KNN Example

Cancer Diagnosis

Breast Cancer Diagnosis

- The breast cancer data includes 569 examples of cancer biopsies, each with 32 features.
- One feature is an identification number, another is the cancer diagnosis, and 30 are numeric-valued laboratory measurements.
- The diagnosis is coded as M to indicate malignant or B to indicate benign.

Breast Cancer Data

- http://archive.ics.uci.edu/ml.
-

Breast Cancer Data

```
import numpy as np
import pandas as pd

df = pd.read_csv('data/wisc_bc_data.csv')

print(df.columns)
print(df.diagnosis.value_counts())

# print(df.head())
# print(df.area_mean.describe())
print('Mean of area_mean', df.area_mean.mean())
print('Mean of radius_mean', df.radius_mean.mean())
print('Mean of smoothness_mean', df.smoothness_mean.mean())

X = df.drop(['id', 'diagnosis'], axis='columns')
y = df.diagnosis
```

Breast Cancer Data

- Columns:
  - `['id', 'diagnosis', 'radius_mean', 'texture_mean', 'perimeter_mean', .....`
- First column is id field which has no predictive value.
- Second column is the diagnosis which is what we are trying to predict.
- Value counts for diagnosis:
  - B 357, M 212

X and y

- Drop ‘id’ and ‘diagnosis’ columns for X.
- y is the diagnosis column

Three Features

- area_mean (~655),
- radius_mean (~14),
- smoothness_mean (0.096)
- Have very different ranges of values
- This suggests we should scale the features before applying kNN.

Building a KNeighborsClassifier

```
import numpy as np
import pandas as pd
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix

df = pd.read_csv('data/wisc_bc_data.csv')

X = df.drop(['id', 'diagnosis'], axis='columns')
y = df.diagnosis'

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=5)

clf = KNeighborsClassifier()
clf.fit(X_train, y_train)
```

Evaluating the Model

```
print('Test Accuracy', clf.score(X_test, y_test))

yhat = clf.predict(X_test)
cm = confusion_matrix(y_test, yhat)
```

Confusion Matrix

Predicted / Actual	N	P	
B (N)	TN	FP	
M (P)	FN	TP	

confusion_matrix()

- First paramer – actual values
- Second paramet – predicted values
- Labels – B is first, M is second

Confusion Matrix

Predicted / Actual	N	P	
B (N)	72	0	
M (P)	6	36	

Scaling in scikit-learn

- StandardScaler
  - Each feature has mean 0 and SD 1.
- MinMaxScaler
- RobustScaler
  - Similar to StandardScaler but adjusts the median and quartiles. Therefore ignores outliers which could be measurement errors.
- Normalize
  - In 2D scales every point onto the unit circle.

Scaling in scikit-learn

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# or
# X_train = scaler.transform(X_train)
# X_test = scaler.transform(X_test)
```

Scaling

- Notice that the Scaler is fit using the training data.
- This is in line with not using the test data when building the model.