**CS 240 PROJECT**

**Name: Ahmet Ziyaeddin Sarı**

**Student No: 215363500**

**Introduction:**

In this project we are going to analyze a baseball data which contain pitching and batting statistics from 1871 to 2016. We will be analyzing the data by using modules like Pandas, Numpy, ThinkPlot, and Thinkstats2. This module will be helping us to put the data in data frames and plot them on tables.

**Part 1:**

By using this baseball data, we can analyze multiple things. But first of all, we need to ask some questions to observe. For example:

1. By using “Salaries.csv”, can we analyze in which league are players paid more? Is there any relationship between the leagues? Is American
2. By using “BattingPost.csv”, can we analyze the relationship with hits and runs? If they can hit, does it mean they can also run? Is there a positive relationship or negative relationship?
3. By using “BattingPost.csv”, can we analyze the relationship between homeruns and wins? Are homeruns effective in winning?

I’m going to observe the second question which analyzes the relationship between hits and runs.

**Part 2:**

The columns I am going to use is from the file “Batting.csv”. I’m going to observe if there is any relationship with hits and runs. I am going to observe the similarity between the hitter and runner.

from \_\_future\_\_ import print\_function, division

%matplotlib inline

import numpy as np

These are the modules I have used with in this project.

import pandas as pd

import thinkstats2

import thinkplot

I have imported the “Batting.csv” by using the Pandas module.

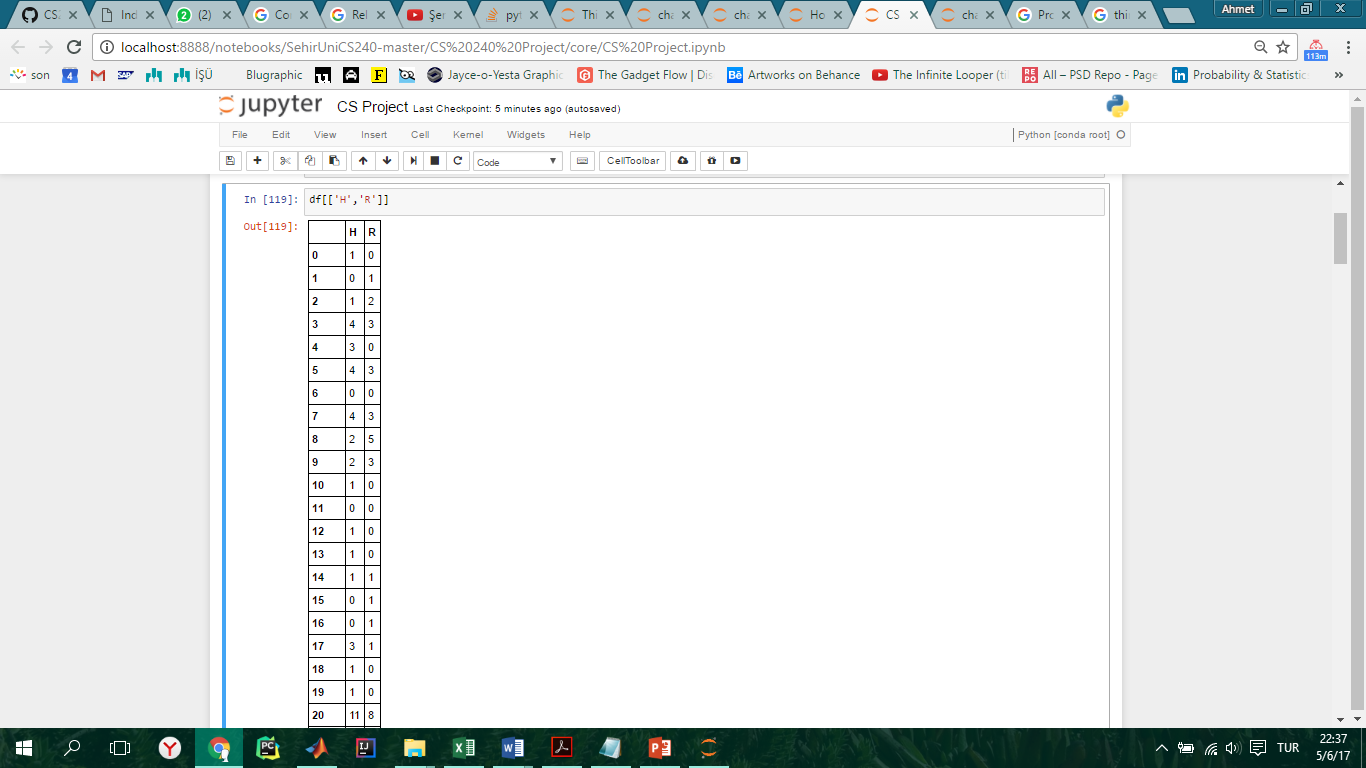
df = pd.read\_csv('../core/BattingPost.csv')

hits = df['H'].dropna()

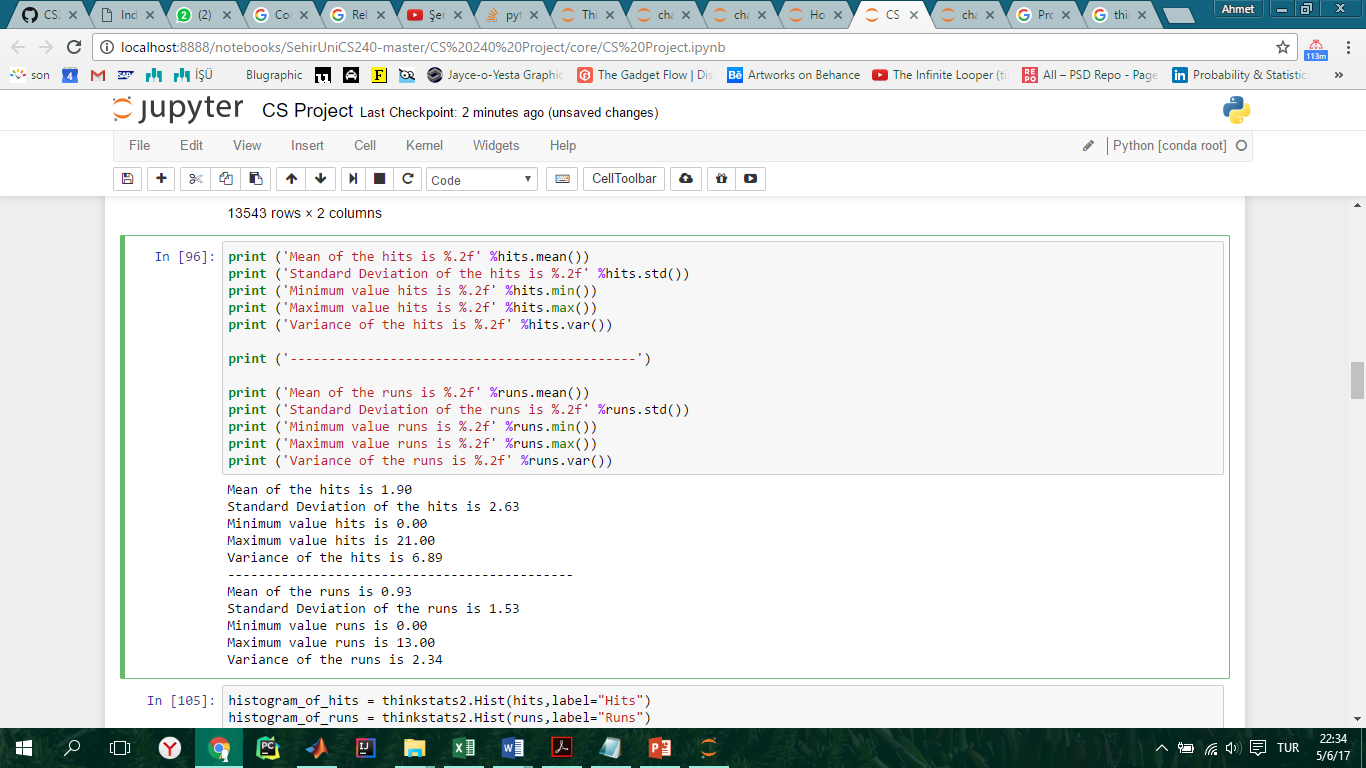
I have imported the “Batting.csv” by using the Pandas module. I called hits and run columns. With .dropna(), I cleaned the empty cells.

