COMPSCI 389
Introduction to Machine Learning

Days: Tu/Th. **Time:** 2:30 – 3:45 **Building:** Morrill 2 **Room:** 222

Topic 5.3: Evaluation Part 3

Prof. Philip S. Thomas (pthomas@cs.umass.edu)

Overview

- In this lecture we will conduct a series of experiments to highlight:
  - The need for confidence intervals
  - How confidence intervals can be computed
  - The different types of intervals that are often reported

- The code covered in this lecture will be provided in:

  `5.3 Evaluation Part 3.ipynb`

Load data, split train/test, compute predictions and errors

```python
# Load the data set
df = pd.read_csv("data/GPA.csv", delimiter=',')

# We already loaded X and y, but do it again as a reminder
X = df.iloc[:, :-1]
y = df.iloc[:, -1]

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.8, shuffle=True)

# Train the NearestNeighbor model
model = KNearestNeighbors(k=1)
model.fit(X_train, y_train)

# Compute predictions for X_test
predictions = model.predict(X_test)

# Compute the sample squared errors
squared_errors = (predictions - y_test) ** 2
display("Squared errors: ", squared_errors)
```

The algorithm used won't be important for this discussion.

Let's use most of the data for evaluation for now.

We compute the squared error for all the testing points (80% of the data). We will "simulate" using less data by looking at portions of these errors.

squared_errors

```
16809    2.170701
31412    0.030047
27099    0.277381
41341    0.440007
27941    0.523206
          ...
2874     0.006944
21169    0.017777
39080    0.572534
24225    3.635379
22171    0.005377
Name: gpa, Length: 34643, dtype: float64
```
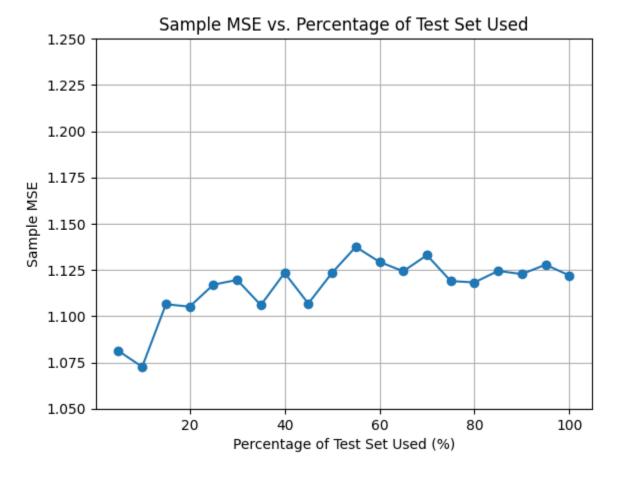
Idea

- We would like to see what happens if we have test sets of different sizes.

- We would like to see what happens if we have different samples in the test set.

- Ideally, we would generate completely new test sets (and squared errors) for each experiment.
  - We only have 43,303 points total and cannot generate more.

- Instead, we use random subsets of the test set (80% of the data)
  - We vary the size of these subsets.
  - We pretend that random subsets are independent (they are not)
    - Note: This will be important at the end!

Compute the sample MSE from `percentage`% of the squared errors.

```python
# Function to compute average MSE for a given percentage of squared errors
def compute_average_mse(percentage, squared_errors):
    # Get the number of samples that we should use from squared_errors
    subset_size = int(percentage * len(squared_errors))

    # Randomly select that many indices (without replacement)
    indices = np.random.choice(len(squared_errors), subset_size, replace=False)
```
Randomly select `percentage`% of the points, so each call produces results from a different simulated sample
```python
    # Get the average of the squared errors at the selected indices
    average_mse = squared_errors.iloc[indices].mean()
    return average_mse
```

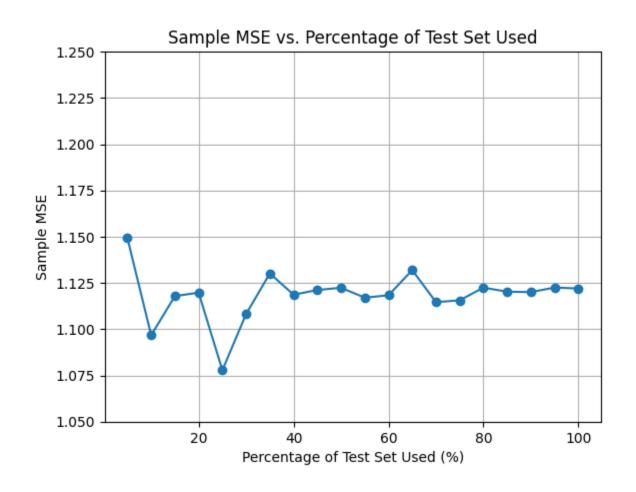# Plot: Sample MSE using different amounts of the test data.

