Microsoft: DAT210x Programming with Python for Data Science

6. Data Modeling II > Lab: SVC > Assignment 3

Bookmarks

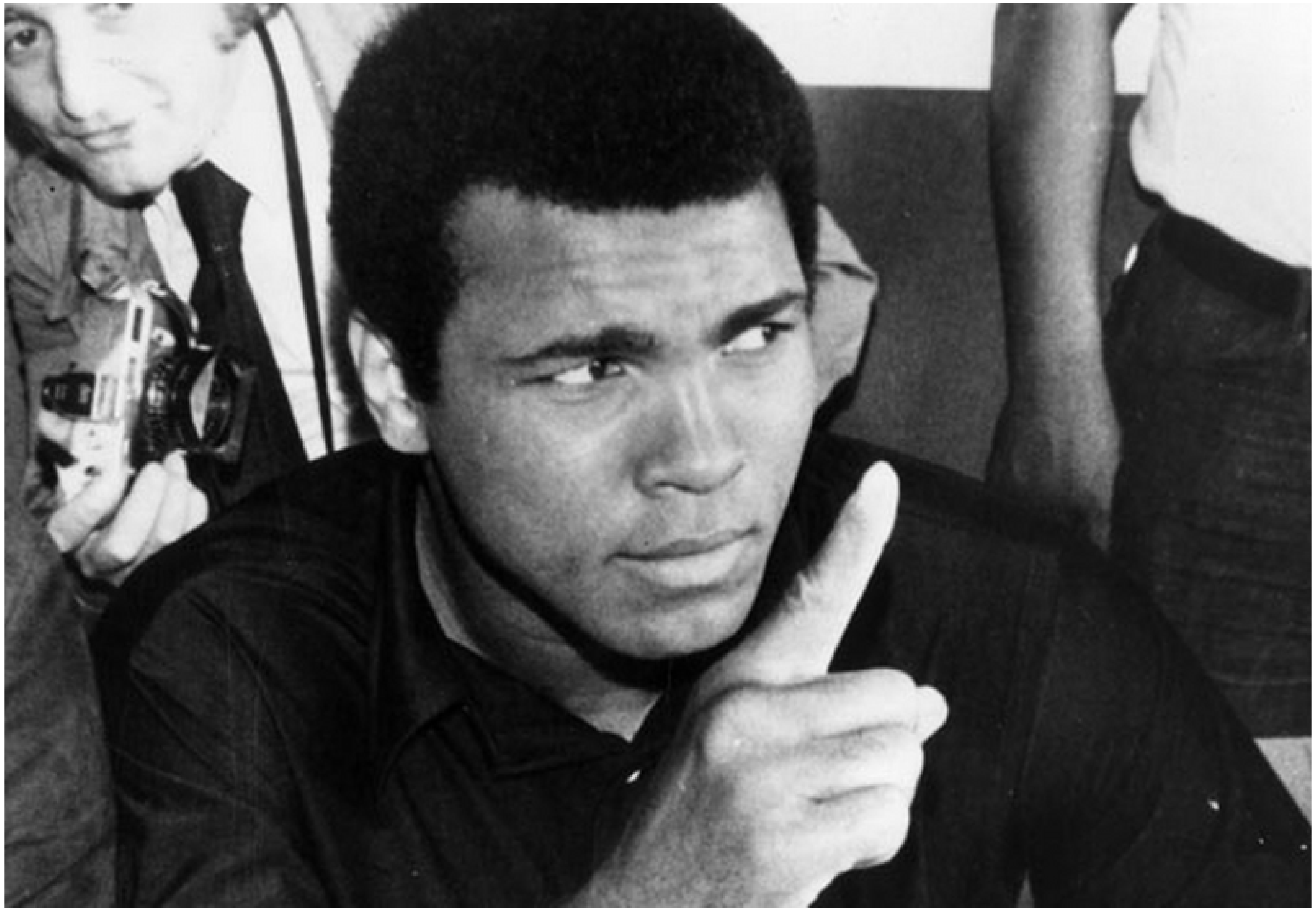
Bookmark

Lab Assignment 3

Growing up, everyone has a hero. For many people, that hero was Muhammad Ali. He taught people it was okay to be proud of who they were, at a time when others would not accept that. He showed people how to stand up for their beliefs in the face of oppression and tyranny. He made people value themselves, and encouraged them care for those around them. He showed us what bravery truly meant, how to be a heck of a boxer, and so much more. Every single person who met Muhammad Ali, either in the ring or outside of it, had a motivating story to share about their encounter.

On June 3, 2016, Muhammad Ali passed away at the age of 74 due to septic shock. Thirty years earlier, he was diagnosed with Parkinson's syndrome, a neurodegenerative condition that doctors attributed to his boxing-related brain injuries.

Dive Deeper



Parkinson's disease itself is a long-term disorder of the nervous system that affects many aspects of a person's mobility over time. It's characterized by shaking, slowed movement, rigidity, dementia, and depression. In 2013, some 53 million people were diagnosed with it, mostly men. Other famous personalities affected by it include actor Michael J Fox, and olympic cyclist Davis Phinney.

