PSY9511: Seminar 3

Regularization and variable selection

Esten H. Leonardsen

11.09.24

Outline

1. Assignment 1
2. Assignment 2
3. Regularization
    - Variable selection
    - Shrinkage (+ live coding 🥳)
    - Dimensionality reduction

Assignment 1

Assignment 1: Coding

- Create a vector of 100 standard normally distributed numbers and visualize them with a histogram.
- Show rows 5, 8, 9, and 10 of the Auto dataset.
- Show the last three columns of the Auto dataset.
- Show all cars with five cylinders in the Auto dataset.

Assignment 1: Coding

- Create a vector of 100 standard normally distributed numbers and visualize them with a histogram.
- Show rows 5, 8, 9, and 10 of the Auto dataset.
- Show the last three columns of the Auto dataset.
- Show all cars with five cylinders in the Auto dataset.

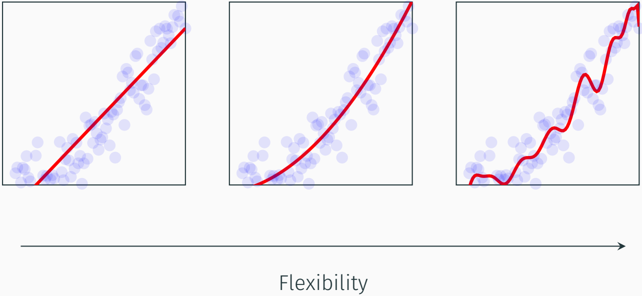
http://localhost:8889/notebooks/notebooks%2FAssignment%201.ipynb

Assignment 2: Bias-variance trade-off

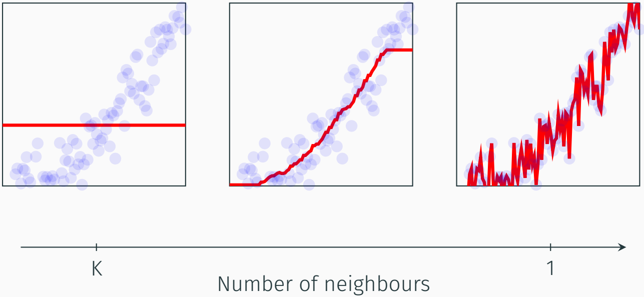
Assignment 2: Bias-variance trade-off

Flexibility

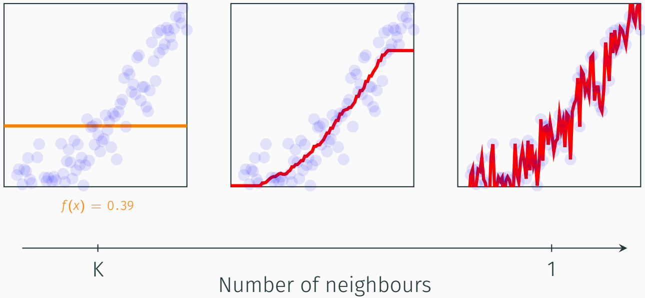
Assignment 2: Bias-variance trade-off



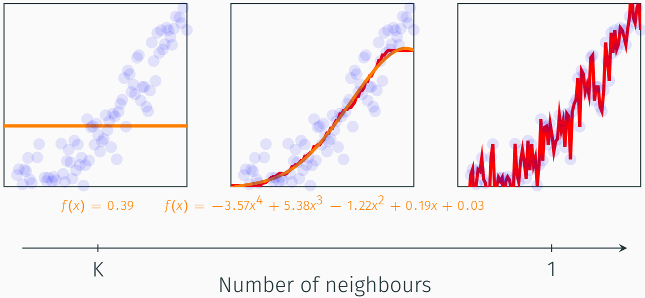
Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off

