4dt905_sb224sc_A5

January 26, 2026

1 Assignment 5

1.1 The Bootstrap

Author: Samuel Fredric Berg

Student ID: sb224sc

Date: 2026-01-26

Course: Machine Learning 4DT905

1.1.1 Conceptual

1. Shuffle the given dataset to ensure randomness. Partition the dataset into k equal-sized groups (folds). For fold k, take the fold as *test* dataset and the remaining $k-1$ folds as *training* dataset. Fir a model on the *training* dataset and evaluate it with the *test* dataset. Retain the resulting metrics and discard model. Compute the average of the retained metrics to estimate the models ture preformance.

2. *i.* **Validation set approach** is a single split of the dataset into 70% taining and 30% testing. This results in a higher variance, lower data efficiency and a lower computational cost then the k-fold cross validation.

3. *ii.* **Leave-one-out cross validation** (LOOCV) is a special case of k-fold cross validation where $k = n$ where n is the number of observations. This results in a higher computational cost, lower bias and higher variance then the k-fold cross validation.

1.1.2 Practical

Imports

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import statsmodels.api as sm
import numpy as np
```

Load data

[2]:
```
df = pd.read_csv("../data/Auto.csv", index_col=0)
```

Number of features and names

[3]:
```
df_names = df.columns.tolist()
print(f"Number of columns: {len(df_names)}")
print(f"Column names: {df_names}")
```

```
Number of columns: 9
Column names: ['mpg', 'cylinders', 'displacement', 'horsepower', 'weight',
'acceleration', 'year', 'origin', 'name']
```

Statistical summary of features

[5]:
```
print(df.describe())
print(df["name"].value_counts())
```

```
              mpg   cylinders  displacement  horsepower       weight  \
count  392.000000  392.000000    392.000000  392.000000   392.000000
mean    23.445918    5.471939    194.411990  104.469388  2977.584184
std      7.805007    1.705783    104.644004   38.491160   849.402560
min      9.000000    3.000000     68.000000   46.000000  1613.000000
25%     17.000000    4.000000    105.000000   75.000000  2225.250000
50%     22.750000    4.000000    151.000000   93.500000  2803.500000
75%     29.000000    8.000000    275.750000  126.000000  3614.750000
max     46.600000    8.000000    455.000000  230.000000  5140.000000

       acceleration        year      origin
count    392.000000  392.000000  392.000000
mean      15.541327   75.979592    1.576531
std        2.758864    3.683737    0.805518
min        8.000000   70.000000    1.000000
25%       13.775000   73.000000    1.000000
50%       15.500000   76.000000    1.000000
75%       17.025000   79.000000    2.000000
max       24.800000   82.000000    3.000000
name
amc matador         5
ford pinto          5
toyota corolla      5
chevrolet impala    4
amc hornet          4
                   ..
ford mustang gl     1
vw pickup           1
dodge rampage       1
ford ranger         1
chevy s-10          1
Name: count, Length: 301, dtype: int64
```

Number of datapoints

[6]:
```
print(f"Number of datapoints: {len(df)}")
```

```
Number of datapoints: 392
```

Display data in table format

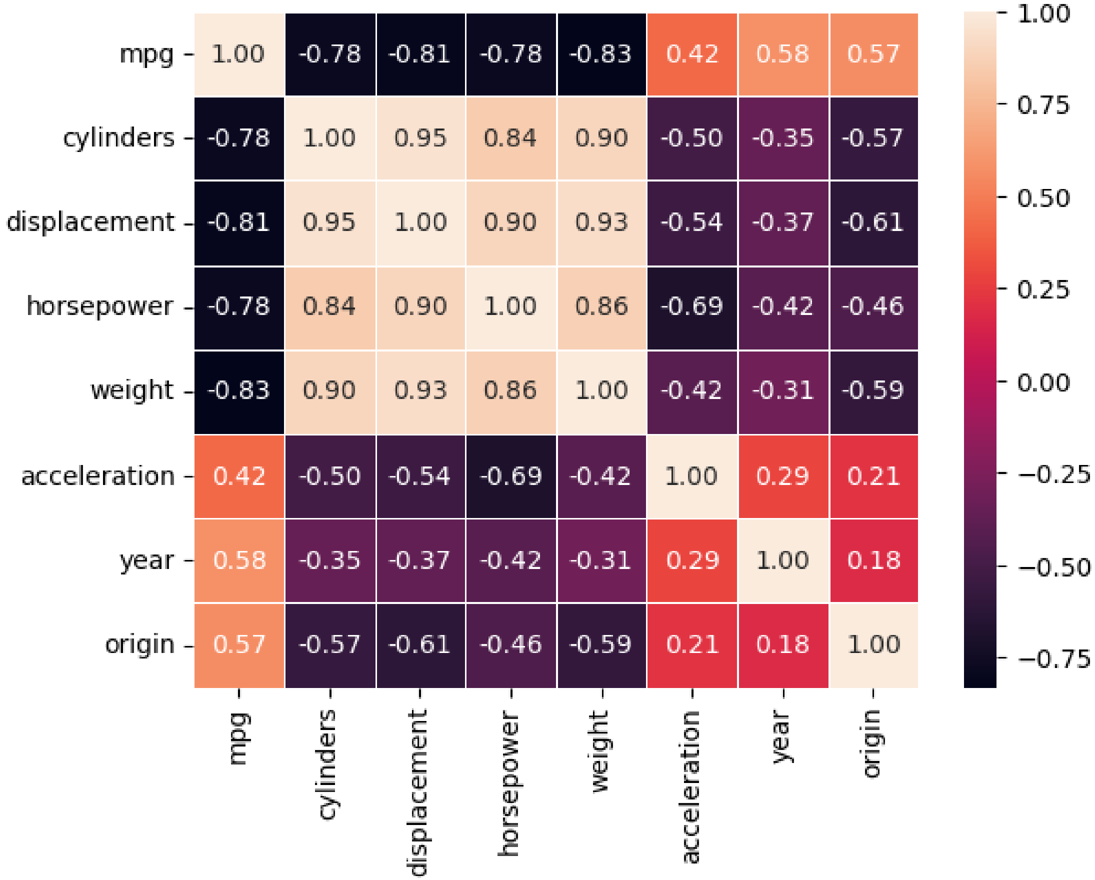
```
[7]: print(df.head(5))
```