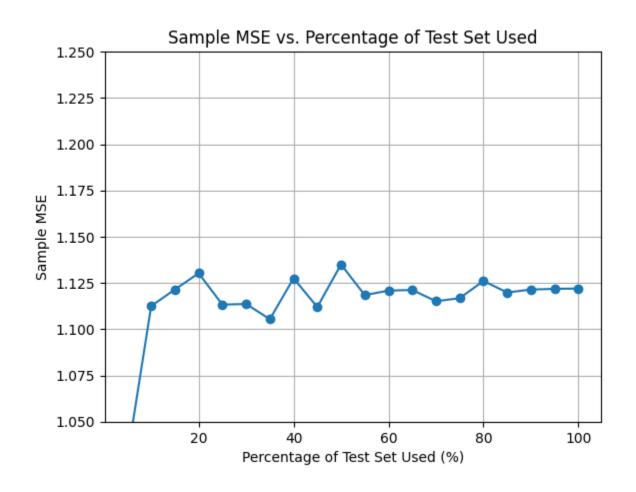**Note**: The amount of **training** data is not varied.

This represents the setting where a fixed model is given, and we are trying to evaluate that model, regardless of how it was created.
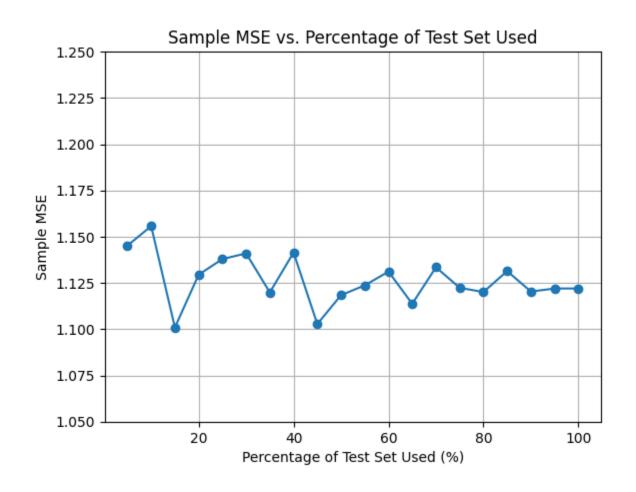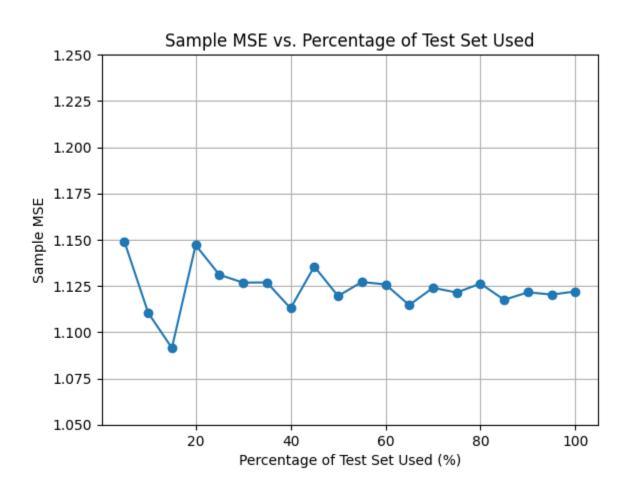

Note: In practice, we would only have *one point* from this plot: the sample MSE using our one testing set. This plot is meant to give further insight about how much we can (not) trust that value.

Simulates the sample MSE we would get with different amounts of testing data

Plot: Sample MSE using different amounts of the test data.

Sample MSE vs. Percentage of Test Set Used

Plot: Sample MSE using different amounts of the test data.

Plot: Sample MSE using different amounts of the test data.

Plot: Sample MSE using different amounts of the test data.

