Supervised Learning using SKlearn

The Iris dataset in scikit-learn

```
from sklearn import datasets
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
plt.style.use('ggplot')
iris = datasets.load_iris()
type(iris)
```

```
sklearn.datasets.base.Bunch
```

```
print(iris.keys())
```

```
dict_keys(['data', 'target_names', 'DESCR', 'feature_names', 'target'])
```

Scikit-learn fit and predict

- All machine learning models implemented as Python classes
  - They implement the algorithms for learning and predicting
  - Store the information learned from the data
- Training a model on the data = 'fitting' a model to the data
  - `.fit()` method
- To predict the labels of new data: `.predict()` method

Using scikit-learn to fit a classifier

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(iris['data'], iris['target'])
```

```
KNeighborsClassifier(algorithm='auto', leaf_size=30,
    metric='minkowski',metric_params=None, n_jobs=1,
    n_neighbors=6, p=2,weights='uniform')
```

```
iris['data'].shape
```

```
(150, 4)
```

```
iris['target'].shape
```

```
(150,)
```

Predicting on unlabeled data

```
X_new = np.array([[5.6, 2.8, 3.9, 1.1],
       [5.7, 2.6, 3.8, 1.3],
       [4.7, 3.2, 1.3, 0.2]])

prediction = knn.predict(X_new)
X_new.shape
```

```
(3, 4)
```

```
print('Prediction: {}'.format(prediction))
```

```
Prediction: [1 1 0]
```

Measuring model performance

- In classification, accuracy is a commonly used metric
- Accuracy = Fraction of correct predictions
- Which data should be used to compute accuracy?
- How well will the model perform on new data?

Measuring model performance

- Could compute accuracy on data used to fit classifier
- NOT indicative of ability to generalize
- Split data into training and test set
- Fit/train the classifier on the training set
- Make predictions on test set
- Compare predictions with the known labels

Train/test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
    train_test_split(X, y, test_size=0.3,
                     random_state=21, stratify=y)
knn = KNeighborsClassifier(n_neighbors=8)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
print(\"Test set predictions:\\n {}\".format(y_pred))
```

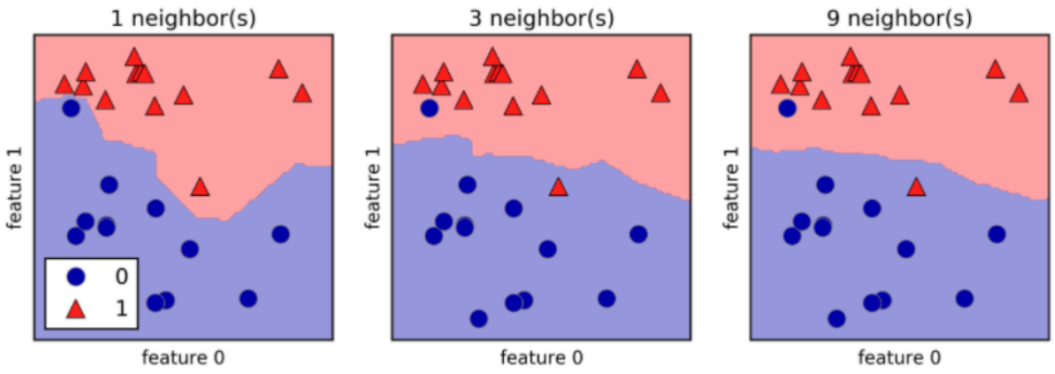
```
Test set predictions:
 [2 1 2 2 1 0 1 0 0 1 0 2 0 2 2 0 0 0 1 0 2 2 2 0 1 1 1 0 0
 1 2 2 0 0 2 2 1 1 2 1 1 0 2 1]
```

```
knn.score(X_test, y_test)
```

```
0.9555555555555556
```

Model complexity

- Larger k = smoother decision boundary = less complex model
- Smaller k = more complex model = can lead to overfitting



Model complexity and over/underfitting