runs= df['R'].dropna()

**Part 3:**

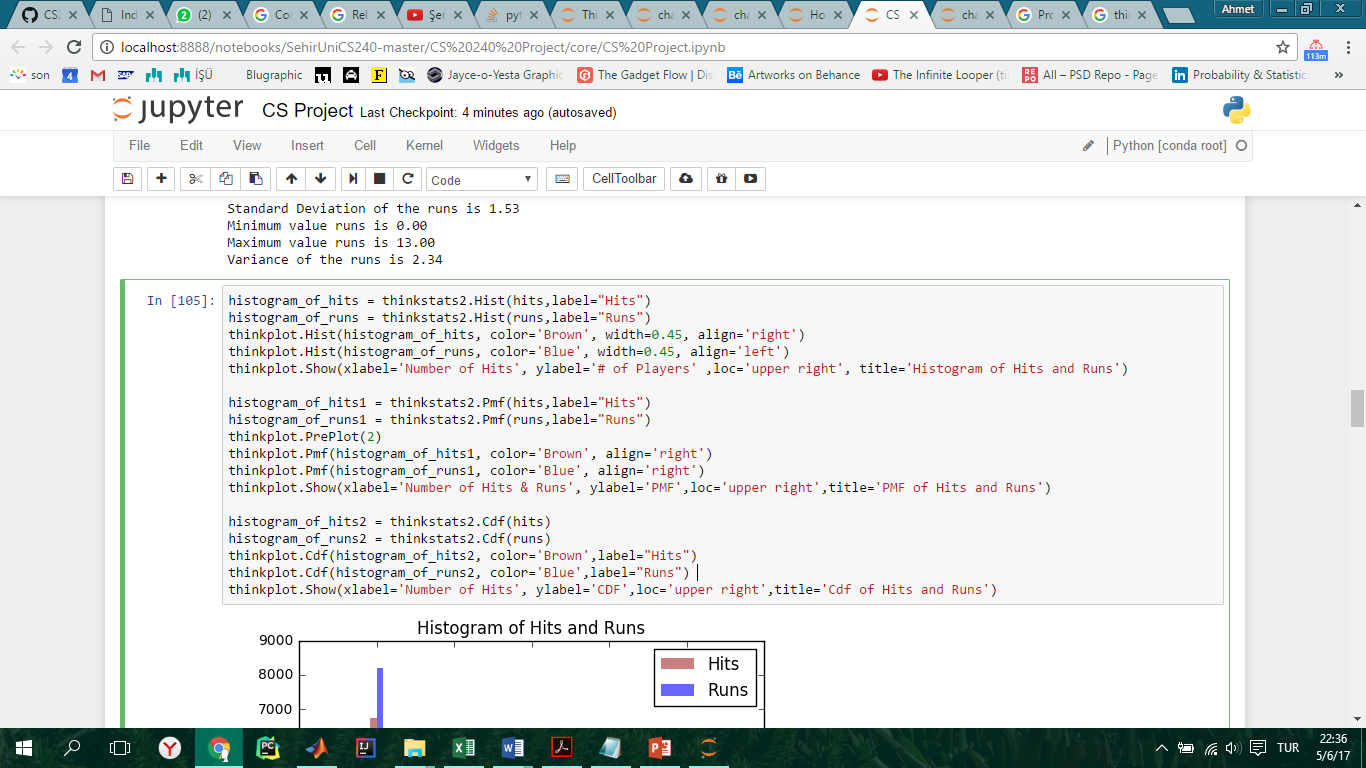
When we look detailed into the columns of hits and runs, I first checked some statistics of hits and runs.

I wanted to observe the mean, standard deviation, maximum values, minimum values, and variance. Here are the results:



I calculate the mean, I used “.mean ()”, for standard deviation “. std ()”, for minimum value “.min ()”, for maximum value “.max()”, and lastly for variance “.var ()” functions.

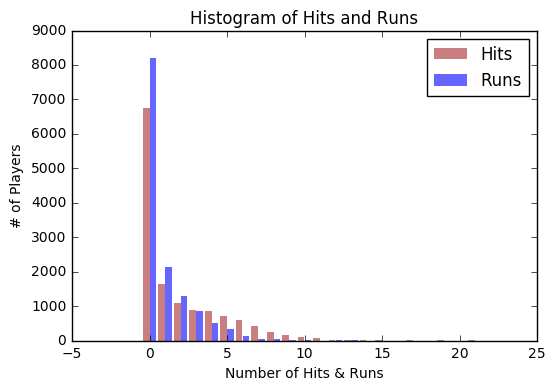
So as you can see on the results on above, we can say in the beginning that mean, standard deviation, and variance of the hits are slightly larger than the runs’.

Here, by using built in functions I found mean, standard deviation, maximum values, minimum values, and variance. Then I checked the Histograms, Pmfs, and Cdfs of both columns hits and runs. Here are the results.

