Assignment 27.

**Problem Statement**

In this assignment, students will be using the K-nearest neighbors algorithm to predict

how many points NBA players scored in the 2013-2014 season.

A look at the data

Before we dive into the algorithm, let’s take a look at our data. Each row in the data

contains information on how a player performed in the 2013-2014 NBA season.

Download 'nba\_2013.csv' file from this link:

https://www.dropbox.com/s/b3nv38jjo5dxcl6/nba\_2013.csv?dl=0

Here are some selected columns from the data:

player - name of the player

pos - the position of the player

g - number of games the player was in

gs - number of games the player started

pts - total points the player scored

There are many more columns in the data, mostly containing information about average

player game performance over the course of the season. See this site for an explanation

of the rest of them.

We can read our dataset in and figure out which columns are present:

import pandas

with open("nba\_2013.csv", 'r') as csvfile:

nba = pandas.read\_csv(csvfile)

Code:

import pandas

with open("Downloads/nba\_2013.csv", 'r') as csvfile:

nba = pandas.read\_csv(csvfile)

# The names of all the columns in the data.

print(nba.columns.values)

selected\_player = nba[nba["player"] == "LeBron James"].iloc[0]

distance\_columns = ['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p', 'x3pa', 'x3p.', 'x2p', 'x2pa', 'x2p.', 'efg.', 'ft', 'fta', 'ft.', 'orb', 'drb', 'trb', 'ast', 'stl', 'blk', 'tov', 'pf', 'pts']

nba.shape

nba=nba.dropna()

nba.shape

nba.head(8)

import math

import matplotlib.pyplot as plt

player = nba[nba["player"] == "Kevin Durant"].iloc[0]

# Use Numeric Columns Only

measure = ['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p', 'x3pa', 'x3p.', 'x2p', 'x2pa', 'x2p.', 'efg.', 'ft', 'fta', 'ft.', 'orb', 'drb', 'trb', 'ast', 'stl', 'blk', 'tov', 'pf', 'pts']

# find the distance

def ed(row):

value = 0

for k in measure:

value += (row[k] - player[k]) \*\* 2

return math.sqrt(value)

# return result

durant = nba.apply(ed, axis=1)

player = nba[nba["player"] == "Carmelo Anthony"].iloc[0]

# Use Numeric Columns Only

measure = ['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p', 'x3pa', 'x3p.', 'x2p', 'x2pa', 'x2p.', 'efg.', 'ft', 'fta', 'ft.', 'orb', 'drb', 'trb', 'ast', 'stl', 'blk', 'tov', 'pf', 'pts']

# find the distance

def ed(row):

value = 0

for k in measure:

value += (row[k] - player[k]) \*\* 2

return math.sqrt(value)

# return result

anthony = nba.apply(ed, axis=1)

player = nba[nba["player"] == "LeBron James"].iloc[0]

# Use Numeric Columns Only

measure = ['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p', 'x3pa', 'x3p.', 'x2p', 'x2pa', 'x2p.', 'efg.', 'ft', 'fta', 'ft.', 'orb', 'drb', 'trb', 'ast', 'stl', 'blk', 'tov', 'pf', 'pts']

# find the distance

def ed(row):

value = 0

for k in measure:

value += (row[k] - player[k]) \*\* 2

return math.sqrt(value)

# return result

james = nba.apply(ed, axis=1)

plt.figure(figsize=(20,7))

plt.plot(durant, label='Durant')

plt.plot(anthony, label='Anthony')

plt.plot(james, label='James')

plt.ylabel("Euclidian-Distances")

plt.legend()

plt.grid(True)

plt.show()

# Select only the numeric columns from the NBA dataset

nba\_numeric = nba[measure]

# Normalize all of the numeric columns

nba\_normalized = (nba\_numeric - nba\_numeric.mean()) / nba\_numeric.std()

nba\_normalized.head(8)

from scipy.spatial import distance

# Find the normalized vector for lebron james.

durant\_normalized = nba\_normalized[nba["player"] == "Kevin Durant"]

# Find the distance between lebron james and everyone else.

euclidean\_distances = nba\_normalized.apply(lambda row: distance.euclidean(row, durant\_normalized), axis=1)

# Create a new dataframe with distances.

distance\_frame = pandas.DataFrame(data={"dist": euclidean\_distances, "idx": euclidean\_distances.index})

distance\_frame.sort\_values(by="dist", inplace=True)

# Find the most similar player to lebron (the lowest distance to lebron is lebron, the second smallest is the most similar non-lebron player)

second\_smallest = distance\_frame.iloc[1]["idx"]

most\_similar = nba.loc[int(second\_smallest)]["player"]

most\_similar

import random

from numpy.random import permutation

# Randomly shuffle the index of nba.

random\_indices = permutation(nba.index)

# Set a cutoff for how many items we want in the test set (in this case 1/3 of the items)

test\_cutoff = math.floor(len(nba)/3)

# Generate the test set by taking the first 1/3 of the randomly shuffled indices.

test = nba.loc[random\_indices[1:test\_cutoff]]

# Generate the train set with the rest of the data.

train = nba.loc[random\_indices[test\_cutoff:]]

# The columns that we will be making predictions with.

x\_columns = ['age', 'g', 'gs', 'mp', 'fg', 'fga', 'fg.', 'x3p', 'x3pa', 'x3p.', 'x2p', 'x2pa', 'x2p.', 'efg.', 'ft', 'fta', 'ft.', 'orb', 'drb', 'trb', 'ast', 'stl', 'blk', 'tov', 'pf']

# The column that we want to predict.

y\_column = ["pts"]

from sklearn.neighbors import KNeighborsRegressor

# Create the knn model.

# Look at the five closest neighbors.

knn = KNeighborsRegressor(n\_neighbors=7)

# Fit the model on the training data.

knn.fit(train[x\_columns], train[y\_column])

# Make point predictions on the test set using the fit model.

predictions = knn.predict(test[x\_columns])

# Get the actual values for the test set.

actual = test[y\_column]

# Compute the mean squared error of our predictions.

mse = (((predictions - actual) \*\* 2).sum()) / len(predictions)

print("actual[:20]:\n", actual[:20])

print("mse:", mse)

**Output:**