$f(x) = 0.39$ $\quad f(x) = -3.57x^4 + 5.38x^3 - 1.22x^2 + 0.19x + 0.03$

Assignment 2: Bias-variance trade-off

$f(x) = 0.39$ $\quad$ $f(x) = -3.57x^4 + 5.38x^3 - 1.22x^2 + 0.19x + 0.03$

$\uparrow$

1

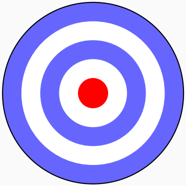
$$f(x) = \underset{\substack{\uparrow \\ 1}}{0.39} \qquad f(x) = -\underset{\substack{\uparrow \\ 1}}{3.57}x^4 + \underset{\substack{\uparrow \\ 2}}{5.38}x^3 - \underset{\substack{\uparrow \\ 3}}{1.22}x^2 + \underset{\substack{\uparrow \\ 4}}{0.19}x + \underset{\substack{\uparrow \\ 5}}{0.03}$$

Assignment 2: Bias-variance trade-off

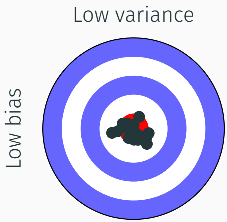
Model flexibility: Denotes the complexity of the approximated function $\hat{y} = \hat{f}(x)$.

- Informally: Wigglyness of the line
- Formally: Number of parameters in the function (degrees of freedom)

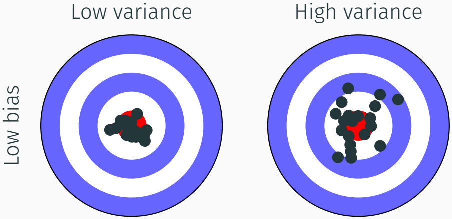
Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off