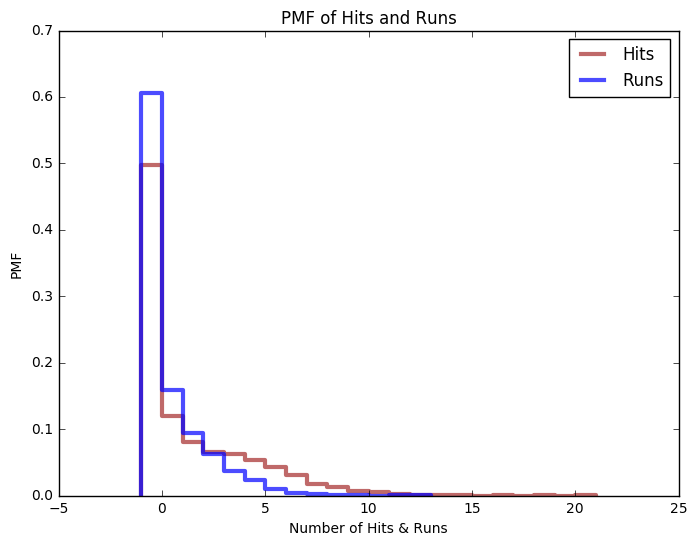
Here in the code to make the Histogram I used “thinkstats.2Hist ()”, to make the Pmf “thinkstats2.Pmf ()”, and to create Cdf “thinkstats2.Cdf ()”.

Of course to plot the Histograms, Pmfs, and Cdfs, we have to use thinkplot module. So here I plotted the histogram by using “thinkplot.Hist ()”, for Pmf “thinkplot.Pmf ()”, and lastly for Cdf “thinkplot.Cdf ()”.

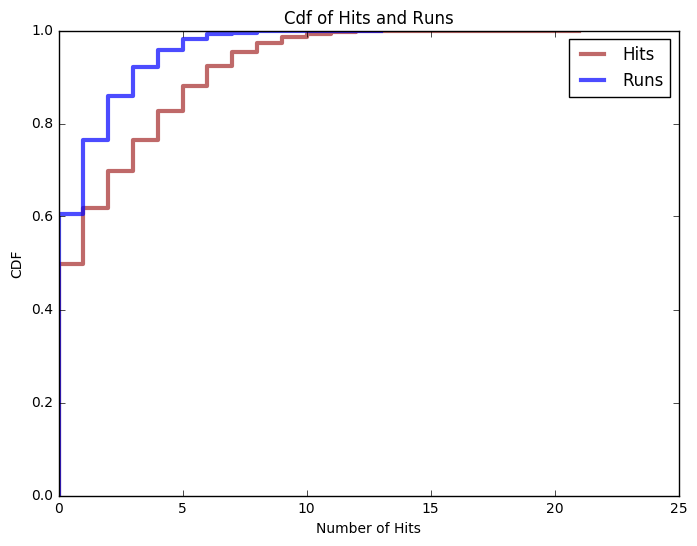
At the end of each function I made it appear by using “thinkplot.Show ()” function and entered parameters inside such as “xlabel” which enabled us to name the x axis, “ylabel” which enabled us to name the y axis, “loc” which let us place the label within the table, and lastly “title” which let us name the table.

Here are the results of the code;

This histogram shows how many players could able to run and able to hit. The x axis shows the number of hits and runs, and y axis shows how many players are there.



This Probability Mass Function (PMF) shows how many players could able to run and able to hit. The x axis shows the number of hits and runs, and y axis shows how many players are there.

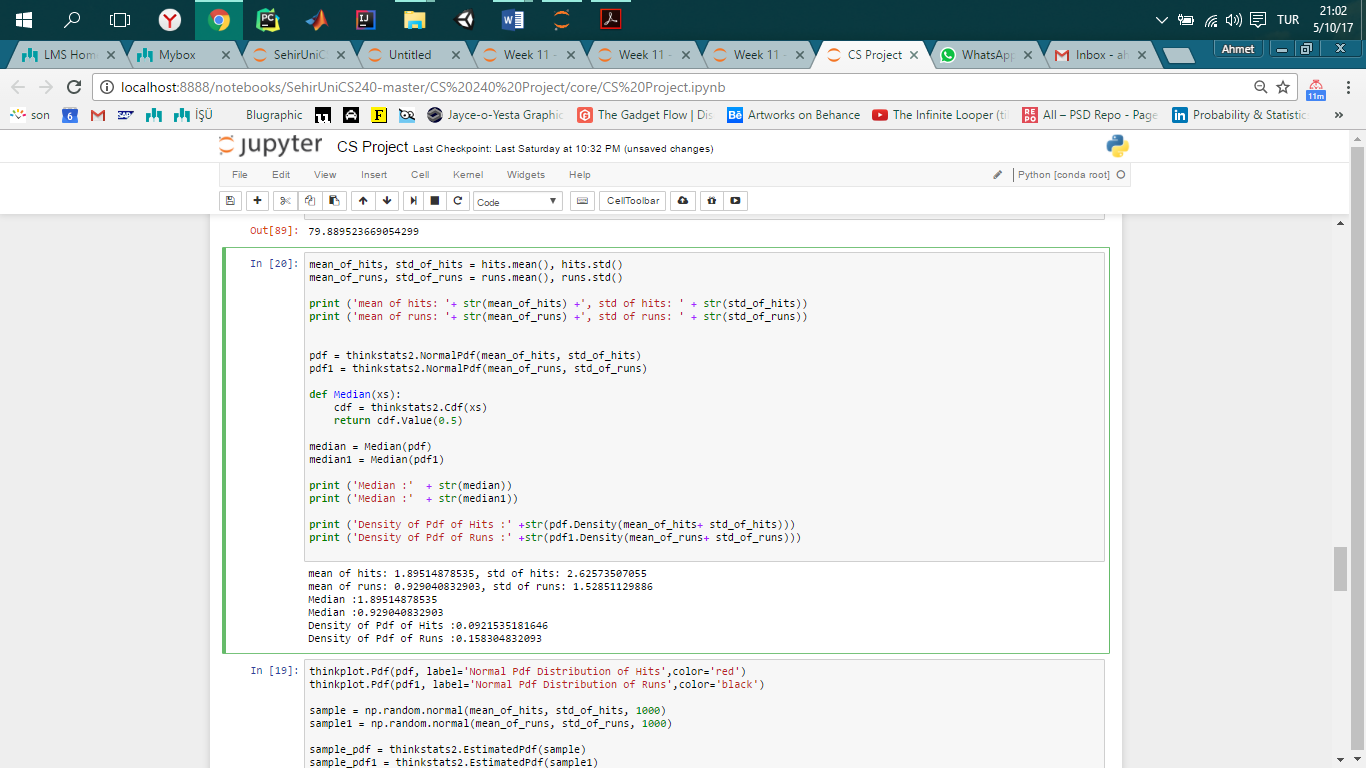


This Cumulative Distribution Function (CDF) shows how many players could able to run and able to hit in percentile. The x axis shows the number of hits and runs, and y axis shows the probabilities in of players in percentiles.

Here we can observe from all the tables that number of players who hit and run zero times is very high. So from the tables we can still say that there is a high relationship between hits and runs. Because number of hits and runs approximately decrease at similar times.

**Part 4:**

For the distribution, I used Normal Pdf distribution in order to see the density of the distribution at a location one standard deviation from the mean. By using Pdf, I also estimated the Kernel Density Estimation (KDE) which is an algorithm that takes a sample and finds an appropriately smooth PDF that fits the data.

