> | 0.1093 |

For the group 21-30, the DB = No. of loans in 21-30/Total number of loans = 0.79 etc.

# Information value




| Information Value (IV) | Predictive Power |
|---|---|
| < 0.02 | useless for prediction |
| 0.02 to 0.1 | weak predictor |
| 0.1 to 0.3 | medium predictor |
| 0.3 to 0.5 | strong predictor |
| > 0.5 | suspicious or too good to be true |

# Chi-square test

```
# Use HouseVotes84 data from  mlbench package
library(mlbench)# For data
library(FSelector)#For method
data(HouseVotes84)

#Calculate the chi square statistics
weights<- chi.squared(Class~., HouseVotes84)

# Print the results
print(weights)

# Select top five variables
subset<- cutoff.k(weights, 5)

# Print the final formula that can be used in classification
f<- as.simple.formula(subset, "Class")
print(f)
```

# Chi-square test

- This data set includes votes for each of the U.S. House of Representatives Congressmen on the 16 key votes identified by the CQA.
  - https://www.rdocumentation.org/packages/mlbench/versions/2.1-1/topics/HouseVotes84

| | Class | V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | V11 | V12 | V13 | V14 | V15 | V16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | republican | n | y | n | y | y | y | n | n | n | y | NA | y | y | y | n | y |
| 2 | republican | n | y | n | y | y | y | n | n | n | n | n | y | y | y | n | NA |
| 3 | democrat | NA | y | y | NA | y | y | n | n | n | n | y | n | y | y | n | n |
| 4 | democrat | n | y | y | n | NA | y | n | n | n | n | y | n | y | n | n | y |
| 5 | democrat | y | y | y | n | y | y | n | n | n | n | y | NA | y | y | y | y |
| 6 | democrat | n | y | y | n | y | y | n | n | n | n | n | n | y | y | y | y |
| 7 | democrat | n | y | n | y | y | y | n | n | n | n | n | n | NA | y | y | y |
| 8 | republican | n | y | n | y | y | y | n | n | n | n | n | n | y | y | NA | y |
| 9 | republican | n | y | n | y | y | y | n | n | n | n | n | y | y | y | n | y |
| 10 | democrat | y | y | y | n | n | n | y | y | y | n | n | n | n | n | NA | NA |
| 11 | republican | n | y | n | y | y | n | n | n | n | n | NA | NA | y | y | n | n |
| 12 | republican | n | y | n | y | y | y | n | n | n | n | y | NA | y | y | NA | NA |
| 13 | democrat | n | y | y | n | n | n | y | y | y | n | n | n | y | n | NA | NA |

# Chi-square test

```
> print(weights)
    attr_importance
V1        0.409330348
V2        0.004534049
V3        0.748864321
V4        0.923255954
V5        0.718768923
V6        0.428332508
V7        0.521967369
V8        0.661876085
V9        0.629797943
V10       0.083809300
V11       0.378240781
V12       0.714922593
V13       0.555971176
V14       0.625283342
V15       0.538263037
V16       0.353273580
```

```
> subset<- cutoff.k(weights, 5)
> f<- as.simple.formula(subset, "Class")
> print(f)
Class ~ V4 + V3 + V5 + V12 + V8
```

# Wrapper methods

- Predictive power of the variables in evaluated jointly

- Set of variables that performs the best:

  - Subset selection

  - Forward selection

  - Backward selection

# Embedded methods

- Inbuilt variable selection methods (without one having to select/reject feature)

- Regularization
  - Controls the value of the parameter
  - Not so important variables are given very low weight (close to zero)
  - Lasso and Ridge regression
  - This is also known as Shrinkage method

# Optimal number of features?

- The ideal model should do justice to both:
  - Good prediction yet not overly complex to interpret and use

- One way to do is to select the best set of features:
  - Subset selection
  - Shrinkage
  - Dimension reduction

