Model Selection and Tuning: Takeaways

Syntax

- Instantiating a KNeighborsClassifier:

```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=1)
```

- Using the range class to iterate through values:

```
for k in range(1,8,2):
    print(k)
```

- Performing grid search using scikit-learn:

```
from sklearn.model_selection import GridSearchCV
knn = KNeighborsClassifier()
hyperparameters = {
    "n_neighbors": range(1,50,2)
}
grid = GridSearchCV(knn, param_grid=hyperparameters, cv=10)
grid.fit(all_X, all_y)
print(grid.best_params_)
print(grid.best_score_)
```

- Fitting and making predictions using the RandomForestClassifier:

```
from sklearn.ensemble import RandomForestClassifier
clf = RandomForestClassifier(random_state=1)
clf.fit(train_X,train_y)
predictions = clf.predict(test_X)
```

Concepts

- Model selection is the process of selecting the algorithm which gives the best predictions for your data. Each algorithm has different strengths and weaknesses and we need to select the algorithm that works best with our specific set of data.
- The k-nearest neighbors algorithm finds the observations in our training set most similar to the observation in our test set and uses the average outcome of those 'neighbor' observations to make a prediction. The 'k' is the number of neighbor observations used to make the prediction.
- Hyperparameter optimization, or hyperparameter tuning, is the process of varying the parameters of each model to optimize accuracy.
    - Grid search is one method of hyperparameter optimization.
    - Grid search trains a number of models across a "grid" of values and then searches for the model that gives the highest accuracy.
- Random forests is a specific type of decision tree algorithm. Decision tree algorithms attempt to build the most efficient decision tree based on the training data.

Resources

- Cross Validation and Grid Search
- Hyperparameter optimization