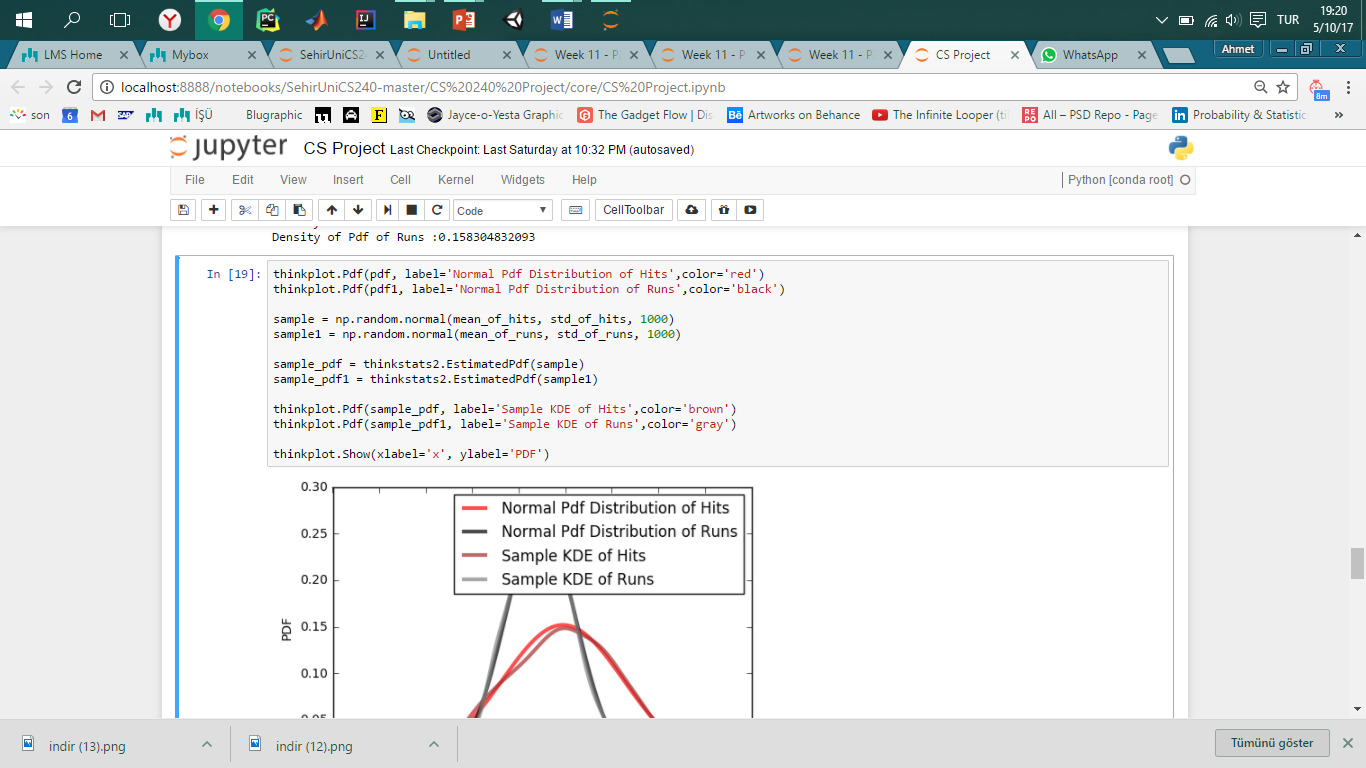


In this code, I again first calculated the mean and standard deviation by usin “.mean ()” and “.std ()” respectively because pdf distribution is drawn based on the mean and standard deviation.

I then used “thinkstats2.NormalPdf ()” and inserted the “mean\_of\_hits” and “std\_of\_hits” variables as parameters and created a Normal Pdf variable called “pdf”. Then I did the same thing for the runs and used “thinkstats2.NormalPdf ()” and inserted the “mean\_of\_runs” and “std\_of\_runs” variables as parameters and created a Normal Pdf variable called “pdf1”.

After creating Pdf distributions of runs and hits I found the medians of both distribution by using the function called “Median ()” and inserted “pdf” and “pdf1” as parameters.

At last I found the densities of but hits and runs by using “.Density ()” function and and inserted the means and stardard deviations.

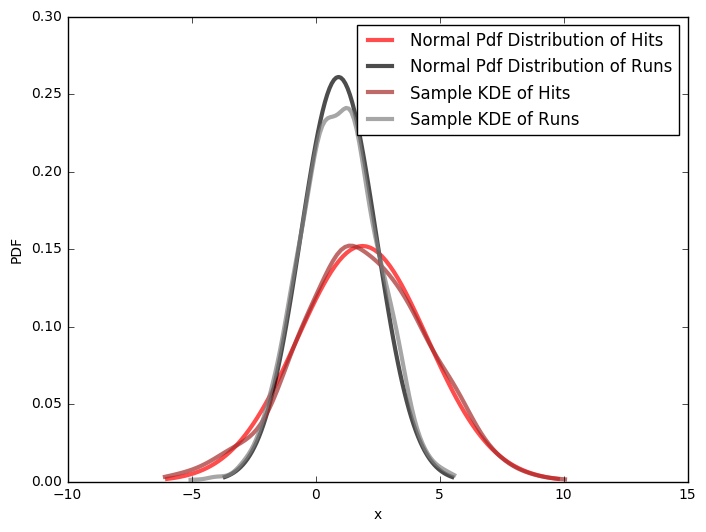
In this part of code, I drew the Normal Pdf distributions by usin “thinkplot.Pdf” functions and for each line I setted a different color.

As I said in the beginning, I found Kernel Density Estimation but in order to do that I had to randomomize the means and stardard deviation and iterate it too many times. So here by using “np.random.normal ()” I randomized it and by putting 1000, the pdf iterated 1000 times.

So then to create the distribution I used “thinkstats2.EstimatedPdf ()” and created KDE distribution. Later I plotted all the distributions by using “thinkplot.Pdf”.