In this lab, you will be applying SVC to the Parkinson's Data Set, provided courtesy of UCI's Machine Learning Repository. The dataset was created at the University of Oxford, in collaboration with 10 medical centers around the US, along with Intel who developed the device used to record the primary features of the dataset: speech signals. Your goals for this assignment are first to see if it's possible to differentiate between people who have Parkinson's and who don't using SciKit-Learn's support vector classifier, and then to take a first-stab at a naive way of fine-tuning your parameters in an attempt to maximize the accuracy of your testing set.

> *"I've never really resented hard work because I've always liked it. Up every morning for roadwork. Going to the gymnasium every day at 12 o'clock. I never change my pattern."*

In honor of Muhammad Ali and hard work, there is no starter code for this lab. Just follow the instructions below.

Lab Question 1

(1/1 point)

Load up the /Module6/Datasets/parkinsons.data data set into a variable X, being sure to drop the name column.

Splice out the status column into a variable y and delete it from X.

Perform a train/test split. 30% test group size, with a random_state equal to 7.

Create a SVC classifier. Don't specify any parameters, just leave everything as default. Fit it against your training data and then score your testing data.

What accuracy did you score?

0.81355932203389836 ✓

You have used 2 of 3 submissions

Lab Question 2

(1 point possible)
That accuracy was just too low to be useful. We need to get it up. Once way you could go about doing that would be to manually try a bunch of combinations of *C*, and *gamma* values for your rbf kernel. But that could literally take forever. Also, you might unknowingly skip a pair of values that would have resulted in a very good accuracy.

Instead, let us allow computers to do what computers do best. Program a naive, best-parameter searcher by creating a nested for-loops. The outer for-loop should iterate a variable *C* from 0.05 to 2, using 0.05 unit increments. The inner for-loop should increment a variable *gamma* from 0.001 to 0.1, using 0.001 unit increments. As you know, Python ranges won't allow for float intervals, so you'll have to do some research on NumPy ARanges, if you don't already know how to use them.

Since the goal is to find the parameters that result in the model having the best score, you'll need a *best_score = 0* variable that you initialize outside of the for-loops. Inside the for-loop, create a model and pass in the *C* and *gamma* parameters into the class constructor. Train and score the model appropriately. If the current *best_score* is less than the model's $sc\ \mathrm{or}\ e$, then update the *best_score*, being sure to print it out, along with the *C* and *gamma* values that resulted in it.

After running your assignment again, what is the highest accuracy score you are able to get?

0.88135593220338981 ✗

You have used 2 of 2 submissions

Lab Question 3

(1/1 point)
Wait a second. Pull open the dataset's label file from: https://archive.ics.uci.edu/ml/datasets/Parkinsons

Look at the units on those columns: Hz, %, Abs, dB, etc. What happened to transforming your data? With all of those units interacting with one another, some pre-processing is surely in order.

Right after you splice out the status column, but before you process the train/test split, inject SciKit-Learn pre-processing code. Unless you have a good idea which one is going to work best, you're going to have to try the various pre-processors one at a time, checking to see if they improve your predictive accuracy.

Experiment with *Normalizer()*, *MaxAbsScaler()*, *MinMaxScaler()*, and *StandardScaler()*.

After trying all of these scalers, what is the new highest accuracy score you're able to achieve?

0.932203389831 ✓

You have used 1 of 2 submissions

Lab Question 4

(1/1 point)
The accuracy score keeps creeping upwards. Let's have one more go at it. Remember how in a previous lab we discovered that SVM's are a bit sensitive to outliers and that just throwing all of our unfiltered, dirty or noisy data at it, particularly in high-dimensionality space, can actually cause the accuracy score to suffer?

Well, let's try to get rid of some useless features. Immediately after you do the pre-processing, run ISO on your dataset. The original dataset has 22 columns and 1 label column. So try experimenting with PCA n_component values between 4 and 14. Are you able to get a better accuracy?

If you are not, then forget about PCA entirely, unless you want to visualize your data. However if you are able to get a higher score, then be *sure* keep that figure in mind, and comment out all the PCA code.

In the same spot, run Isomap on the data, before sending it to the train / test split. Manually experiment with every inclusive combination of n_neighbors between 2 and 5, and n_components between 4 and 6. Are you able to get a better accuracy?

If you are not, then forget about isomap entirely, unless you want to visualize your data. However if you are able to get a higher score, then be *sure* keep that figure in mind.

If either PCA or Isomap helped you out, then uncomment out the appropriate transformation code so that you have the highest accuracy possible.

What is your highest accuracy score on this assignment to date?

0.966101694915 ✓ **Answer:** 0.966101694915

EXPLANATION

You should have scored 0.966101694915, an amazing improvement over your original 0.813559322034 score. Your classifier isn't great yet, but at least it's useable. There are a few more tricks you can use to get it better yet, but that'll have to wait for another lab.

You have used 1 of 2 submissions