# Subset selection

- Fit models with each possible combinations of the *p* features

- Total number of models: $2^p$

- If $p = 2$

$$Y = A_0$$
$$Y = B_0 + B_1 X_1$$
$$Y = C_0 + C_1 X_2$$
$$Y = D_0 + D_1 X_1 + D_2 X_2$$

# Subset selection

- Requires massive computational power
- To reduce the computational power its broken into two stages:
  - Stage 1: Fit all combination of models that has only *k* predictors out of total *P* predictors.
    - Pick the best model from the set of all *k* predictors models (lets call this *Model(k)*)
  - Stage 2: Select the one that's the best from *Model(1)*, *Model(2)*, …, *Model(p)*
- Use RSS, CV error, Adjusted R-square

# Using test error

- RSS & R square has monotonic relationship with the number of variables

- If we use training error for selection we might end up selecting the model that exactly has p variables

- Hence use **Test RSS for Regression Problems & Test Deviance for Classification Problems**

# Stepwise selection

- P > 20 it is almost impossible to use best subset selection
  - Forward Stepwise Approach
  - Backward Stepwise Approach
  - Hybrid

# Forward Stepwise

- Start with a null model
- Add predictors to the model **one at a time**. Choose the best model among the best for each k based on RSS
- **If a variable is retained it never drops from the model**

| | Subset selection | Forward stepwise |
|---|---|---|
| One variable | X1 | **X1** |
| Two variables | X1 X3 | **X1** X2 |
| Three variables | X1 X3 X4 | **X1 X2** X4 |
| Four variables | X1 X2 X3 X5 | **X1 X2 X4** X5 |

# Backward selection

- It is the reverse of forward:
  - Start with all predictors and then drop one at a time and then select the best model

| | Backward stepwise | Forward stepwise |
|---|---|---|
| | X1 **X2** X3 X4 X5 | **X1** |
| | X1 X3 **X4** X5 | **X1** X2 |
| | X1 **X3** X5 | **X1 X2** X4 |
| | X1 **X5** | **X1 X2 X4** X5 |
| | **X1** | **X1 X2 X4 X3 X5** |

# Backward

- Computational power requirement is similar as that of forward selection

- Selection is made through RSS or Deviance

# Hybrid

- Combines forward and backward
- Forward: Variable included never drops off
- Backward: Variable already dropped from the model never show up in the model again
- Hybrid starts with adding one variable at a time **like forward** but **drop variables during the process if a variable no longer improve fit**

# Hybrid

- It is more like subset selection − consider more models compared to forward and backward