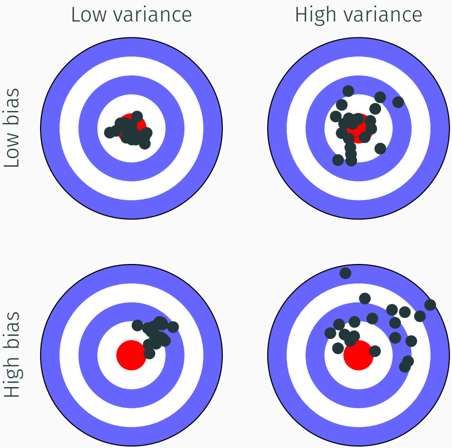


Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off

Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off

Flexibility

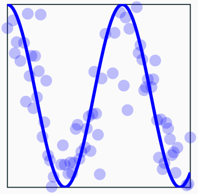
Assignment 2: Bias-variance trade-off

High bias

High variance

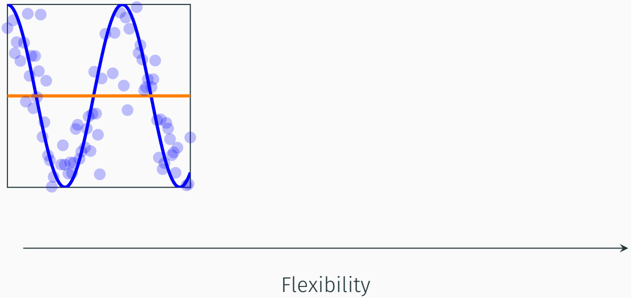
Flexibility

Assignment 2: Bias-variance trade-off

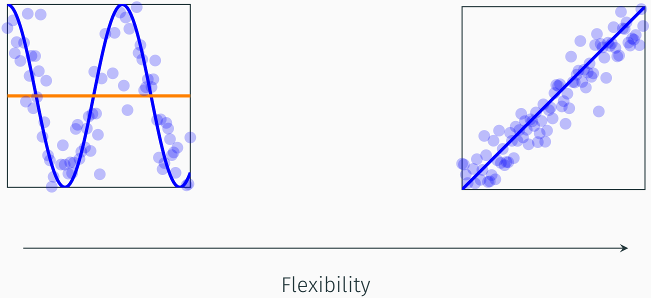


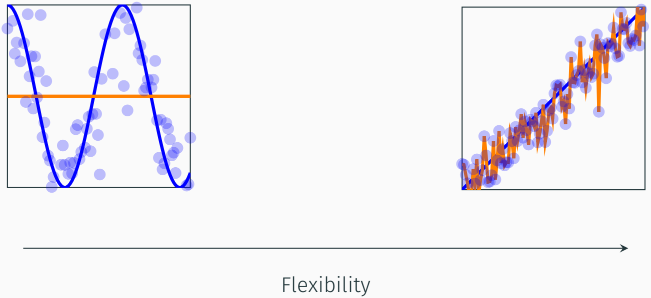
Flexibility

Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off



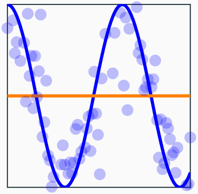

Flexibility

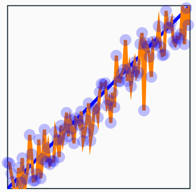
Assignment 2: Bias-variance trade-off

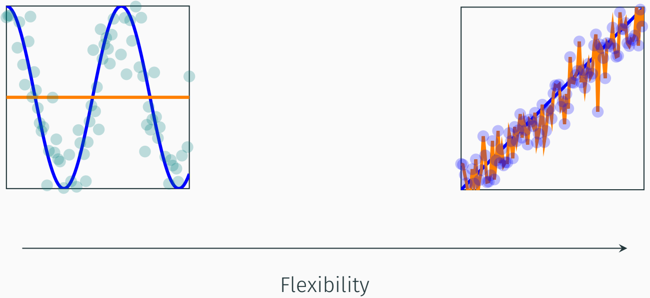
Bias and variance: Two ways the model can be bad

- High bias: The model misses in *systematic* ways
- High variance: The model misses in *unsystematic* ways

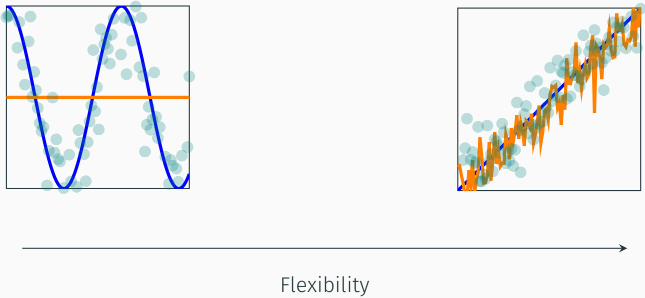
Assignment 2: Bias-variance trade-off






Flexibility

Assignment 2: Bias-variance trade-off



Assignment 2: Bias-variance trade-off

Underfitting and overfitting: Bias-variance trade-off in practice

- Underfitting: The model is equally bad on training and test data *due to not having captured the true relationship between inputs and outputs*
- Overfitting: The model is good on training data, but bad on test data *because it has found patterns in the noise during training*

Assignment 2

UNIVERSITETET
I OSLO

Assignment 2

- Download the Auto.csv dataset from the ISLP website.
- Read the Auto.csv-dataset into memory.
- In the horsepower-column, some values are missing. These are encoded with '?'. Remove these rows from the dataset.
- Create a new column 'muscle'. This column should contain a 1 for all muscle cars (e.g. cars that have above average horsepower) and 0 for the rest.
- Split the dataset into a training set and a test set, by randomly drawing 80% of the rows for the former and 20% of the rows for the latter.

http://localhost:8890/notebooks/notebooks/Assignment%202.ipynb

- Fit a simple linear regression model using horsepower as the predictor and mpg as the outcome using the training data.
- Create a scatter plot with horsepower on the x-axis and mpg on the y-axis using the testing data. Plot the regression line found by the model in the plot.
- Use the model to generate predictions for the training set. Calculate and report the mean absolute error (MAE) of these predictions.
- Use the model to generate predictions for the test set. Calculate and report the MAE of the predictions.
- Reflection: Is the training and testing MAE is lower? Does this match your expectation? What would be the general pattern we expect here (e.g. one is lower than the other, they are the same, etc.), and why do we expect that?

http://localhost:8890/notebooks/notebooks/Assignment%202.ipynb

Assignment 2

- Fit a multivariate linear regression model using horsepower, weight, displacement, and year as predictors and mpg as the outcome.
- Print the intercept and coefficients of the model .
- Use the model to generate predictions for the training set. Calculate and report the MAE of these predictions.
- Use the model to generate predictions for the test set. Calculate and report the MAE of the predictions.
- Reflection: Is the training MAE lower or higher than in the simple linear regression model? Does it have to be this way, or could it have been otherwise? What about the testing MAE?

http://localhost:8890/notebooks/notebooks/Assignment%202.ipynb

Assignment 2

- Fit a logistic regression model using weight, displacement and year as predictors and our newly created muscle-column as the outcome. Why don't we use horsepower as a predictor in this model?
- Use the model to generate predictions for the training set. Calculate and report the accuracy of these predictions.
- Use the model to generate predictions for the testing set. Calculate and report the accuracy of these predictions.

