# 65020

# Open Source Tools for Intelligent Systems

Based on lectures of Dr. Jonathan Alon

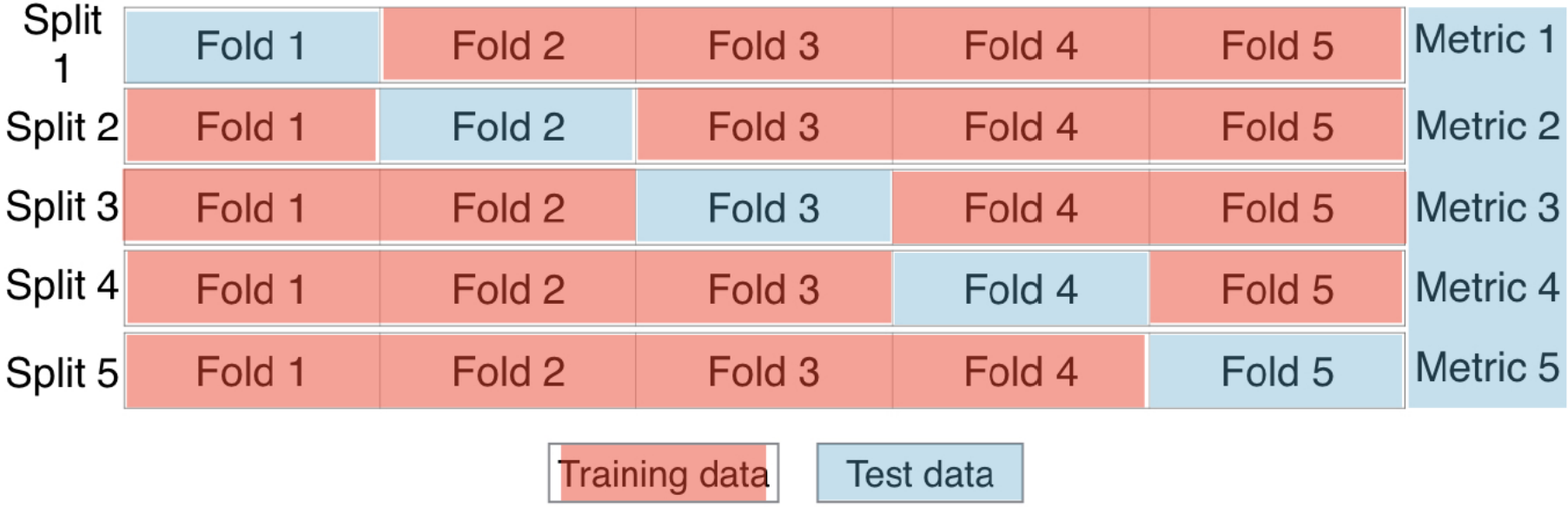
## Python Packages

- IPython and Jupyter – provide the computational environment in which many Python-using data scientists work.
- NumPy – provides the ndarray object for efficient storage and manipulation of dense data arrays.
- pandas – provides the DataFrame object for efficient storage and manipulation of labeled/columnar data.
- matplotlib – provides capabilities for a flexible range of data visualizations.
- seaborn – provides an API on top of Matplotlib that defines simple high-level functions for common statistical plot types, and integrates with the functionality provided by pandas DataFrames.
- SciPy – provides a collection of packages for scientific computing.
- scikit-learn – provides efficient and clean implementations of the most important and established machine learning algorithms.

## Cross-Validation Motivation

- Model performance is dependent on way the data is split
- Not representative of the model's ability to generalize
- Solution: Cross-validation!