- It retains the computational advantage of forward and backward selection

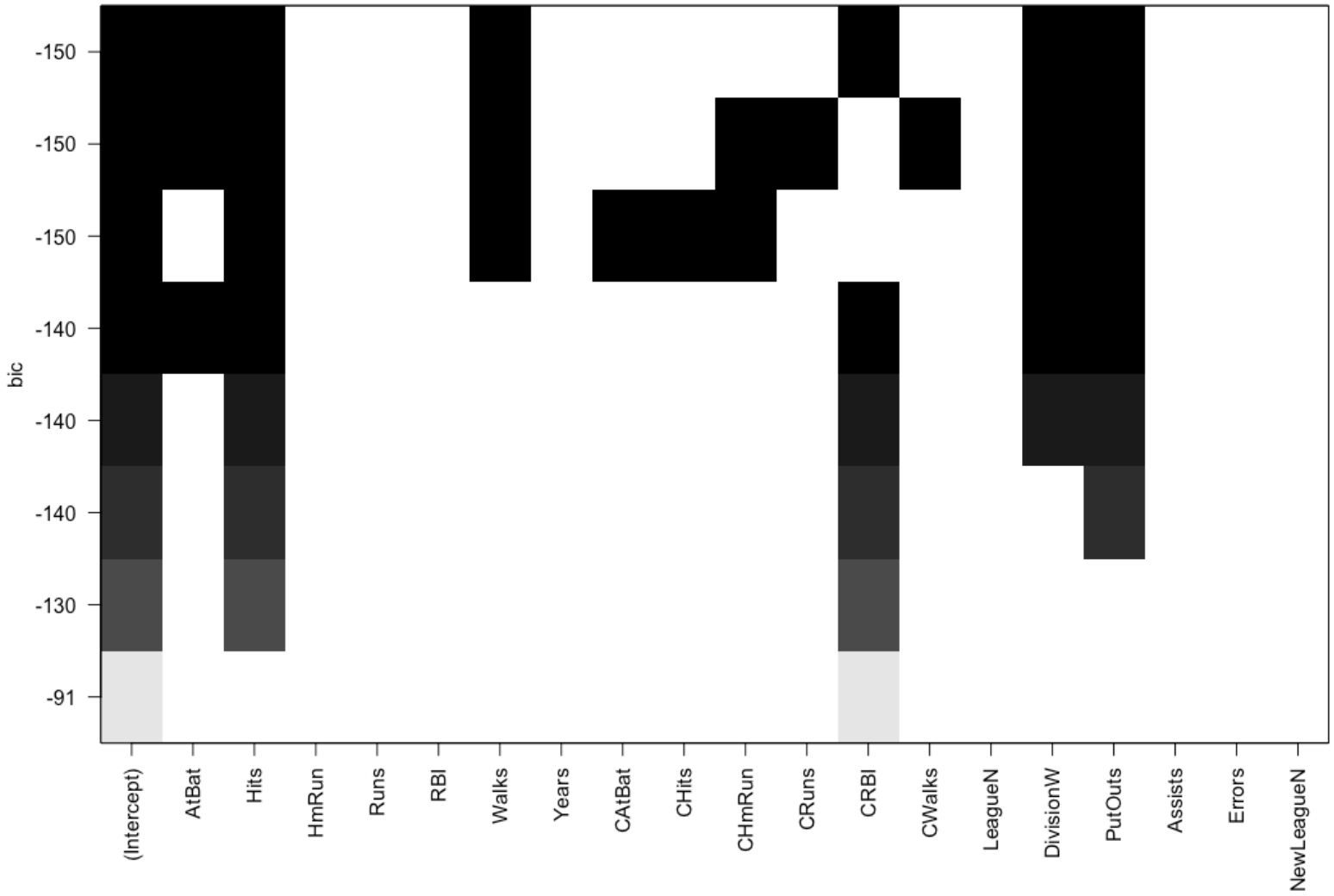
```
> library(ISLR)
> names(Hitters)
 [1] "AtBat"     "Hits"      "HmRun"     "Runs"      "RBI"       "Walks"     "Years"     "CAtBat"    "CHits"     "CHmRun"    "CRuns"
[12] "CRBI"      "CWalks"    "League"    "Division"  "PutOuts"   "Assists"   "Errors"    "Salary"    "NewLeague"
> dim(Hitters)
[1] 263  20
> library(leaps)
> fit <- regsubsets(Salary~.,Hitters,nvmax=8)
> summary(fit)
Subset selection object
Call: regsubsets.formula(Salary ~ ., Hitters, nvmax = 8)
19 Variables  (and intercept)
           Forced in Forced out
AtBat          FALSE      FALSE
Hits           FALSE      FALSE
HmRun          FALSE      FALSE
Runs           FALSE      FALSE
RBI            FALSE      FALSE
Walks          FALSE      FALSE
Years          FALSE      FALSE
CAtBat         FALSE      FALSE
CHits          FALSE      FALSE
CHmRun         FALSE      FALSE
CRuns          FALSE      FALSE
CRBI           FALSE      FALSE
CWalks         FALSE      FALSE
LeagueN        FALSE      FALSE
DivisionW      FALSE      FALSE
PutOuts        FALSE      FALSE
Assists        FALSE      FALSE
Errors         FALSE      FALSE
NewLeagueN     FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
1  ( 1 ) " "   " "  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"  " "    " "     " "       " "     " "     " "    " "
2  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"  " "    " "     " "       " "     " "     " "    " "
3  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"  " "    " "     " "       "*"     " "     " "    " "
4  ( 1 ) " "   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"  " "    " "     "*"       "*"     " "     " "    " "
5  ( 1 ) "*"   "*"  " "   " "  " " " "   " "   " "    " "   " "    " "   "*"  " "    " "     "*"       "*"     " "     " "    " "
6  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   " "    " "   "*"  " "    " "     "*"       "*"     " "     " "    " "
7  ( 1 ) " "   "*"  " "   " "  " " "*"   " "   "*"    "*"   "*"    " "   " "  " "    " "     "*"       "*"     " "     " "    " "
8  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   " "    " "   "*"    "*"   " "  "*"    " "     "*"       "*"     " "     " "    " "
```