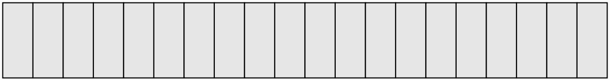
`http://localhost:8890/notebooks/notebooks/Assignment%202.ipynb`

Assignment 2: Data splitting

In[1]:
```python
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.sample(frac=0.2)
```

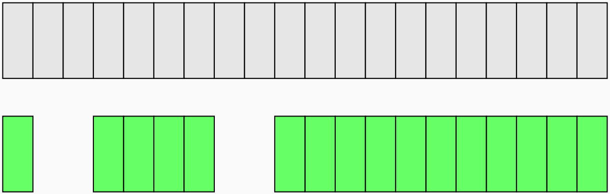
Assignment 2: Data splitting

In[1]:
```
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.sample(frac=0.2)
```



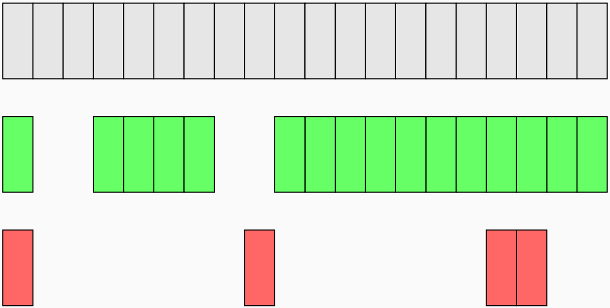
Assignment 2: Data splitting

In[1]:
```
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.sample(frac=0.2)
```



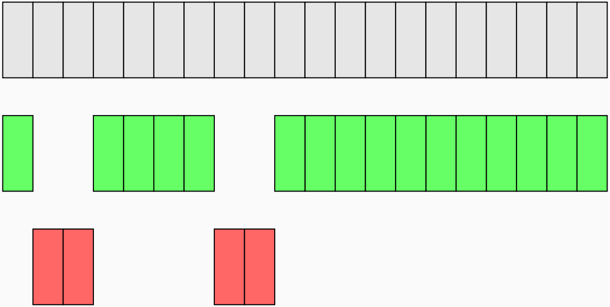
Assignment 2: Data splitting

In[1]:
```
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.sample(frac=0.2)
```



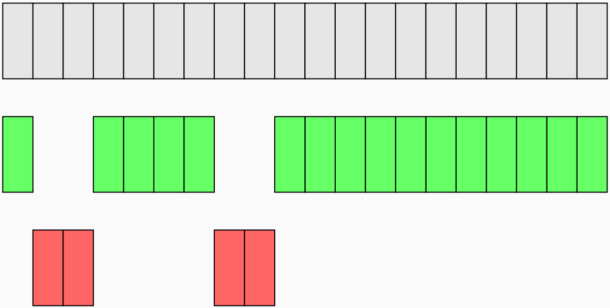
Assignment 2: Data splitting

In[1]:
```
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



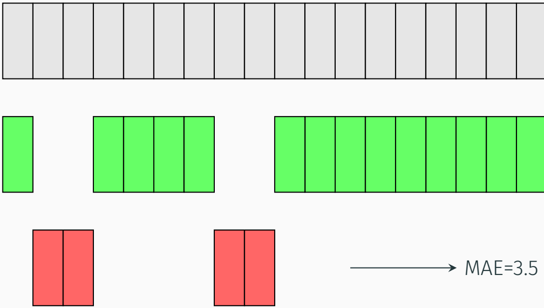
Assignment 2: Random seeds

In[1]:
```
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