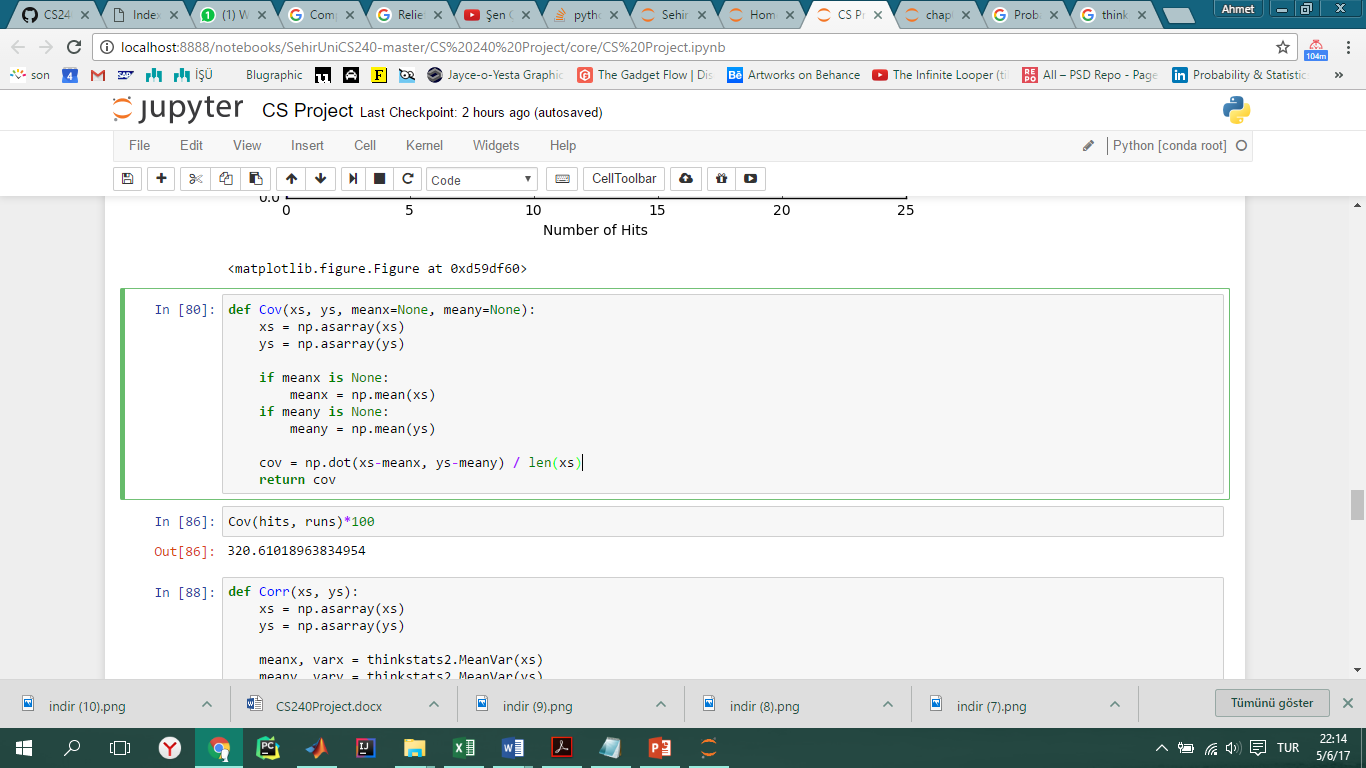
At last I showed everything on the table by using “thinkplotShow ()” and settted “xlabel” which shows the probability and “ylabel”.

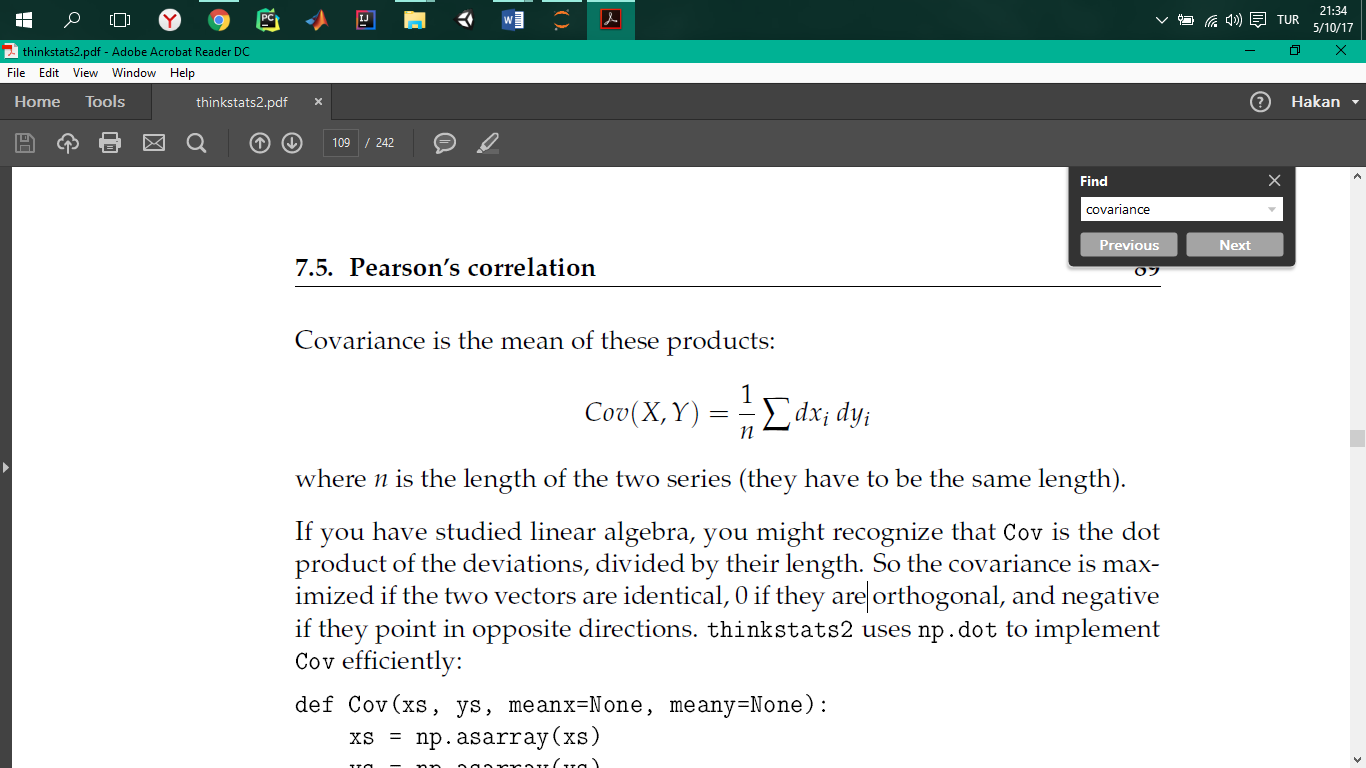
And here is the result:



