01-K Nearest Neighbors with Python (Explanation)

Graphical user interface, text

Description automatically generated

Start by just importing Panthers PD importing them as pd and then importing my visualization libraries. That's going to be Matplotlib. pyplot as plt. import seaborne as sns.

We're going to pretend that we've been given a classified data set from a company and actually a lot of times if you're beginning to interview for data science positions at certain companies they will give you a dataset as a take home project. Want you to analyze it or do some sort of machine learning algorithm on it but they won't actually tell you specific values or what is represented by a column. And that's an order to protect their customer data or just to give you an anonymized data source.

Graphical user interface, text, application

Description automatically generated

We'll start by studying df is equal to pd.read\_csv and I'm going to go ahead and read in the classified data. And one other note is I'm going to set the index call to zero and then let's go ahead and check the head of that data frame. So you notice we have a bunch of data but we just have a target class. Column 1 or 0. And essentially just random letters for the column names. And this is essentially just anonymised classified data. So you don't know any of these numbers represent. Or you also don't know what these column names represent. You just know that you need to use these features that are unknown to you as far as what they actually represent in order to predict a target class 1 or 0 because the K and classifier predicts the class of a given test observation by identifying the observations that are nearest to it.

Graphical user interface, text, application, email

Description automatically generated

The scale of the variable actually matters a lot and any variables that are on a large scale will have a much larger effect on the distance between observations and because of this when you're using KNN to say fire which going to want to do is try to standardize everything to the same scale.

Luckily Scikit learned actually has a lot of built in tools to help you through this process. We're going to go ahead and say from sklearn pre-processing import StandardScaler and you can just

use tab to help yourself autocomplete some of this in standard scaler is going to look a lot as if it

were just a normal model you're going to go ahead and create an instance of standard scaler just like you would for the machine learning algorithm.

And I'm going to go ahead and call my instance scaler and then I'm going to say scaler and I'm going to fit it to my data and I just want to fit it to my data not the actual target class. So instead of just saying df, I will say the drop target class along axis is equal to 1. So that's basically all the feature columns that are going to fit this scalar object to. And then what I can do is use the scalar object to do a transformation. So I end up doing as you say scaled\_features or whatever value you want here. Variable name I should say take that scalar object and call the transform method on it. And what the transform method does it just performs the standardization by centering and scaling.

So you already have the object fitted created and we're going to do is use transform actually transform that data and we're going to go ahead and pass in that data one more time.

This features target class and then set axis = 1. And now if we go ahead and check out the scaled features notice we have an array of values and this is the scaled version of these actual values. Now if you'll notice these actual values are quite close to each other. It's always a good idea to do some sort of standard scaling transformation on this. What I'm going to go ahead and do is use that scaled features variable to recreate a feature. So df\_feat =pd.dataframe assigning scaled\_ features as the data and then for the column names is equal to the idea that columns. Columns is a way we can grab a list of all the column names and this is referencing that original. So here we have all the column names. However remember this is actually just the features. So I want everything but the last one. And you can actually do that just by doing slice notation. So this is everything but the last one. And now we have a standard scale or standardized version of our data. And now our data is ready to be put into a machine learning algorithms such as KNN which really depends on the distance between each feature.

Graphical user interface, text, application

Description automatically generated

OK let's go ahead and move along to the train test split now that our data is ready we'll say from sklearn import the cross-validation import train test split. Set my test size = 0.3.

Now what we need to do is say are x and y so are X we can actually just pass this in either the df\_feat or just scaled features. It's up to you. So you could just the array of values. And now we have our trained test split. Now it's time to use KNN and remember that we are trying to come up with a model to predict whether someone will be inside that target class or not.

Graphical user interface, text, application, email

Description automatically generated

We're going to go ahead and do it start with K's equal to 1 and then see how we can use the elbow method to choose a caved value to start off. So again from S-K learn that neighbors that's the family that this is an important in the model we want is Caye neighbors classifier that I'm going to go ahead and say k and n is equal to K neighbors classifier and if we do shift tab here we can see the various parameters where we're going to want to do is specify the and underscore neighbors parameter.

That's the number of neighbors you want for this model. Well go ahead and state that the number of neighbors just to start out with is equal to 1. And then we go and say k and n fit and then pass our training data. So X train, y train and we went ahead and fit that data. Now let's go ahead and grab some predictions off of it so we can do some evaluation. I'm going to say pred is equal to knn. predict and then passing my test data. Now if I look at this I have predictions of a class.

These people belong to. They of these anonymised features. Let's go ahead and do a prediction and evaluation off our KNN model and then we're going to go ahead to move on to using the elbow method to choose a correct value I'm going to say from sklearn that metrics I'll go ahead and import a classification report.

Graphical user interface, text, application, email

Description automatically generated

And the confusion matrix, print out the confusion matrix. Have y\_test and our predictions and let's also print out our classification report with Y\_test and our predictions.

We can squeeze even more from our model by choosing an even better k value. Let me go ahead and show you how we can do that. Well we want to do is use the **elbow method** to choose a correct k value. I'm going to go ahead and say my error rates is an empty list and what I'm going to do is basically iterate many models using many different k values and plot out their error rate and see which one has the lowest error rate.

When are you going to break this down step by step.

Graphical user interface, text, application

Description automatically generated

I have a empty list called error\_rate and I'm going to go in and say for i in range from 1 to 40 and you can choose smaller number instead of 40 if you don't want to do that many K values we're basically going to check every possible k value from 1 to 40 and for all of those values I'm going to call k neighbors classifier So I create a model at that specific k value. Then I go ahead and fit that model to my training set. So X train y train. I'm going to go ahead and say pred\_i and predicts off the test set. And then finally I'm going to say my error rate the pens.

So I added new a new item to this list and I'm going to append the mean of pred\_i not equal to Y test. And that's essentially the average error rate.So that's the average of where my predictions were not equal to the actual test values. Take the high mean of that and then append that error rates to this list.

And what I want to do now is if we go ahead and just take a look at what our error rate looks like it's a bunch of values here.. So I'll say appeal that figure size and then we'll continue on by saying Piazzi plot. I'm going to plot range 1 through 40 versus my error rates and I'm going to add a couple more arguments here I'm going to say the color is blue.

Text, letter

Description automatically generated

We start off with K= 1.

Now this comes at the cost than the fact that you have to run all these models plot them all out and actually spend the time to figure out what your optimal value is.