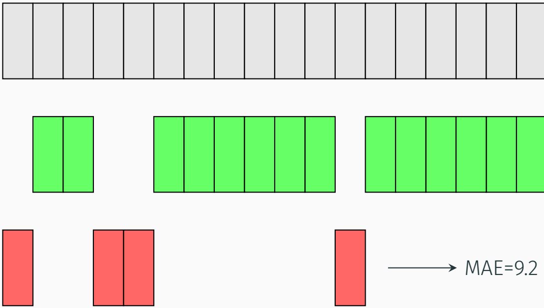
Assignment 2: Random seeds

In[1]:
```
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



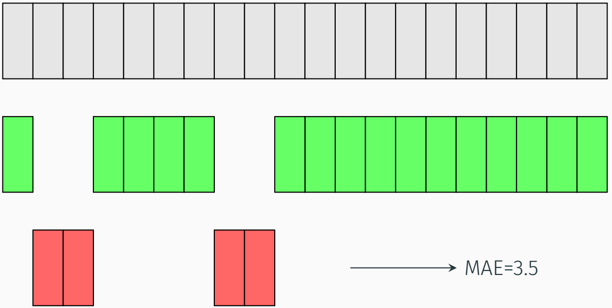
Assignment 2: Random seeds

In[1]:
```
import pandas as pd


df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



Assignment 2: Random seeds

In[1]:
```
import pandas as pd
import numpy as np
np.random.seed(42)
df = pd.DataFrame(...)
train = df.sample(frac=0.8)
test = df.drop(train.index)
```



Assignment 2: Log-odds vs probability vs class

```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

Assignment 2: Log-odds vs probability vs class

```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223
```

Assignment 2: Log-odds vs probability vs class

```r
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223
```

```r
predict(model, df, type="response")
```

```
     1      2      3      4      5      6      7      8      9     10     11
0.7766 0.8726 0.7314 0.7293 0.7405 0.9853 0.9844 0.9895 0.9233 0.8112 0.8352
```

Assignment 2: Log-odds vs probability vs class

In[1]:

```
model = LogisticRegression()
model.fit(df[['year', 'weight']], df['muscle'])
model.predict(df[['year', 'weight']])
```

Out[1]:

```
array([0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0])
```

In[1]:

```
model.predict_proba(df[['year', 'weight']])
```

Out[1]:

```
array([[0.14, 0.86], [0.08, 0.92], [0.17, 0.83], [0.18, 0.82]])
```

Assignment 2: Eye test

```
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223
```

Assignment 2: Eye test

```
model <- glm(muscle ~ year + weight, data=df, family='binomial')
predict(model, df)
```

```
     1      2      3      4      5      6      7      8      9     10     11
1.2460 1.9245 1.0019 0.9911 1.0485 4.2506 4.1465 4.5522 2.4889 1.4578 1.6223
```

```
model <- lm(mpg ~ horsepower, df)
summary(model)
```

```
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 19.59412 0.96187 20.371 < 2e-16 ***
horsepower102 0.40588 4.08087 0.099 0.920840
horsepower103 0.70588 4.08087 0.173 0.862789
horsepower105 0.90588 1.49529 0.606 0.545091
```

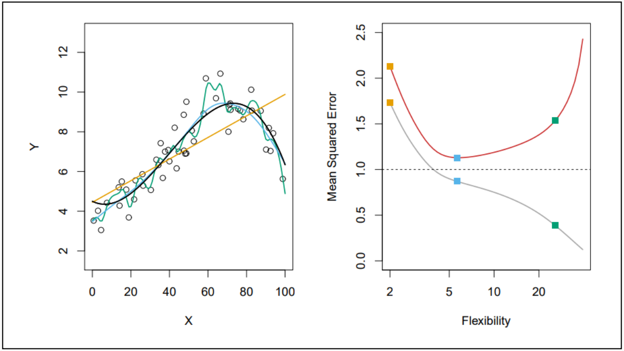
Regularization

Regularization: Preparations

$$y \sim \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$

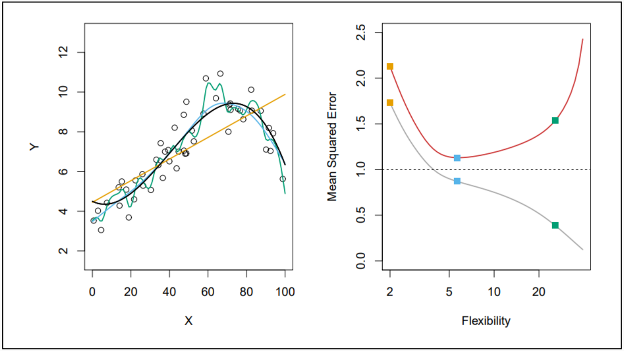
Regularization: Preparations

$$y \sim \beta_0 + \beta_1 * x_1 + \beta_2 * x_2$$



Regularization: Preparations

$$y \sim \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \beta_3 * x_3 + \beta_4 * x_4 + \beta_5 * x_5 + \beta_6 * x_6$$



Regularization: Preparations

In[1]:

```python
import pandas as pd


df = pd.read_csv('/Users/esten/Downloads/Auto.csv')
train = df.iloc[:int(len(df) * 0.8)]
validation = df.iloc[int(len(df) * 0.8):]

print(f'Using len(train) samples for training')
print(f'Using len(validation) samples for validation')
```

Out[1]:

```
Using 317 samples for training
Using 80 samples for validation
```

Regularization: Preparations

1. Variable selection
   a. Best subset selection
   b. Forward stepwise selection
   c. Backward stepwise selection
2. Shrinkage
   a. LASSO
   b. Ridge Regression
3. Dimensionality reduction: Lecture 6 and self-study

Regularization: Preparations

1. Variable selection
   a. Best subset selection
   b. Forward stepwise selection
   c. Backward stepwise selection
2. Shrinkage
   a. LASSO
   b. Ridge Regression
3. Dimensionality reduction: Lecture 6 and self-study

Variable selection

Variable selection

The number of predictors we are using in our model directly impacts model complexity.

Variable selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Variable selection

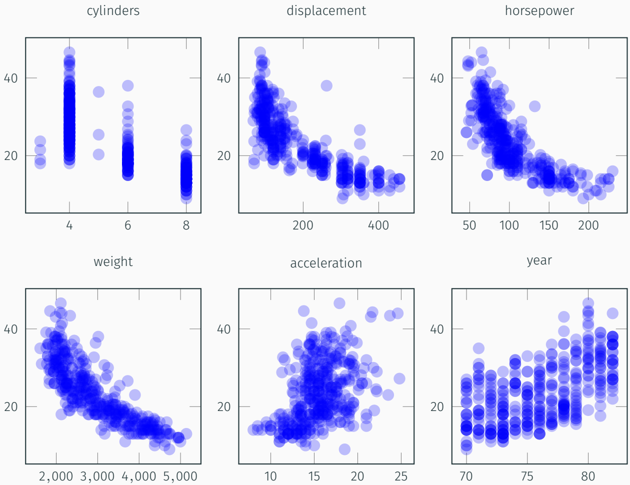
Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Motivation

To reduce model complexity (and therefore risk of overfitting), and to simplify subsequent interpretations.

Variable selection



Variable selection: Best subset selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Train models on all subsets p and select the best one.

Variable selection: Best subset selection

In[1]:

```
import numpy as np

from itertools import chain, combinations
from sklearn.linear_model import LinearRegression

subsets = list(chain.from_iterable(combinations(predictors, r)
for r in range(len(predictors)+1)))

best = 'mse': float('inf'), 'subset': None

for subset in subsets:
    if len(subset) == 0:
        continue

    model = LinearRegression()
    model.fit(train[list(subset)], train[target])
    predictions = model.predict(validation[list(subset)])
    mse = np.mean((predictions - validation[target]) ** 2)

    if mse < best['mse']:
        best = 'mse': mse, 'subset': subset

print(f'MSE: best["mse"]:.2f, predictors: best["subset"]')
```

Out[1]:

```
MSE: 29.68, predictors: ('cylinders', 'displacement', 'horsepower', 'weight', 'year')
```

Variable selection: Best subset selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Train models on all subsets p and select the best one.

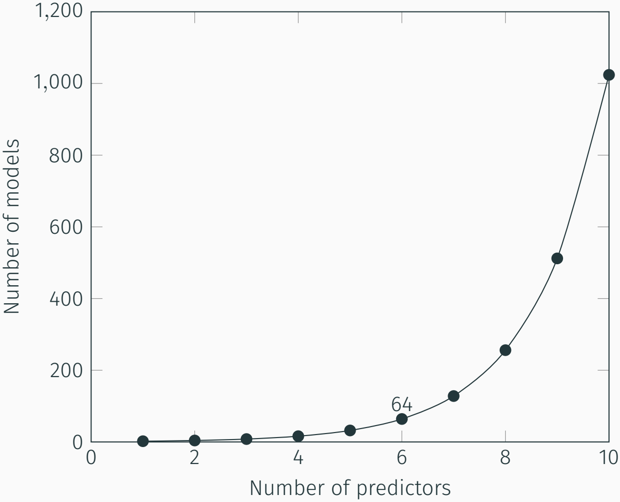
+ Positives

Guaranteed to find the optimal solution.
Simple implementation

- Drawbacks

Need to train many ($2^{|P|}$) models.

Variable selection: Best subset selection



Variable selection: Forward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

$$\boxed{\begin{array}{c} y \sim \mathbb{1} \\ mse = 146.47 \end{array}}$$

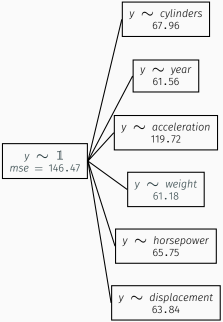
Variable selection: Forward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

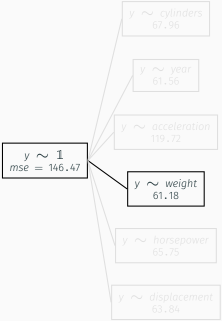


Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

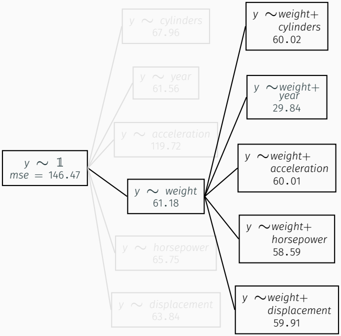


Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

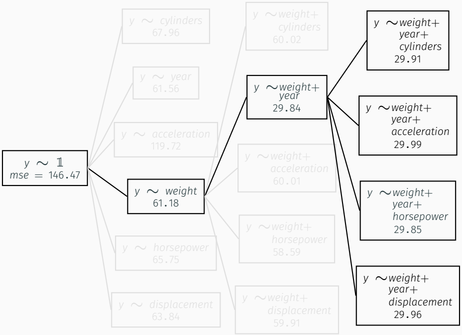


Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

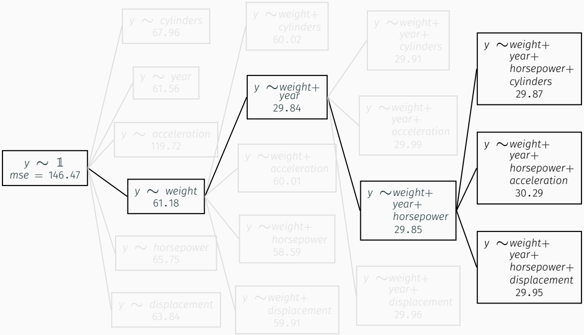


Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

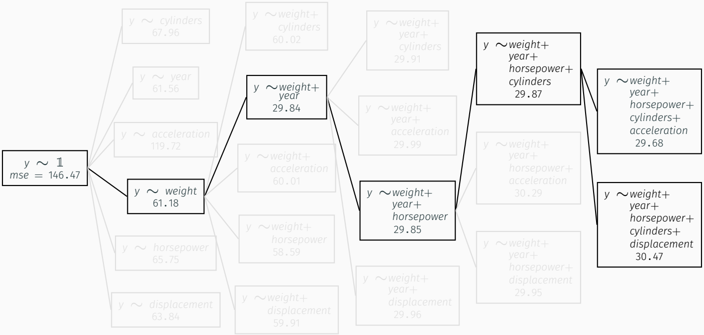


Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

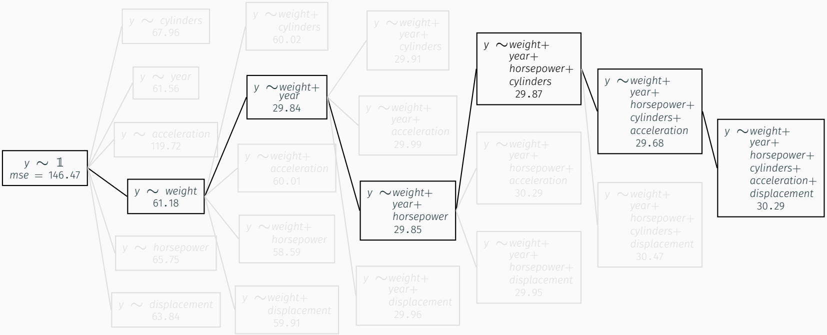


Problem

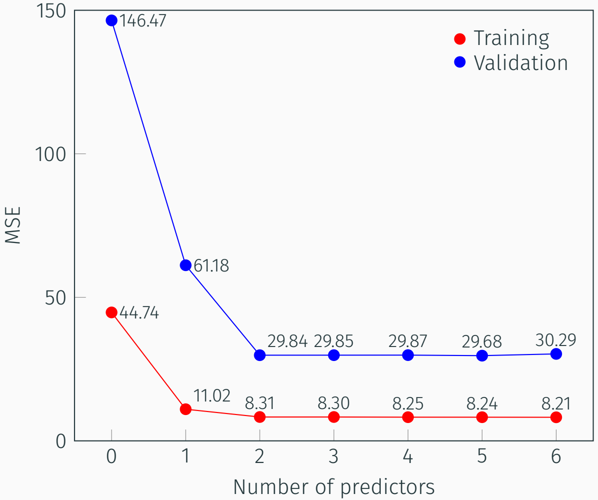
We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.



Variable selection: Forward stepwise selection



Variable selection: Forward stepwise selection

In[1]:

