02/18/19 Yasin Ceran

Types of Machine Learning

- Supervised
- Unsupervised
- Reinforcement

Supervised Learning

$$(x_i, y_i) \propto p(x, y) \text{ i.i.d.}$$

$$x_i \in \mathbb{R}^p$$

$$y_i \in \mathbb{R}$$

$$f(x_i) \approx y_i$$

Classification and Regression

- target y discrete
- Will you pass?

- target y continuous
- How many points will you get in the exam?

Generalization

Not only $f(x_i) \approx y_i$,

also for new data: $f(x) \approx y$

Relationship to Statistics

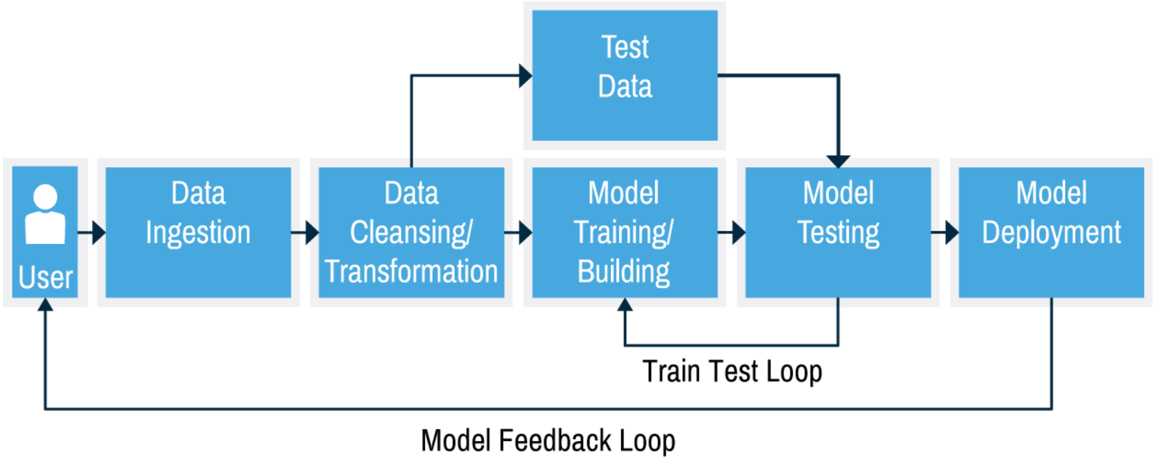
Statistics

- model first
- inference emphasis

Machine learning

- data first
- prediction emphasis

The Machine Learning Work-Flow



Representing Data

$$
\begin{array}{r c}
\text{one sample} & \\
&
\end{array}
X = \begin{pmatrix}
1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\
6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\
2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\
1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\
0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\
5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\
3.7 & 7.8 & 2.6 & 3.2 & 6.3
\end{pmatrix}
\qquad
y = \begin{pmatrix}
1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7
\end{pmatrix}
$$

Training and Test Data

training set

$$X = \begin{pmatrix} 1.1 & 2.2 & 3.4 & 5.6 & 1.0 \\ 6.7 & 0.5 & 0.4 & 2.6 & 1.6 \\ 2.4 & 9.3 & 7.3 & 6.4 & 2.8 \\ 1.5 & 0.0 & 4.3 & 8.3 & 3.4 \\ 0.5 & 3.5 & 8.1 & 3.6 & 4.6 \\ 5.1 & 9.7 & 3.5 & 7.9 & 5.1 \\ 3.7 & 7.8 & 2.6 & 3.2 & 6.3 \end{pmatrix} \qquad y = \begin{pmatrix} 1.6 \\ 2.7 \\ 4.4 \\ 0.5 \\ 0.2 \\ 5.6 \\ 6.7 \end{pmatrix}$$