# Selecting model

```
> summary(fit)$rsq
[1] 0.3214501 0.4252237 0.4514294 0.4754067
0.4908036 0.5087146 0.5141227 0.5285569
```

# BIC graph

> plot(fit, scale="bic")

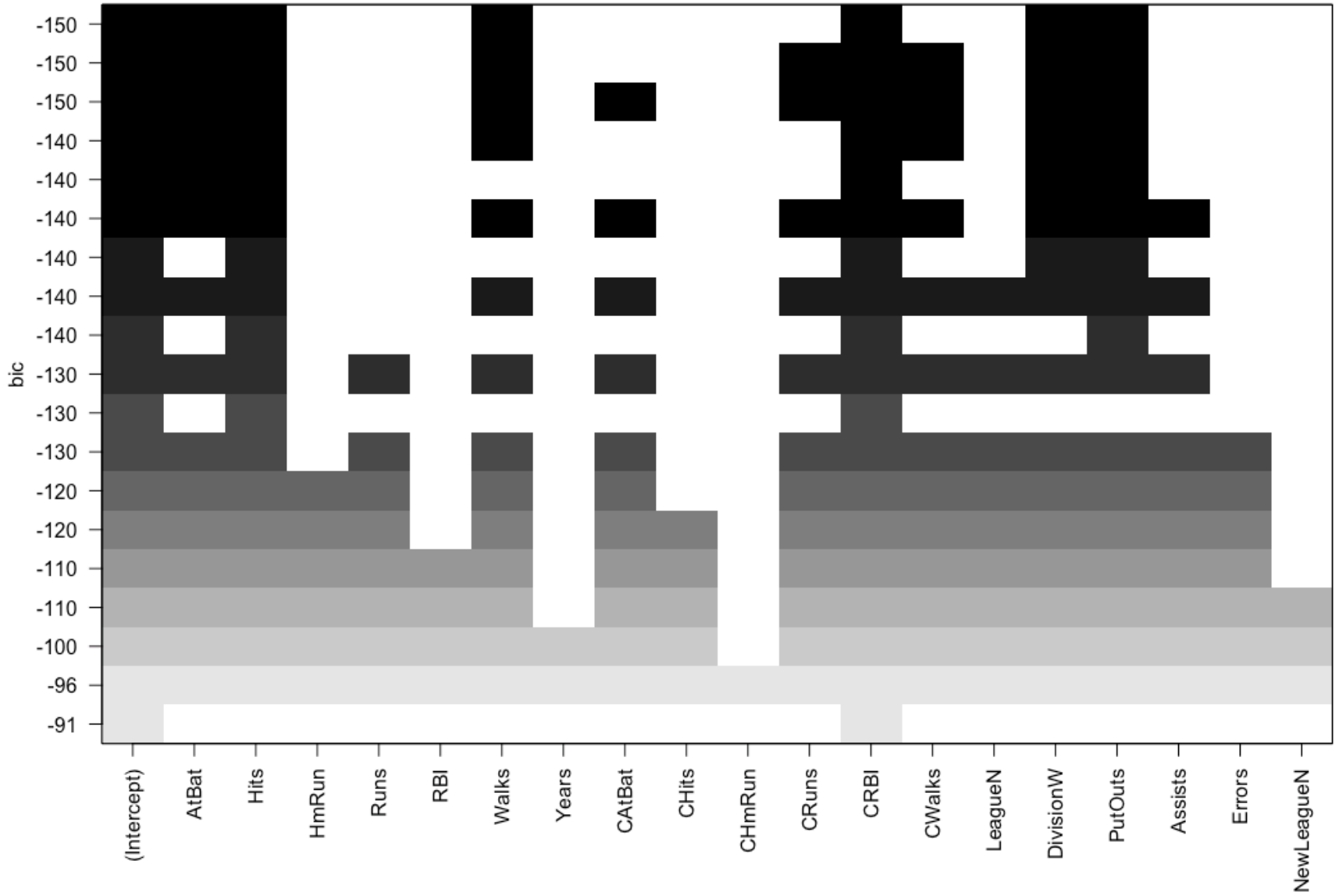


# Forward and Backward

```
> fit <- regsubsets(Salary~.,data=Hitters, nvmax=19, method="forward")
> summary(fit)
```

```
Selection Algorithm: forward
          AtBat Hits HmRun Runs RBI Walks Years CAtBat CHits CHmRun CRuns CRBI CWalks LeagueN DivisionW PutOuts Assists Errors NewLeagueN
1  ( 1 )  " "   " "  " "   " "  " " " "   " "   " "   " "   " "   " "   "*"  " "   " "     " "       " "     " "     " "    " "
2  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "   " "   " "   " "   "*"  " "   " "     " "       " "     " "     " "    " "
3  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "   " "   " "   " "   "*"  " "   " "     " "       "*"     " "     " "    " "
4  ( 1 )  " "   "*"  " "   " "  " " " "   " "   " "   " "   " "   " "   "*"  " "   " "     "*"       "*"     " "     " "    " "
5  ( 1 )  "*"   "*"  " "   " "  " " " "   " "   " "   " "   " "   " "   "*"  " "   " "     "*"       "*"     " "     " "    " "
6  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "   " "   " "   " "   "*"  " "   " "     "*"       "*"     " "     " "    " "
7  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "   " "   " "   " "   "*"  "*"   " "     "*"       "*"     " "     " "    " "
8  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   " "   " "   " "   "*"   "*"  "*"   " "     "*"       "*"     " "     " "    " "
9  ( 1 )  "*"   "*"  " "   " "  " " "*"   " "   "*"   " "   " "   "*"   "*"  "*"   " "     "*"       "*"     " "     " "    " "
10  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"   " "   " "   "*"   "*"  "*"   " "     "*"       "*"     "*"     " "    " "
11  ( 1 ) "*"   "*"  " "   " "  " " "*"   " "   "*"   " "   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     " "    " "