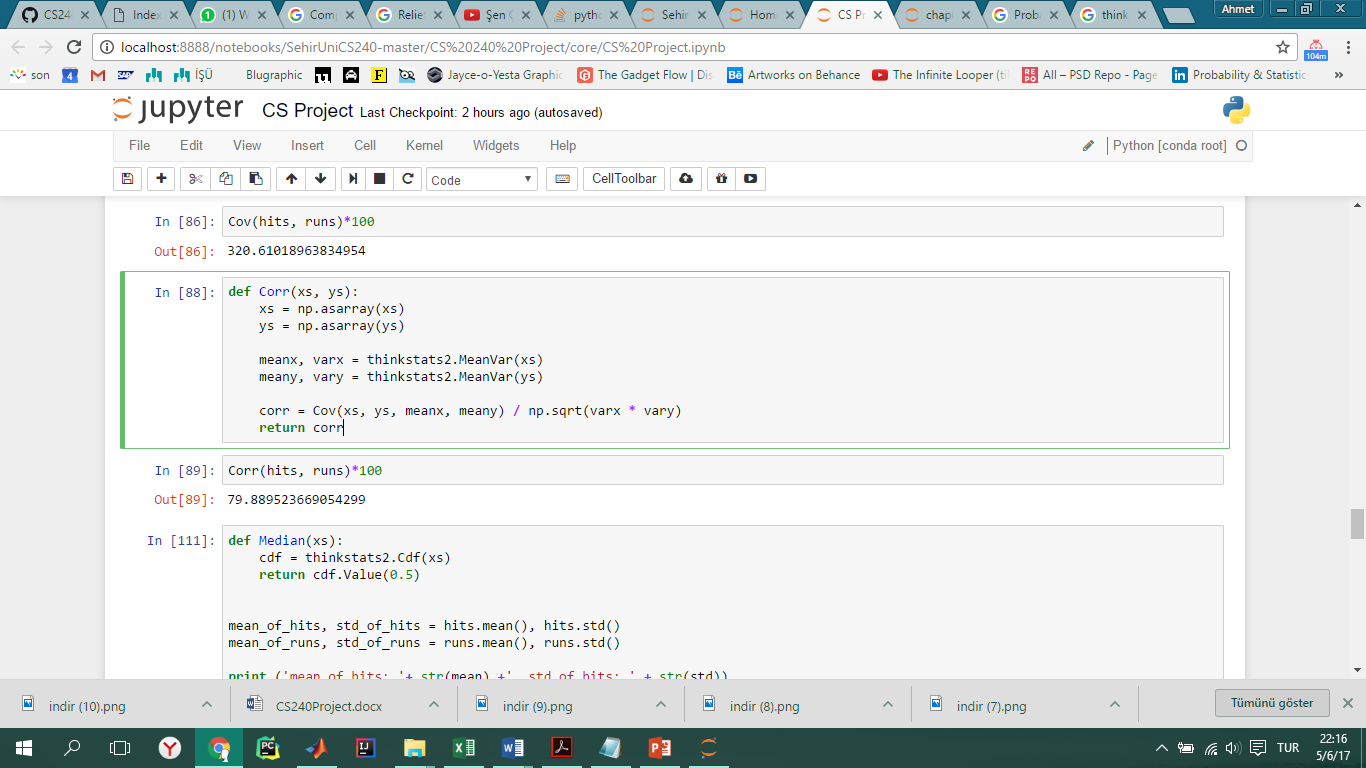
**Part 5:**

The question was by using “BattingPost.csv”, can we analyze the relationship with hits and runs? If the player can hit, does it mean they can also run? Is there a positive relationship or negative relationship? In order to answer this question we will have to use the covariance test which will give us the tendency of two variables to vary together and correlation test which will give us the strength of the relationship between two relationships.

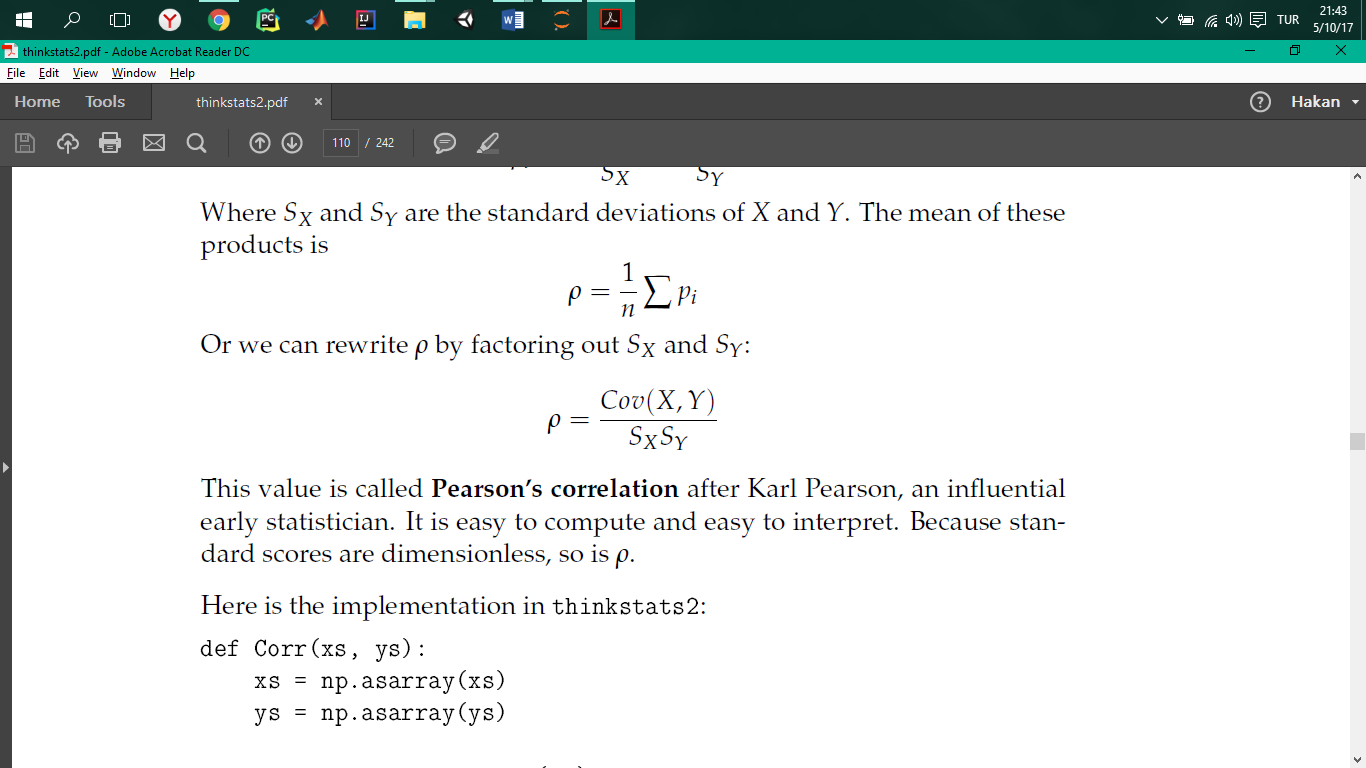
By default “Cov” computes deviations from the sample means, or you can provide known means. If “xs” and “ys” are Python sequences, “np.asarray” converts them to NumPy arrays. If they are already NumPy arrays, “np.asarray” does nothing.



