COMP 474 UU,COMP 6741 UU 2204

Lab Session #8

Introduction

Welcome to Lab #8: This week, we'll practice more machine learning with *scikit-learn*.

Follow-up Lab #7

Solution Task #1 (TF-IDF Vectors)

Here's a sample program for this task from last week that prints out various intermediate steps.

Solution Task #2 (Search)

Here's a solution for the re-implement of google

Solution Task #3 (k-Means)

Here's an example solution for the first part (clustering the test sentences).

Task #1: kNN Regression

The goal here is to implement the kNN regression exercise from lecture Worksheet #6. To start, use the statements below to import the required libraries (here is a nice cheat sheet for working with scitkit-learn from Datacamp):

```
import numpy as np

from sklearn.neighbors import KNeighborsRegressor
```

Now, create a dataset using the samples from the worksheet and train the dataset with the KNeighborsRegressor with n_neighbors = 2.

```
dataset = np.array([[135,0,5,3],[90,123,2,5],[159,2,1,1]])
```

For feature vectors we need the first three columns:

```
X = dataset[:,0:3]
```

For the training labels, we use the last column from the dataset:

```
y = dataset[:, 3]
```

Create regressor object and train the model.

```
clf = KNeighborsRegressor(2)
```

```
clf.fit(X, y)
```

Make predictions on the test data features

test data features = [109,5,3]

Task #2: kNN Classification

For these experiments, we will use the kNN algorithm as discussed in the lecture. Luckily, it's already implemented for us in *scitkit-learn:*

```
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
```

To see how this works, let's start with some real data dataset of sklearn. Here we are using wine dataset:

```
from sklearn.datasets import load_wine

X, y = load_wine(return_X_y=True)
```

Now create train and test data. Use Scikit-Learn's train_test_split helper function to split the wine dataset into training and testing subset.

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

Here data is split into 80% train data and 20% test data.

It is always a good practice to scale the features so that all of them can be uniformly evaluated.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)


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

Now you can "train" a classifier (for kNN, this simply stores the vectors with their labels):

```
clf = KNeighborsClassifier(n_neighbors=3)
clf.fit(X_train, y_train)
```

Here, "3" is *k*, the number of neighbors voting when classifying unseen data (see the documentation). Note that this is a standard pattern when creating a ML model with Scikit-learn, you can use other algorithms (e.g., Naive Bayes, SVM) in the same way.

Make predictions on the test data

```
y_pred = clf.predict(X_test)
```

Evaluate the performance of your classifier

Now run an evaluation to compute the Precision, Recall, F1-measure, and Accuracy of your classifier using the evaluation tools in *scikit-learn*. Finally, compute and print out the confusion matrix.

Task #3: Project Team Meeting

You should use any remaining time of the lab session to meet with your TA and team members in a breakout session to discuss any outstanding issues and plan for the remaining week ahead.

Please post any open questions you still have in the Moodle Discussion Forum!

That's all for this lab!

Last modified: Thursday, 18 March 2021, 5:20 PM

◄ Worksheet #08 Jump to... Lecture Slides #10 ►