## Cross-Validation



## Cross-Validation and Model Performance

- 5 folds = 5-fold CV
- 10 folds = 10-fold CV
- k folds = k-fold CV
- More folds = More computationally expensive

## Cross-Validation in scikit-learn

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
cv_results = cross_val_score(reg, X, y, cv=5)
print(cv_results)
```

```
[ 0.63919994  0.71386698  0.58702344  0.07923081 -0.25294154]
```

```
np.mean(cv_results)
```

```
0.3532759243958705 8
```

## Why consider alternatives to least squares?

- *Prediction Accuracy*: especially when p > n (#predictors > #observations), to control the variance.
- *Model Interpretability*: By removing irrelevant features that is, by setting the corresponding coefficient estimates to zero we can obtain a model that is more easily interpreted. We will present some approaches for automatically performing feature selection.

## Three Classes of Methods

- *Subset Selection*. We identify a subset of the p predictors that we believe to be related to the response. We then fit a model using least squares on the reduced set of variables.
- *Shrinkage*. We fit a model involving all p predictors, but the estimated coecients are shrunken towards zero relative to the least squares estimates. This shrinkage (also known as regularization) has the effect of reducing variance and can also perform variable selection.
- *Dimension Reduction*. We project the p predictors into a M-dimensional subspace, where M < p. This is achieved by computing M dierent linear combinations, or projections, of the variables. Then these M projections are used as predictors to t a linear regression model by least squares.