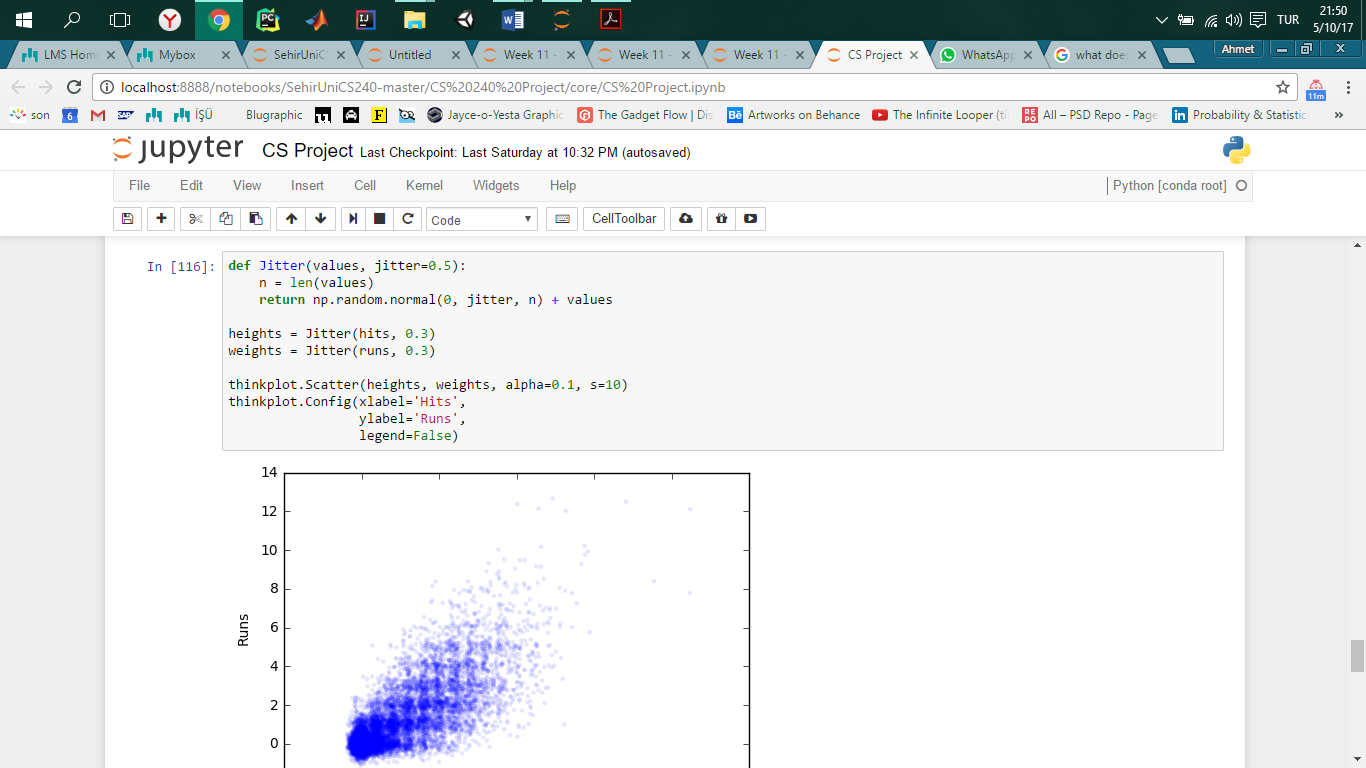
Here I tried covairance test by using “Cov ()” function. By entering parameters as “hits” and “runs”, I got the covariance between hits and runs.

At the end I got 320.61, which is pretty high and it means that assets returns to move together. So we can say that there is no much of difference by just checking the covarince. But we also have to check the correlation in order to conclude. So here is the Pearson’s correlation code:

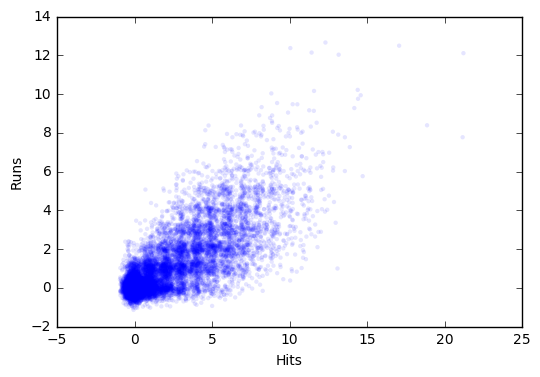
“MeanVar” computes mean and variance slightly more efficiently than separate calls to “np.mean” and “np.var”.



Pearson’s correlation is always between -1 and +1 (including both). If *r* is positive, we say that the correlation is positive, which means that when one variable is high, the other tends to be high. If *r* is negative, the correlation is negative, so when one variable is high, the other is low.

The reason we multiply with 100 is because then we get the percentage of the correlation. Here the correlation is 79.88 which means that there is a high and positive relationship between hits and runs. Most of the times we can say that if the players can’t hit the ball then they also cannot run during the game. Let’s visualize the relationship by using scatter plot.

In the beginning I started with adding jitter which is a random noise added to the variable. Then I plotted by “thinkplot. Scatter ()” and added the parameter “heights” and “weights” which have the hits and runs that are jittered by 0.3.



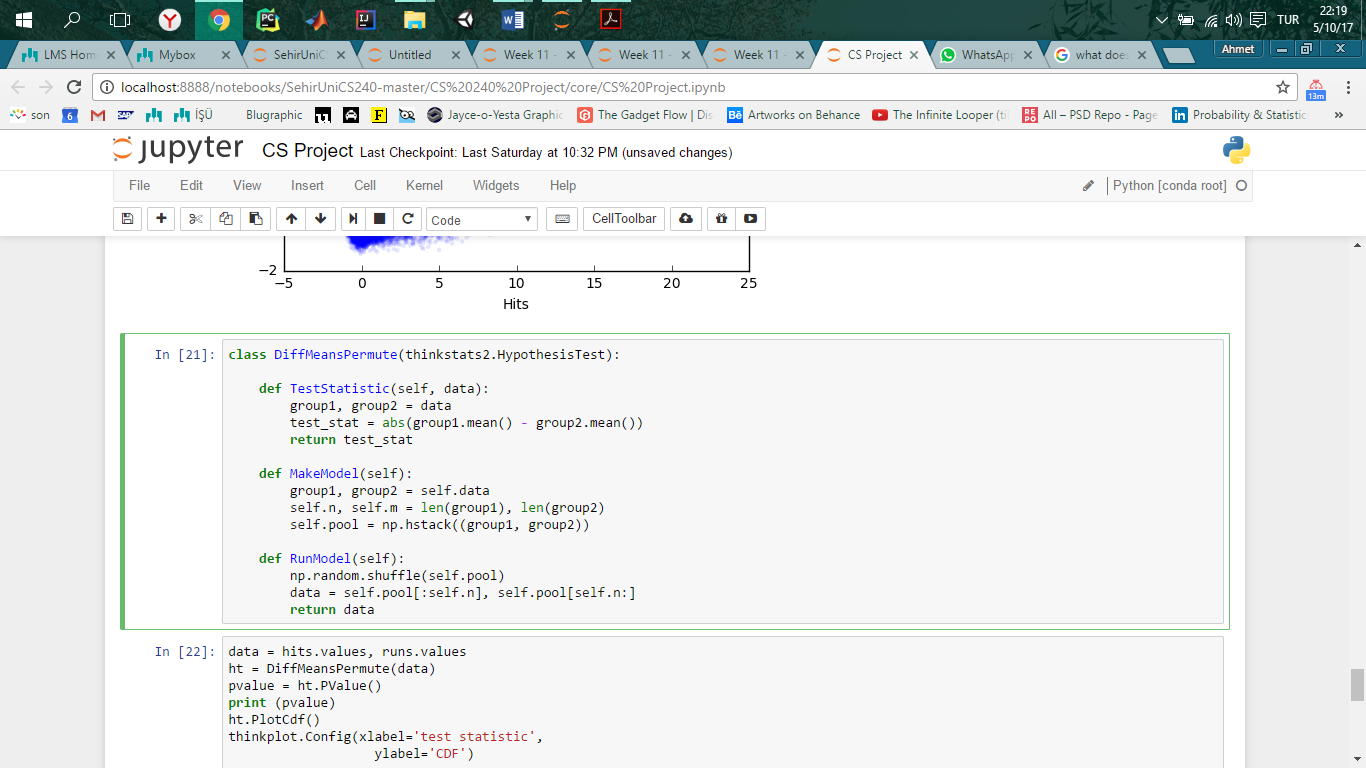
Hereby by checking the red circle drawn on the crowd, we can say that hits and runs have high similarity around 0. So when someone can’t hit they also can’t run.