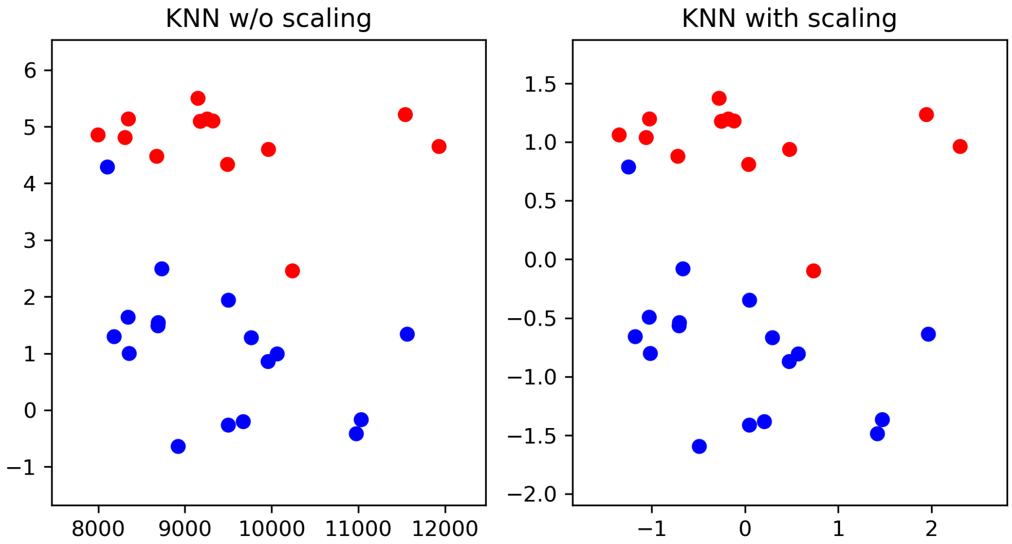
test set

Jupyter Notebook

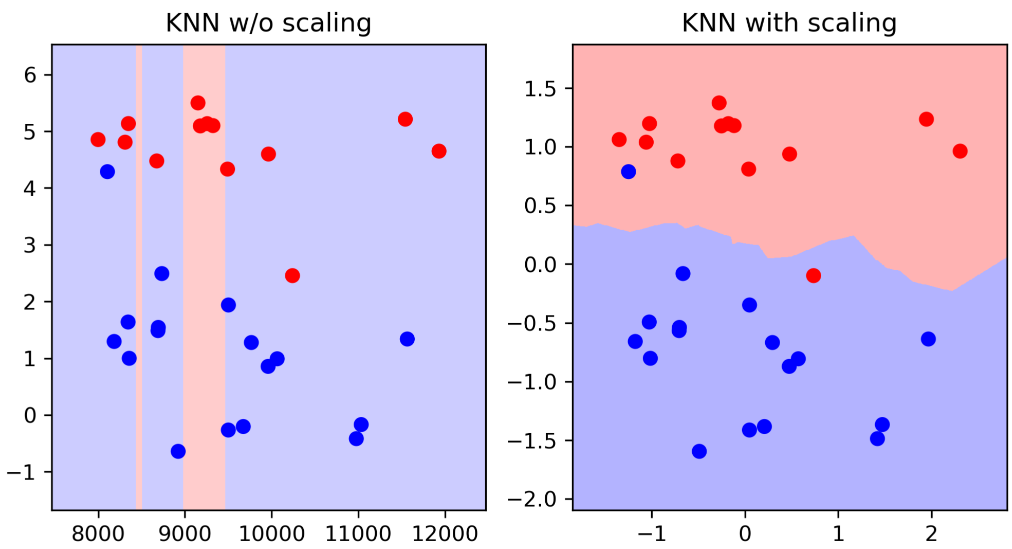
Part 1- Data Loading

Preprocessing

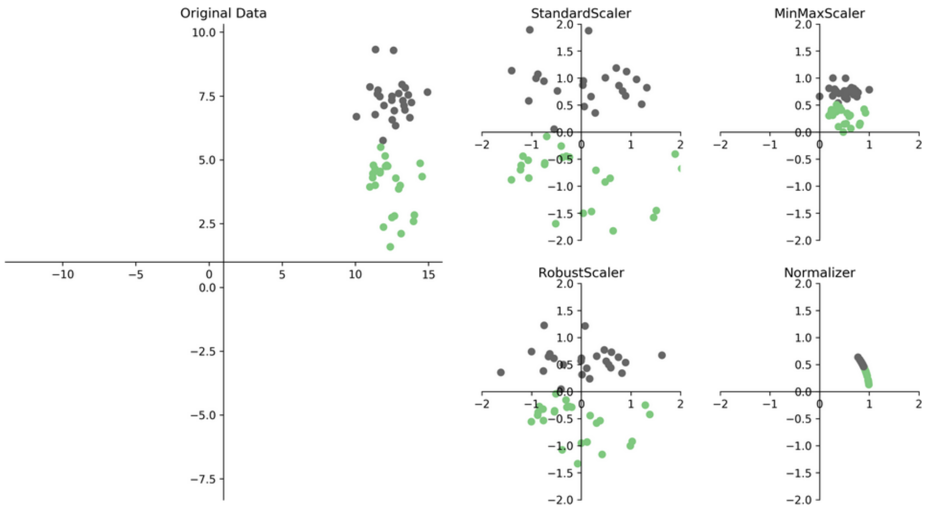
Scaling and Distances



Scaling and Distances



Ways to Scale Data



Categorical Variables

$$\{'red','green','blue'\} \subset R^p?$$

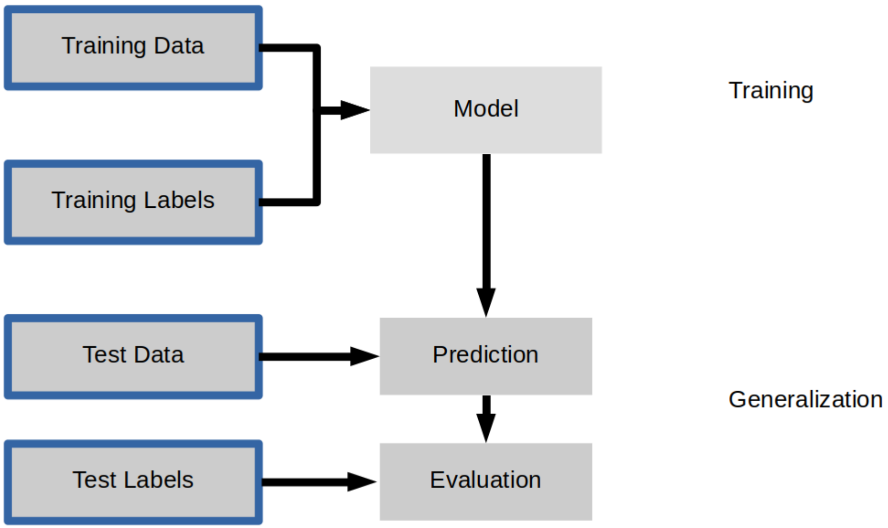
Categorical Variables

$$\begin{array}{ccc} \text{“red”} & \text{“green”} & \text{“blue”} \\ \end{array}$$
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

