Solutions 3

Jumping Rivers

During the lecture we fit a logistic regression model to the breast cancer data for classifying tumors in patients. We are going to fit a KNN classifier to the same data.

- Construct the pipeline ready for fitting the model

```
from sklearn.datasets import load_breast_cancer
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.neighbors import KNeighborsClassifier

cancer = load_breast_cancer()
X_train, y_train = cancer.data, cancer.target

model = Pipeline([
  ('pre', StandardScaler()),
  ('model', KNeighborsClassifier())
])
```

- We want to find the best value of K for the classifier when optimising for recall, our motivation is that we want to correctly identify as many of the malignant tumours as possible. Start with a grid search over `k = [1,5,10,20,50,100]`

```
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import make_scorer, recall_score

def recall(y_true,y_pred):
  return recall_score(y_true, y_pred, pos_label=0)

rec = make_scorer(recall)

clf = GridSearchCV(model, param_grid={
  'model__n_neighbors' : [1,5,10,20,50,100]
}, cv=10, iid=False, return_train_score=False,
scoring=rec)
clf.fit(X_train,y_train)
```

```
## GridSearchCV(cv=10, error_score='raise-deprecating',
##        estimator=Pipeline(memory=None,
##      steps=[('pre', StandardScaler(copy=True, with_mean=True, with_std=True)), ('model', KNeighborsCl
##            metric_params=None, n_jobs=None, n_neighbors=5, p=2,
##            weights='uniform'))]),
##        fit_params=None, iid=False, n_jobs=None,
##        param_grid={'model__n_neighbors': [1, 5, 10, 20, 50, 100]},
##        pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
##        scoring=make_scorer(recall), verbose=0)
```

- Create a plot of the K parameter against the average recall score found in the cross validation grid search

```
import pandas as pd

output = pd.DataFrame(clf.cv_results_)[['param_model__n_neighbors','mean_test_score']]

import seaborn as sns
import matplotlib.pyplot as plt
plt.figure()
sns.lineplot(x='param_model__n_neighbors',y='mean_test_score',data = output)
plt.show()
```

- What region of K looks like it will give the best value?

```
## for me it is between 1 and 20
```

- Re-run your grid search across that region

```
clf = GridSearchCV(model, param_grid={
  'model__n_neighbors' : list(range(1,21))
}, cv=10, iid=False, return_train_score=False,
scoring=rec)
clf.fit(X_train,y_train)
```

```
## GridSearchCV(cv=10, error_score='raise-deprecating',
##        estimator=Pipeline(memory=None,
##      steps=[('pre', StandardScaler(copy=True, with_mean=True, with_std=True)), ('model', KNeighborsCl
##            metric_params=None, n_jobs=None, n_neighbors=5, p=2,
##            weights='uniform'))]),
##        fit_params=None, iid=False, n_jobs=None,
##        param_grid={'model__n_neighbors': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17,
##        pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
##        scoring=make_scorer(recall), verbose=0)
```

- What is the best parameter choice and the corresponding recall score?

```
clf.best_params_
```

```
## {'model__n_neighbors': 4}
```

```
clf.best_score_
```

```
## 0.9480519480519481
```

- Is this better than the Logistic regression in the notes?

```
## for me it is worse
```