12  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"   " "   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     " "    " "
13  ( 1 ) "*"   "*"  " "   "*"  " " "*"   " "   "*"   " "   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     "*"    " "
14  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"   " "   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     "*"    " "
15  ( 1 ) "*"   "*"  "*"   "*"  " " "*"   " "   "*"   "*"   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     "*"    " "
16  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"   "*"   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     "*"    " "
17  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   " "   "*"   "*"   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     "*"    "*"
18  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"   "*"   " "   "*"   "*"  "*"   "*"     "*"       "*"     "*"     "*"    "*"
19  ( 1 ) "*"   "*"  "*"   "*"  "*" "*"   "*"   "*"   "*"   "*"   "*"   "*"  "*"   "*"     "*"       "*"     "*"     "*"    "*"
```

# BIC – forward selection



# Embedded methods (Shrinkage)

- Regularized regression models – A technique that regularize the estimates or shrink the co-efficient towards zero

- Slight modification to the least square estimation

$$RSS = \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_i x_{ij})^2$$

$$\min_{\beta_j} \left[ RSS + \lambda \sum_{j=1}^{p} \beta_j^2 \right]$$

# Embedded methods (Shrinkage)

$$\min_{\beta_j} \left[ RSS + \lambda \sum_{j=1}^{p} \left| \beta_j \right| \right]$$

- Lasso
  - Variable selection property of lasso
  - Beta = 0 for unimportant variables
  - How to choose lambda?
    - Cross-validation

# Dimension reduction vs. Feature selection

- Feature selection
  - Automatic
  - Univariate
  - Subset
  - Stepwise

- Dimension Reduction
  - Principal Component Analysis
  - [X1 X2 X3 X4] −> [Z1, Z2]