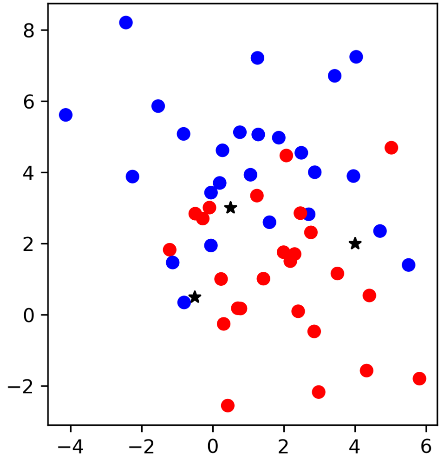
Jupyter Notebook

Part 2- PreProcessing

Supervised ML Workflow

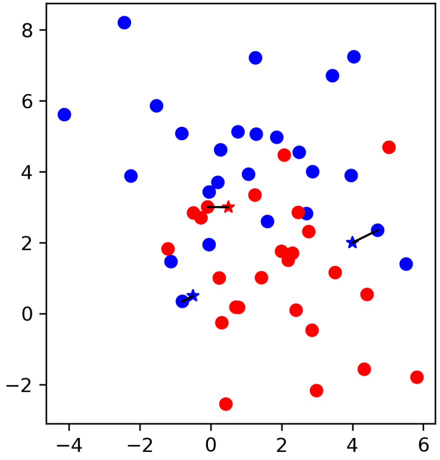


Nearest Neighbors



$$f(x) = y_i, i = \text{argmin}_i ||x_i - x||$$

Nearest Neighbors



$$f(x) = y_i, i = \text{argmin}_i ||x_i - x||$$

training set

$$X = \begin{pmatrix} 1.1 & 2.2 \\ 6.7 & 0.5 \\ 2.4 & 9.3 \\ 1.5 & 0.0 \\ 0.5 & 3.5 \\ 5.1 & 9.7 \\ 3.7 & 7.8 \end{pmatrix} \qquad y = \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