Notice: There was significant variation across runs!

Overlay of 100 runs

Sample MSE vs. Percentage of Test Set Used

Each entry in the tables from before showed *one single point* from this plot.

	Model	MSE	RMSE	MAE	R^2
0	k-NN k=1 sigma=None	1.152084	1.073352	0.823743	-0.687769
1	k-NN k=100 sigma=None	0.579404	0.761186	0.596919	0.151190
2	k-NN k=100 sigma=100	0.579572	0.761297	0.596952	0.150943
3	k-NN k=200 sigma=100	0.577554	0.759970	**0.596220**	0.153901
4	k-NN k=300 sigma=100	**0.577443**	**0.759897**	0.596408	**0.154062**
5	k-NN k=400 sigma=100	0.577620	0.760013	0.596670	0.153804
6	k-NN k=500 sigma=100	0.578077	0.760314	0.597044	0.153135

Note: It isn't precisely the indicated point, since this old table was for a model trained with more data.
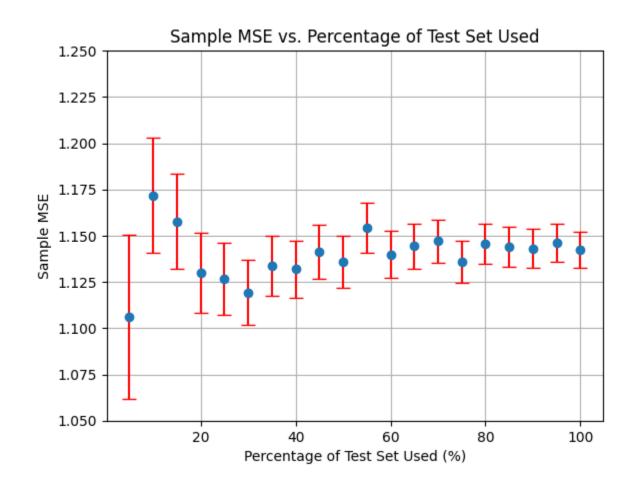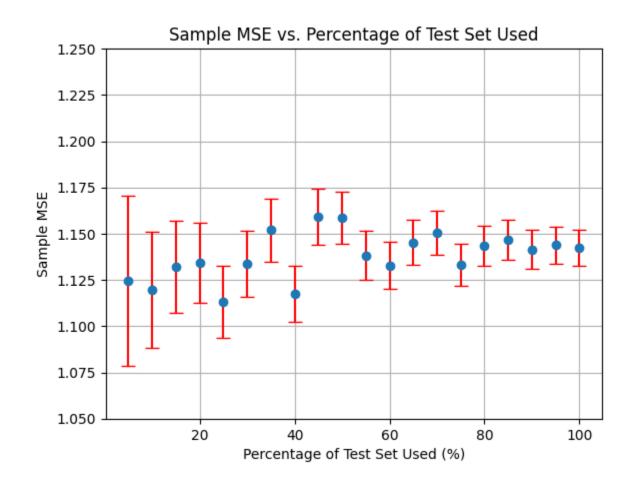
Variance

- The previous plots show the *variance* of the sample statistics (sample metrics).

- When we compute sample statistics from data, they are not exactly equal to the population statistic (parameter).

- If we had many data sets and computed the sample statistics many times, we would see this variance.

- With only one data set, we get our estimate (e.g., the sample MSE)

- How much should we trust this estimate?
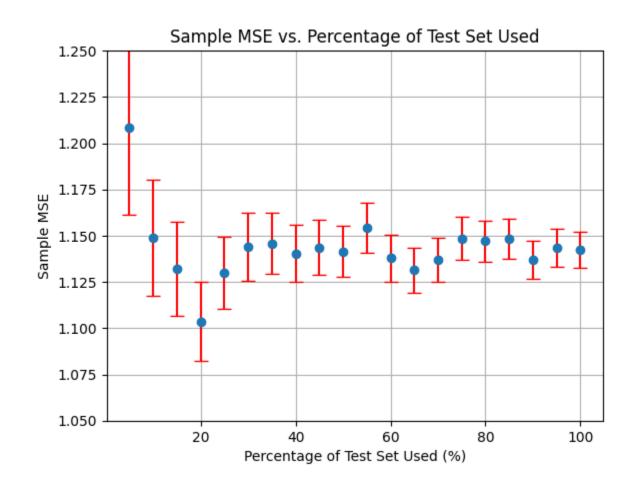 - We can use *standard error* to get an idea.