## Shrinkage Methods

*Ridge regression* and *Lasso*

- The subset selection methods use least squares to fit a linear model that contains a subset of the predictors.
- As an alternative, we can fit a model containing all p predictors using a technique that *constrains* or *regularizes* the coefficient estimates, or equivalently, that *shrinks* the coefficient estimates towards zero.
- It may not be immediately obvious why such a constraint should improve the fit, but it turns out that shrinking the coefficient estimates can significantly reduce their variance.

## Regularized Regression – Why Regularize?

- Recall: Linear regression minimizes a loss function
- It chooses a coefficient for each feature variable
- Large coefficients can lead to overfitting
- Penalizing large coefficients: Regularization

## Ridge Regression

- Loss function = OLS loss function

$$\sum_{i=1}^{n}\left(y_i - \beta_o - \sum_{j=1}^{p}\beta_j\, x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = \text{RSS} + \lambda\sum_{j=1}^{p}\beta_j^2$$

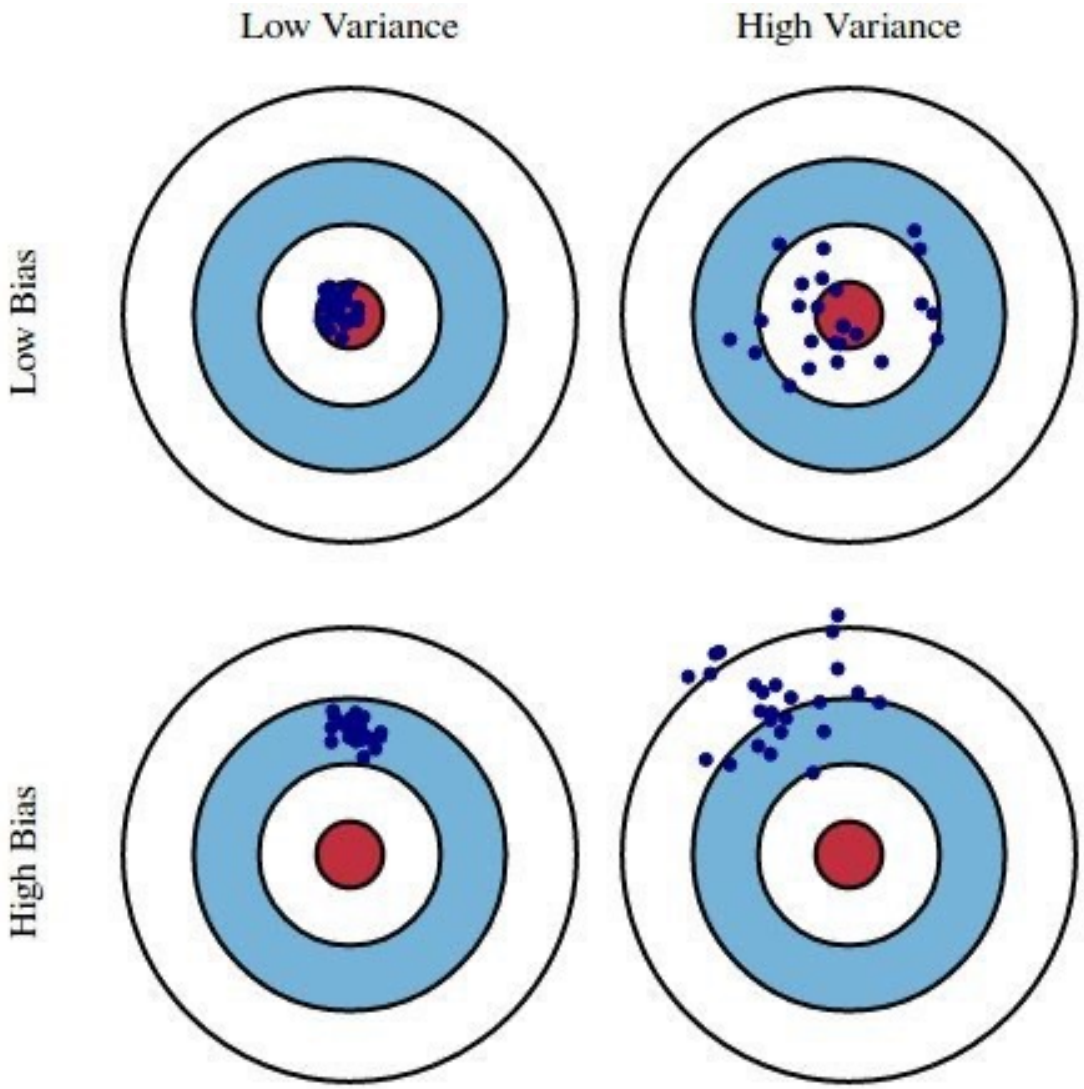
- Alpha: Parameter we need to choose
- Picking alpha here is similar to picking k in k-NN (next lesson)
- Hyperparameter tuning (later in course)
- Alpha controls model complexity
  - Alpha = 0: We get back OLS (Can lead to overfitting)
  - Very high alpha: Can lead to underfitting

## Ridge Regression in scikit-learn

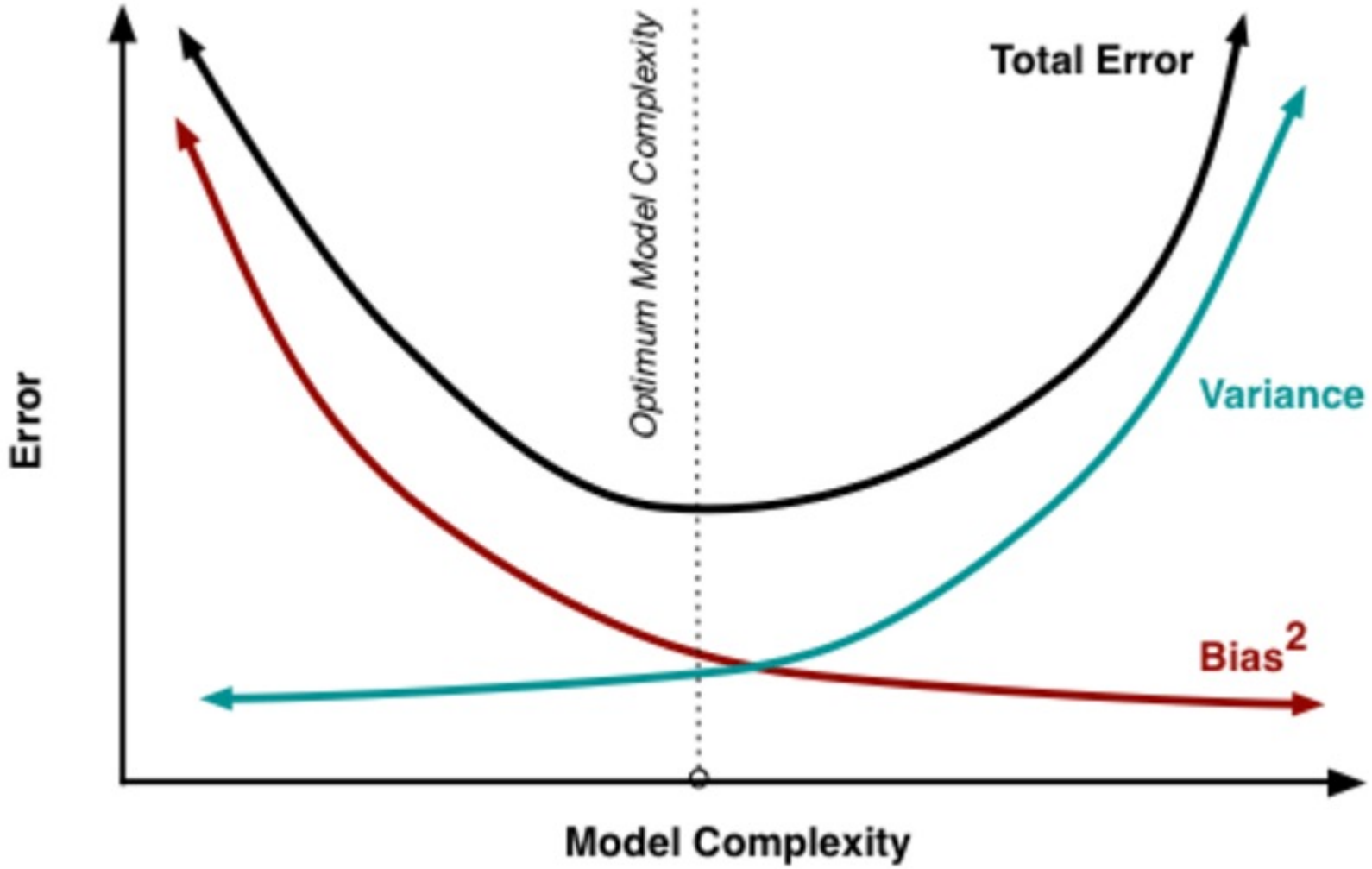
```
from sklearn.linear_model import Ridge
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.3, random_state=42)
ridge = Ridge(alpha=0.1, normalize=True)
ridge.fit(X_train, y_train)
ridge_pred = ridge.predict(X_test)
ridge.score(X_test, y_test)
```