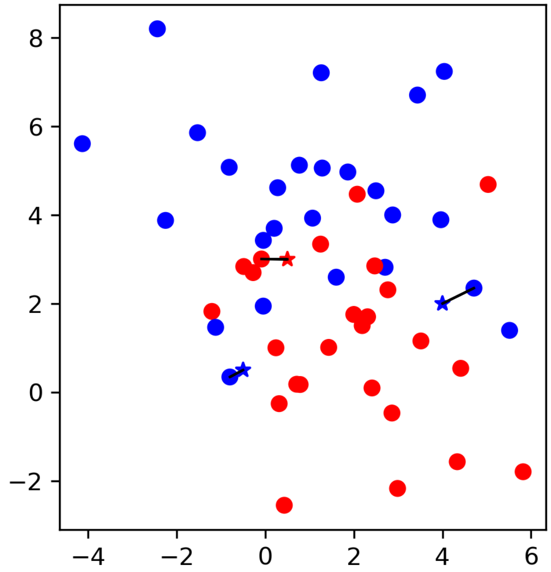
test set

KNN with scikit-learn

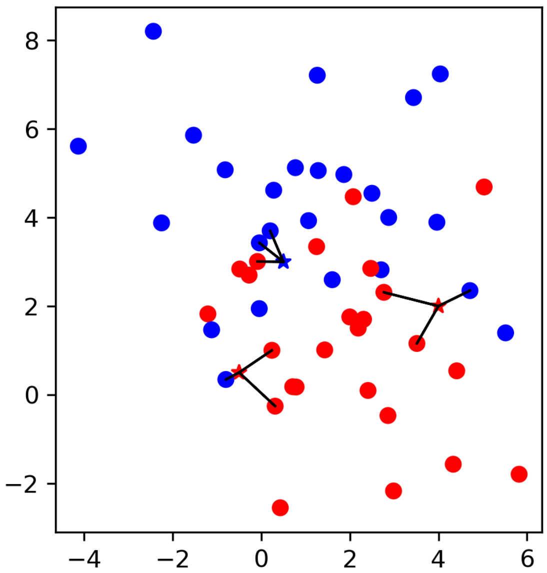
```
     sklearn.model_selection        train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y)

     sklearn.neighbors        KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
knn.fit(X_train, y_train)
print("accuracy: {:.2f}".format(knn.score(X_test, y_test)))
y_pred = knn.predict(X_test)
```