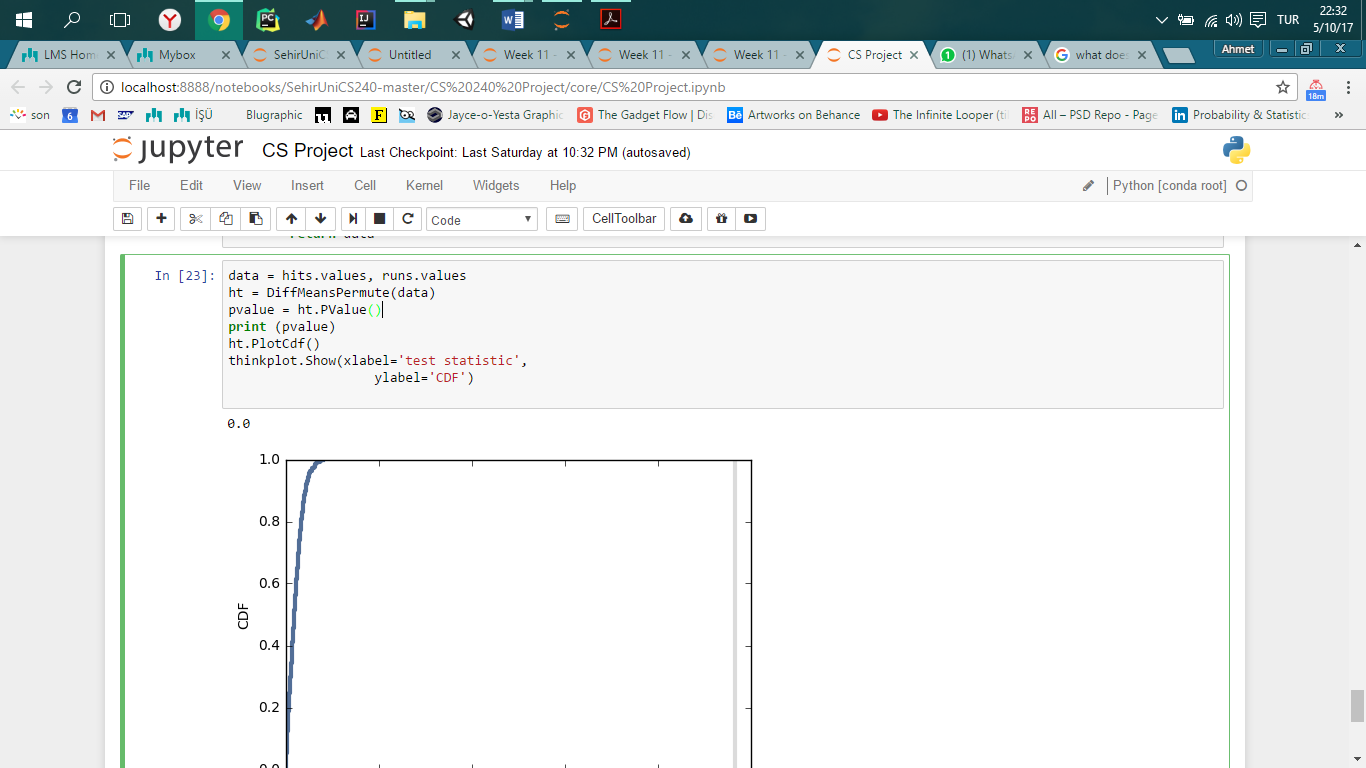
**Part 6:**

In this part, I will apply a hypothesis test to answer the question. In “thinkstats2” module there is already built in function called “HypothesisTest” and by using this we can apply hypothesis testing.

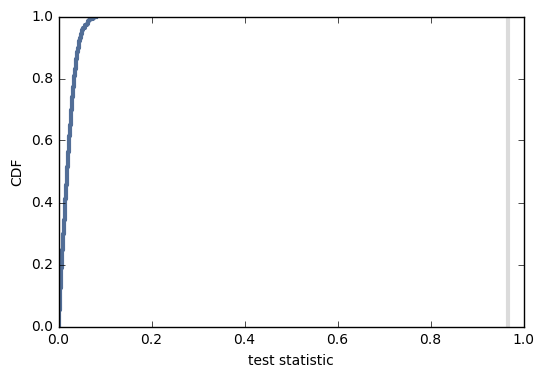
**Test Statistics :** There is a high relationship between hitting the ball and running in the field. If the player hits the ball then they can also run

**Null Hypothesis:** There is no relationship between hits and runs.

* “data” is a pair of sequences, one for each group. The test statistic is the absolute difference in the means.
* “MakeModel” records the sizes of the groups, n and m, and combines the groups into one NumPy array, “self. Pool”.
* “RunModel” simulates the null hypothesis by shuffling the pooled values and splitting them into two groups with sizes n and m. As always, the return value from RunModel has the same format as the observed data.



As we can see here **p-value is 0**, so this means that it is **statically significant.**



**Part 7:**

As a conclusion, I can say that there is a high similarity between hits and runs. When we observe the Histograms, Pmfs, and Cdfs the values are plotted almost on the same places. Later when we check the covariance and correlation we can conclude that the similarity is pretty high. At the end when we ran the Hypothesis Test, we saw that out test statistic is statically significant.

So the answer of the second question is;

There is high positive relationship between hits and runs.

**References:**

Think Stats: Probability and Statistics for Programmers