```
0.6996938275127317 9
```

## Bias and Variance



## Bias-Variance Tradeoff



## Lasso

- Ridge regression does have one obvious disadvantage: unlike subset selection, which will generally select models that involve just a subset of the variables, ridge regression will include all p predictors in the final model
- The Lasso is a relatively recent alternative to ridge regression that overcomes this disadvantage. The lasso coefficients, minimize the quantity
- Loss function = $$\sum_{i=1}^{n}\left(y_i - \beta_o - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2 + \lambda\sum_{j=1}^{p}|\beta_j| = \text{RSS} + \lambda\sum_{j=1}^{p}|\beta_j|$$
- In statistical parlance, the lasso uses an L1 penalty instead of an L2 penalty.

## Lasso

- As with ridge regression, the lasso shrinks the coefficient estimates towards zero.
- However, in the case of the lasso, the L1 penalty has the effect of forcing some of the coefficient estimates to be exactly equal to zero when the tuning parameter is sufficiently large.
- Hence, much like best subset selection, the lasso performs *variable selection*.
- We say that the lasso yields *sparse* models – that is, models that involve only a subset of the variables.
- As in ridge regression, selecting a good value of  for the lasso is critical; cross-validation is again the method of choice.

## Lasso Regression in scikit-learn

```
from sklearn.linear_model import Lasso
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size = 0.3, random_state=42)
lasso = Lasso(alpha=0.1, normalize=True)
lasso.fit(X_train, y_train)
lasso_pred = lasso.predict(X_test)
lasso.score(X_test, y_test)
```

```
0.59502295353285506
```

## Lasso Regression for Feature Selection

- Can be used to select important features of a dataset
- Shrinks the coefficients of less important features to exactly 0

## Lasso for feature selection in scikit-learn

```
from sklearn.linear_model import Lasso

names = boston.drop('MEDV', axis=1).columns

lasso = Lasso(alpha=0.1)

lasso_coef = lasso.fit(X, y).coef_

_ = plt.plot(range(len(names)), lasso_coef)
_ = plt.xticks(range(len(names)), names, rotation=60)
_ = plt.ylabel('Coefficients')
plt.show()
```

## Lasso for feature selection in scikit-learn