accuracy: 0.77

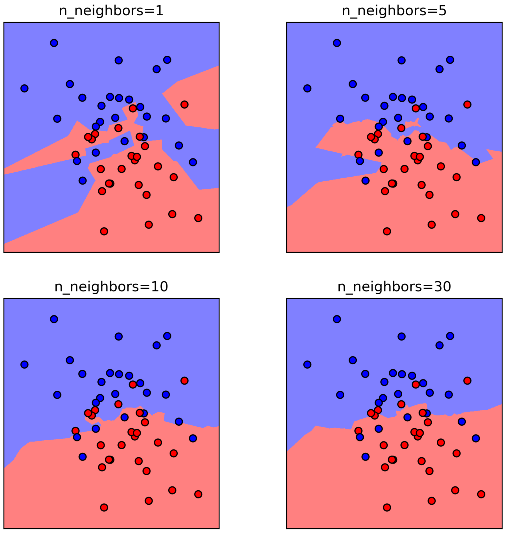
More neighbors



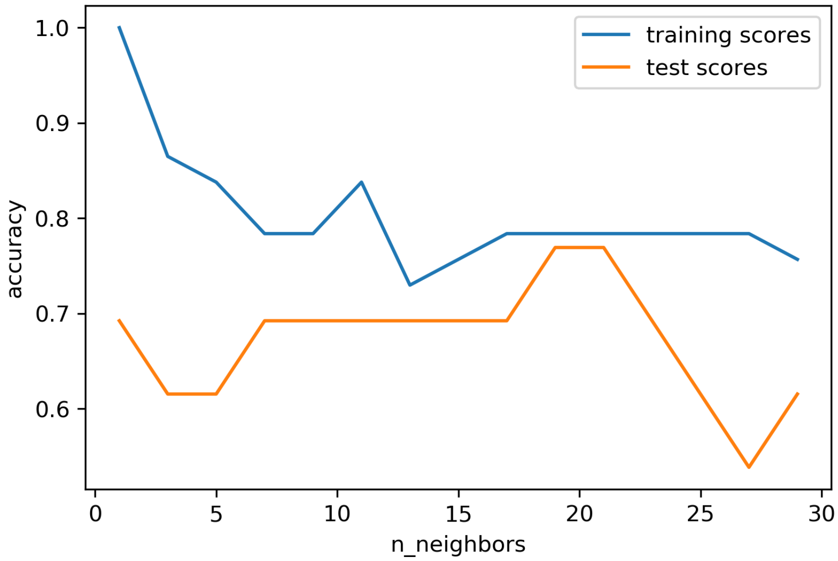
More neighbors



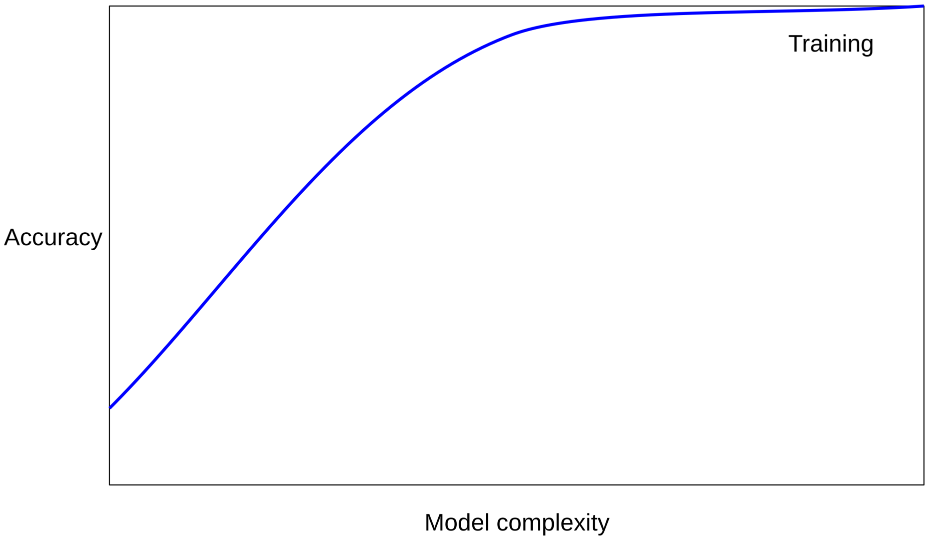
Influence of n_neighbors



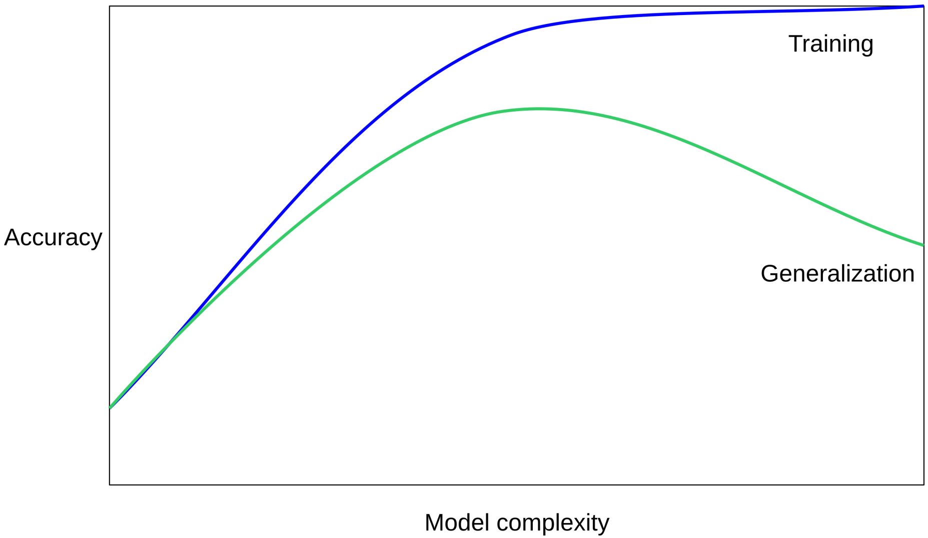
Model complexity



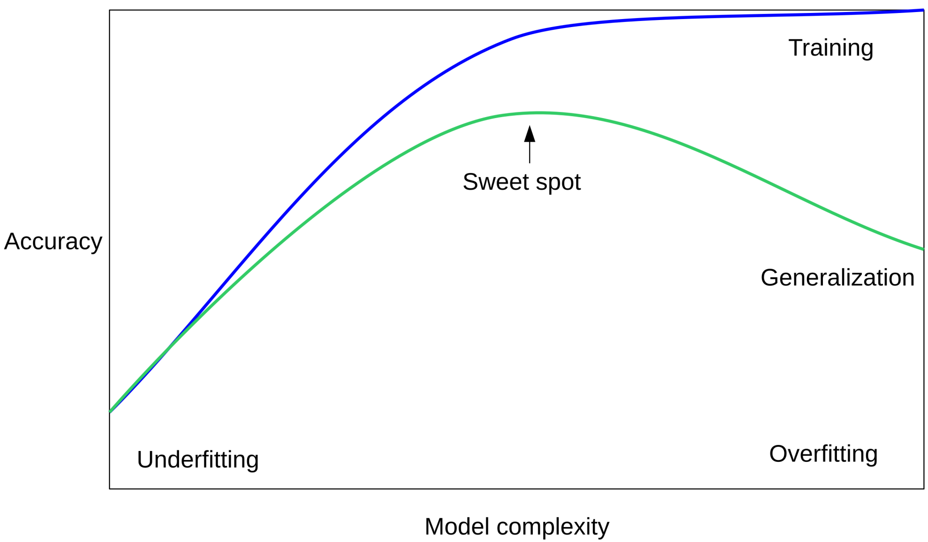
Overfitting and Underfitting



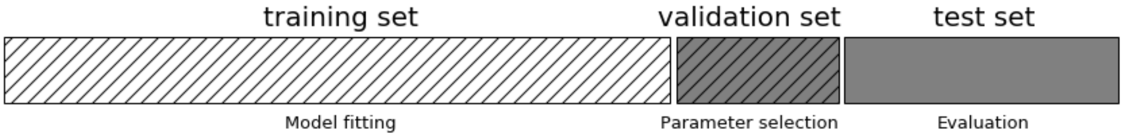
Overfitting and Underfitting



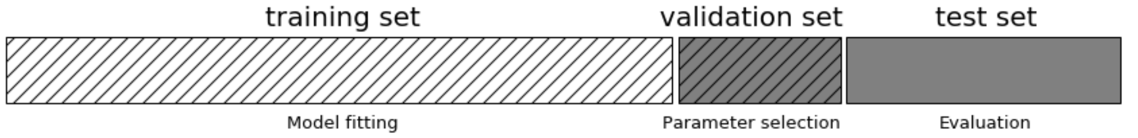
Overfitting and Underfitting



Threefold split



Threefold split



pro: fast, simple con: high variance, bad use of data

Implementing threefold split

```
X_trainval, X_test, y_trainval, y_test = train_test_split(X, y)
X_train, X_val, y_train, y_val = train_test_split(X_trainval, y_trainval)

val_scores = []
neighbors = np.arange(1, 15, 2)
    i    neighbors:
    knn = KNeighborsClassifier(n_neighbors=i)
    knn.fit(X_train, y_train)
    val_scores.append(knn.score(X_val, y_val))
print("best validation score: {:.3f}".format(np.max(val_scores)))