Error Bars: Standard Error

- We can plot error bars on each point

- The width of each error bar is the *standard error*
  - The point is the average of the `squared_error` samples
  - The error bar width is the standard error of the `squared_error` samples
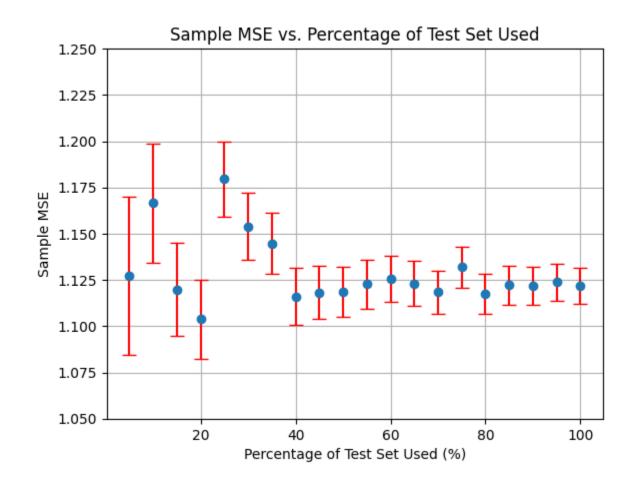 - Recall:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{n-1}}, \qquad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \qquad \mathrm{SE} = \frac{\sigma}{\sqrt{n}}$$

  - Here, x_i is one of the `squared_error` values.
  - The standard deviation σ captures how much the squared errors vary.
  - $\pm 1.6 \times$ SE gives a 95% confidence interval (under normality assumptions)

Plot: Sample MSE using different amounts of the test data, with error bars (standard error).

Plot: Sample MSE using different amounts of the test data, with error bars (standard error).

Plot: Sample MSE using different amounts of the test data, with error bars (standard error).

Plot: Sample MSE using different amounts of the test data, with error bars (standard error).
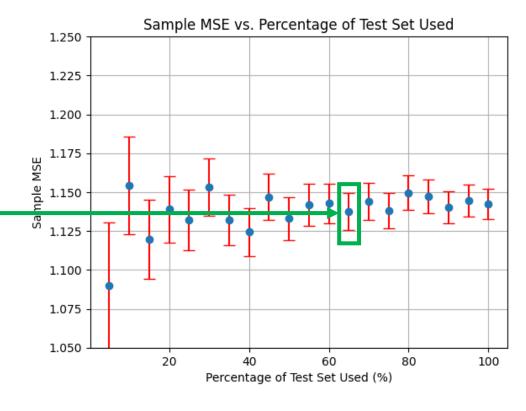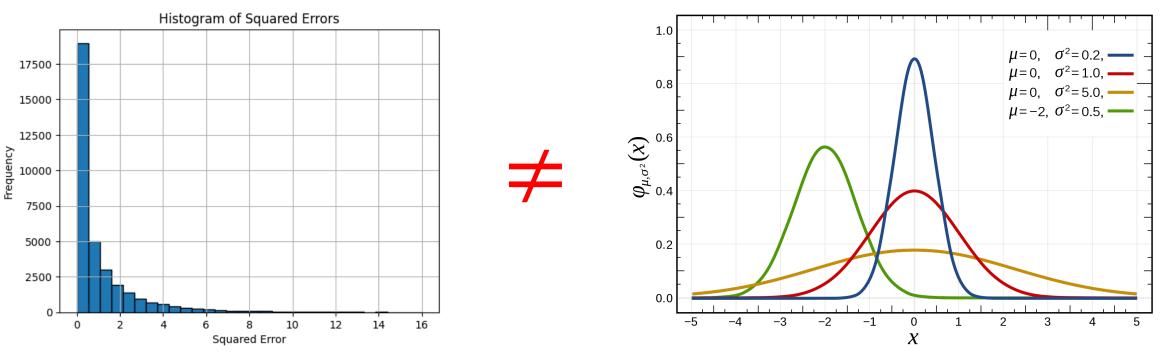
Error Bars: Standard Error

- Notice that given a single sample MSE (one point on this plot) we can compute the standard error.