```
def fit_and_evaluate(train: pd.DataFrame, validation: pd.DataFrame,
                     predictors: List[str], target: str):
    model = LinearRegression()
    model.fit(train[predictors], train[target])

    train_predictions = model.predict(train[predictors])
    validation_predictions = model.predict(validation[predictors])

    return np.mean((train_predictions - train[target]) ** 2), \
           np.mean((validation_predictions - validation[target]) ** 2)


predictors = ['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year']
target = 'mpg'

train['intercept'] = 1
validation['intercept'] = 1
train_mse, validation_mse = fit_and_evaluate(train, validation, predictors=['intercept'], target=target)
print(f'[]: {validation_mse:.2f} ({train_mse:.2f})')

chosen_predictors = []

while len(chosen_predictors) < len(predictors):
    best_predictor = {'train_mse': None, 'validation_mse': float('inf'),
                      'predictor': None}

    for predictor in set(predictors) - set(chosen_predictors):
        train_mse, validation_mse = fit_and_evaluate(train, validation, predictors=chosen_predictors + [predictor], target=target)

        if validation_mse < best_predictor['validation_mse']:
            best_predictor = {'train_mse': train_mse, 'validation_mse': validation_mse, 'predictor': predictor}

    chosen_predictors.append(best_predictor['predictor'])

    print(f'{chosen_predictors}: {best_predictor["validation_mse"]:.2f} ({best_predictor["train_mse"]:.2f})')
```

Variable selection: Forward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with no predictors. Iteratively add the predictor that yields the best model until all are included.

+ Positives

Need to train fewer models.

- Drawbacks

Not guaranteed to find the optimal solution.

Variable selection: Backward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with all predictors. Iteratively remove the predictor that yields the best model until all you have none left.

Variable selection: Backward stepwise selection

Problem

We have a set of predictors $P = \{x_0, x_1, ...\}$ and a target variable y, and we want to find the subset $p \subseteq P$ that yields the best (linear) model for predicting y.

Solution

Start with all predictors. Iteratively remove the predictor that yields the best model until all you have none left.

+ Positives

Need to train fewer models.

- Drawbacks

Not guaranteed to find the optimal solution.

Shrinkage

Shrinkage: Outline

$$y \sim \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_5 + \beta_6 x_6$$