best_n_neighbors = neighbors[np.argmax(val_scores)]
print("best n_neighbors:", best_n_neighbors)

knn = KNeighborsClassifier(n_neighbors=best_n_neighbors)
knn.fit(X_trainval, y_trainval)
print("test-set score: {:.3f}".format(knn.score(X_test, y_test)))
```

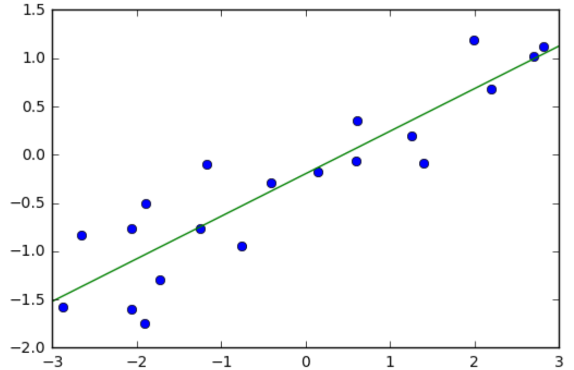
```
best validation score: 0.991
best n_neighbors: 11
test-set score: 0.951
```

Part 3: Jupyter Notebook

Supervised Learning

Linear Models for Regression

Linear Models for Regression



$$\hat{y} = w^T \mathbf{x} + b = \sum_{i-1}^{p} w_i x_i + b$$

Ordinary Least Squares

$$\hat{y} = w^T \mathbf{x} + b = \sum_{i=1}^{p} w_i x_i + b$$

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{p} ||w^T \mathbf{x}_i - y_i||^2$$

Ridge Regression

$$\min_{w \in \mathbb{R}^p} \sum_{i=1}^{n} ||w^T \mathbf{x}_i - y_i||^2 + \alpha ||w||^2$$

Always has a unique solution. Tuning parameter alpha.

Part 4: Jupyter Notebook

Linear Models for Regression