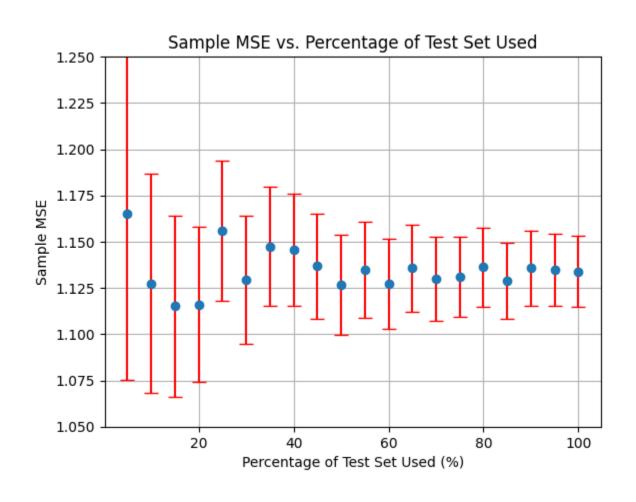- If we computed the sample MSE for our test set and the standard error and observed this

  can we conclude that the actual MSE is within this interval with 95% confidence?

Sample MSE vs. Percentage of Test Set Used
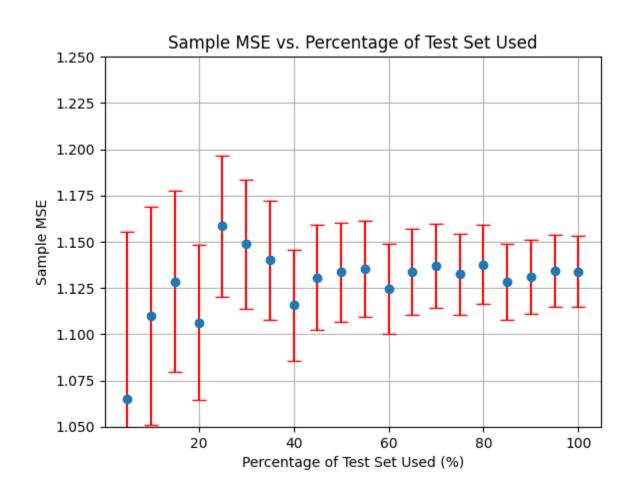
Error Bars: Standard Error

- Yes, if
  - The squared errors are normally distributed
  - The error bars were inflated by a factor of 1.96

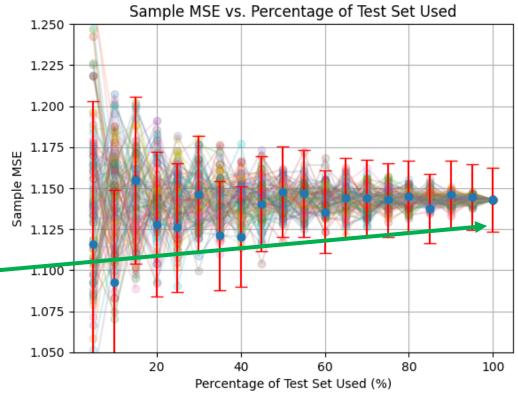(the squared errors are far from normally distributed)

Error Bars: Standard Error x 1.96

Sample MSE vs. Percentage of Test Set Used

Error Bars: Standard Error x 1.96

Sample MSE vs. Percentage of Test Set Used

Error Bars: Standard Error x 1.96

- Remember, each error bar has a 5% error chance.

- With 20 points and error bars, we expect *roughly* one (on average) to not contain the true MSE.

(If data was normally distributed)

Error Bars: Standard Error x 1.96

- Although not entirely reliable (due not lack of normality and evaluating many estimates), error bars give an *idea* of how much results can be trusted.

- **Note**: ML texts/papers almost never inflate standard-error error-bars by a factor of 1.96.

- Note: The blue points and error bars can be computed from a single run

- **Question**: Why are the error bars still so wide when using 100% of the testing data?

Answer

- Recall that the squared errors are computed from a portion of a test set.

- At 100% of the test set, the plotted MSE is always the MSE on the full test set.

- This test set is still finite, and the sample MSE is still an approximation of the true MSE!

- The red error bars show this uncertainty.

Sample MSE vs. Percentage of Test Set Used