ITF31519 - Assignment 1

Tobias Hallingstad

September 29, 2021

1 library's

Numpy In this task Numpy is only used for sorting, but the library has many ways of working with data in lists.

Pandas To parse the data files I use the Pandas library. Pandas stores the data in a `DataFrame` object.

Matplotlib For the graphs in the task I am using Matplotlib's pylpot library. Here I am only using it for scatter and line plots.

Sklearn For the general Machine learning part of the task I decided to use Sklearn. This is because I found the functions needed to solve the task and the library is used in the example code given.

2 Function's

Parsing the data All the data in this task needs to be read form a file, this is done using the Pandas library.

```
data = pd.read_csv('path/to/file')
```

Here `data` is a Pandas `DataFrame` object and only stores the data read form the file given.

Afther reading the file I split the data into data and lable. This is done like this:

```
x = data.iloc[:,0]   # Task 1
x = data.iloc[:,0:4] # Task 2
y = data.iloc[:,1]   # Task 1 and 2
```

Using the `DataFrame` function `iloc[]` I can take a set of colums from data and put them into their own variables

Splitting data In task 1 and task 2 the data needs to be split into one part test data and one part training data. To split the data into the 2 parts I am using the static function from the Sklearn library.

```
testSetPrecentage = 0.2
trainX, testX, trainY, testY =
    train_test_split(x, y, test_size=testSetPrecentage)
```

This function will split add the data and its corresponding lable with a set percentage, defining the size of the test data. In this case the test data is 20% of all the data, and the training data is 80%. Splitting the data is randomized.

Argsort Some of the functions in part 1 requires that the data is sorted. To do this I am using the Numpy `argsort()` this function sort's returns a list of the indexes to get a sorted list. This mens that the `data` variable does not change, but i can still access all the data in sorted order.

Plotting For plotting the data in part 1 I am using a combination of `scatter()` and `plot()`. The scatter function is used to plot each data point, and the plot function is used for the line.

Linear regression In part 1 the task is to use linear regression to analyse the data. For this I decided to use the linear regression object in `sklearn`. The point of linear regression is to finde a mathematical relationship between the learing data and the classification of that data, so when we get some new data point we can finde a estimate of what that data means.

When i splitted the data the data ended up in the wrong format, so I needed to reassign the data.

```
1  trainX = trainX.values
2  trainY = trainY.values
3  testX = testX.values
4  testY = testY.values
```

I can now create the object and fit the data to the object, or model.

```
1  reg = linear_model.LinearRegression()
2  reg.fit(trainX.reshape(-1,1), trainY)
```

Because of how the data is formattet I need to rechape the training data so it fits what the `fit()` function needs. Here the `reshape(-1,1)` means that the data is formattet from a 1D array to a 2D array.

KNN In part 2 the task is to use K nearest neighbors to classefy a new sett with data. I decided to use the `sklearn` library `KNeighborsClassifier`. This algorithm is uesd to finde a class for some new data point using the datapoints closest to this new point. In other words, we find the K number

of nearest neighbors to a new point and determen the class lable for the point.

The implementation of KNN is simple. One just have to assign the `KNeighborsClassifier()`, sett a K. Then `fit()` the data.

```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(trainX, trainY)
```

For this algorithm I did not need to change the data, after splitting. This is because the data already is in a 2D array, unlike in the linear regression part.

To do a prediction in the KNN functon is simple. One just need to call the `knn.predict()` and give it some data to predict. This will return the classlable for each element in the list.

```
predY = knn.predict(testX)
```

Accracy score In part 2 we was also going to calculate the accracy score of the prediction. I decided to use the `metrics` class from `sklearn`. In this code I am just printing the accracy score.

```
print('Accracy:', metrics.accuracy_score(testY, predY))
```

3 procedure's

For part 1 and part 2 I used almost the same procedure. The main difference is that I needed to change how the lists was formated in part 1, bacause if the 1 dimention the list was in. But the general procedure was like this:

Parse the data First I parse the data using `pandas`.

Split the data Afther the data is read and parsed I then split the data from the classifier. This is so I can use the data in the functions later.

Split the data into train and test For training I need a subsett of the original data for making the model and some data to test the model. So I am splitting the data into a set of train/test data. The data is piced ranomly and the precent of data that goes to test data is defined.

Make shure the data is in the correct form In part 1 I needed to change how the data is formated so it fits what the functions need as parameters. This is done now.

Fit the data When all of the data is correctly formattet I can create the model class and fit the training data to the model.

Presenting the model Now that the data has been fitted to the model the data can be worked with or presented. Eather by plotting the data to a graf or calculating the accracy score of the model created.