```
    mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
1  18.0          8         307.0         130    3504          12.0    70
2  15.0          8         350.0         165    3693          11.5    70
3  18.0          8         318.0         150    3436          11.0    70
4  16.0          8         304.0         150    3433          12.0    70
5  17.0          8         302.0         140    3449          10.5    70

   origin                       name
1       1  chevrolet chevelle malibu
2       1          buick skylark 320
3       1         plymouth satellite
4       1              amc rebel sst
5       1                ford torino
```

Correlation matrix

```
[9]: sns.heatmap(df.drop(columns=["name"]).corr(), annot=True, fmt=".2f",␣
       ↪linewidths=0.5)
     plt.show()
```



Accuracy Estimation Function

[ ]:
```
def boot_fn(data, index):
    sample = data.iloc[index]
    X = sample["horsepower"]
    Y = sample["mpg"]
    X = sm.add_constant(X)
    model = sm.OLS(Y, X).fit()

    return model.params


print(boot_fn(df, range(392)))
```

```
const         39.935861
horsepower    -0.157845
dtype: float64
```

[14]:
```
np.random.seed(42)
```

```
print(boot_fn(df, np.random.choice(392, 392, replace=True)))
```

```
const         40.466879
horsepower    -0.163738
dtype: float64
```

[ ]:

```
boot_results = np.zeros((1000, 2))
for idx in range(1000):
    indices = np.random.choice(392, 392, replace=True)
    boot_results[idx, :] = boot_fn(df, indices)

print(f"Standard errors: {boot_results.std(axis=0)}")
```

```
Standard errors: [0.86861119 0.00749939]
```

[17]:

```
X = df["horsepower"]
Y = df["mpg"]
X = sm.add_constant(X)
model = sm.OLS(Y, X).fit()
print(model.params)
print(model.summary())
```

```
const         39.935861
horsepower    -0.157845
dtype: float64
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.606
Model:                            OLS   Adj. R-squared:                  0.605
Method:                 Least Squares   F-statistic:                     599.7
Date:                Mon, 26 Jan 2026   Prob (F-statistic):           7.03e-81
Time:                        10:17:19   Log-Likelihood:                -1178.7
No. Observations:                 392   AIC:                             2361.
Df Residuals:                     390   BIC:                             2369.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         39.9359      0.717     55.660      0.000      38.525      41.347
horsepower    -0.1578      0.006    -24.489      0.000      -0.171      -0.145
==============================================================================
Omnibus:                       16.432   Durbin-Watson:                   0.920
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               17.305
Skew:                           0.492   Prob(JB):                     0.000175
Kurtosis:                       3.299   Cond. No.                         322.
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```

Interpretation The standard error from the model summary is a smaller span compared to the bootstrap estimate. This is expected as the bootstrap method since it is a estimate over several subsets of the data. Meaning currently with the values given the regular estimate is better but most likely it is a bit overfitted.

[18]:
```python
def boot_fn_quadratic(data, index):
    sample = data.iloc[index]
    X = sample["horsepower"]
    Y = sample["mpg"]
    X = np.column_stack((X, X**2))
    X = sm.add_constant(X)
    model = sm.OLS(Y, X).fit()

    return model.params
```

[20]:
```python
boot_results = np.zeros((1000, 3))
for idx in range(1000):
    indices = np.random.choice(392, 392, replace=True)
    boot_results[idx, :] = boot_fn_quadratic(df, indices)

print(f"Standard errors: {boot_results.std(axis=0)}")
```

```
Standard errors: [2.14866982e+00 3.42412395e-02 1.23413967e-04]
```

[21]:
```python
df["horsepower_squared"] = df["horsepower"] ** 2
X = df[["horsepower", "horsepower_squared"]]
X = sm.add_constant(X)
model = sm.OLS(Y, X).fit()

print(model.params)
print(model.summary())
```

```
const                 56.900100
horsepower            -0.466190
horsepower_squared     0.001231
dtype: float64
                            OLS Regression Results
==============================================================================
Dep. Variable:                    mpg   R-squared:                       0.688
Model:                            OLS   Adj. R-squared:                  0.686
Method:                 Least Squares   F-statistic:                     428.0
Date:                Mon, 26 Jan 2026   Prob (F-statistic):           5.40e-99
Time:                        10:26:43   Log-Likelihood:                -1133.2
No. Observations:                 392   AIC:                             2272.
Df Residuals:                     389   BIC:                             2284.
Df Model:                           2
```

```
Covariance Type:            nonrobust
================================================================================
======
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
------
const                56.9001      1.800     31.604      0.000      53.360
60.440
horsepower           -0.4662      0.031    -14.978      0.000      -0.527
-0.405
horsepower_squared    0.0012      0.000     10.080      0.000       0.001
0.001
==============================================================================
Omnibus:                       16.158   Durbin-Watson:                   1.078
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               30.662
Skew:                           0.218   Prob(JB):                     2.20e-07
Kurtosis:                       4.299   Cond. No.                     1.29e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.29e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Interpretation Again the standard error from the model summary is a smaller span compared to the bootstrap estimate. This is due to the same reasoon as in the linear case.