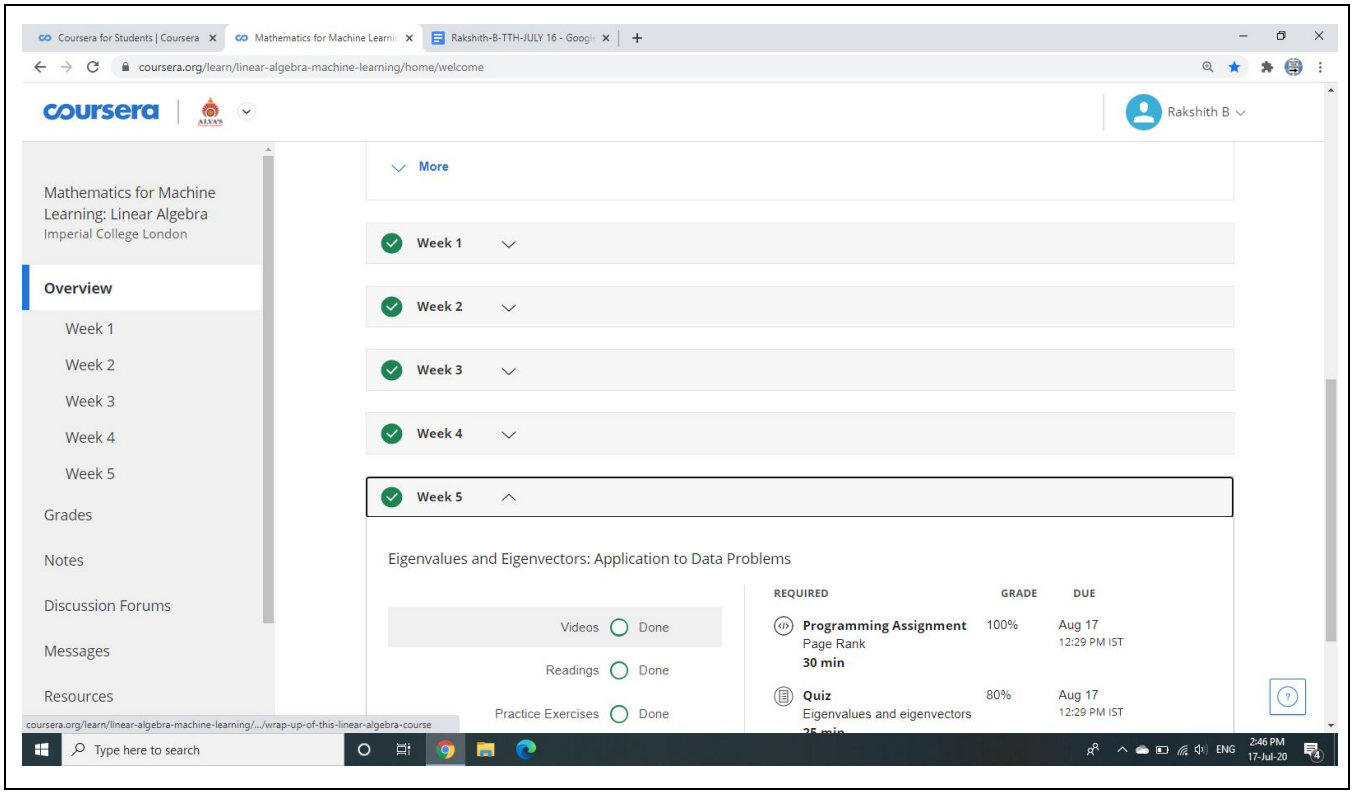
REPORT JULY 17

Date:	17 JULY 2020	Name:	Rakshith B
Course:	Coursera	USN:	4AL16EC409
Topic:	Mathematics for Machine Learning	Semester & Section:	6th SEM B
Github Repository:	Rakshith-B		

Image of the Session



That's it for this course on Linear Algebra in our specialization on Mathematics for Machine Learning. Our goal was to give you some of the underpinnings of Linear Algebra in order to enable you to access Neural Networks, Machine Learning, and Data Science courses more generally. This is intended as a course for a social scientist, engineer, or physicist to develop the insight required to access other courses on Machine Learning Tools and in order to do useful work to solve problems with Machine Learning in the real world. It's not intended as a foundation for a graduate Linear Algebra course and nor we have the time to discuss topics like the Cross Product that don't have applications in Machine Learning. We started out in the course by thinking of some data problems, fitting functions to data, or the apples and bananas price discovery problem, which we use to take us on a journey into Linear Algebra. First, vectors and then matrices. Through it all, you did exercises to try out what you'd learned and then took those into a couple of programming exercise in Python at the end. This would have given you the confidence to say that you're really getting towards having an intuitive understanding of Linear Algebra and how to apply it in code. If you've enjoyed this course, we continue the journey in Machine Learning and Data Science in the next course in this specialization, Multivariate Calculus. There, we look at Calculus and how to find the gradients of functions of more than one variable. This will let us examine Minimization and how to fit models to data. Once we have Minimization and Linear Algebra and Probability, you'll have all the underpinning

This brings us to the end of the fifth module and also, to the end of this course on linear algebra for machine learning.

We've covered a lot of ground in the past five modules, but I hope that we've managed to balance, the speed with the level of detail to ensure that you've stayed with us throughout.

There is a tension at the heart of mathematics teaching in the computer age. Classical teaching approaches focused around working through lots of examples by hand without much emphasis on building intuition. However, computers now do nearly all of the calculation work for us, and it's not typical for the methods appropriate to hand calculation to be the same as those employed by a computer. This can mean that, despite doing lots of work, students can come away from a classical education missing both the detailed view of the computational methods, but also the high level view of what each method is really doing. The concepts that you've been exposed to over the last five modules cover the core of linear algebra. That you will need as you progress your study of machine learning. And we hope that at the very least, when you get stuck in the future, you'll know the appropriate language. So that you can quickly look up some help when you need it. Which, after all, is the most important skill of a professional coder. [SOUND]

build an expression that tells us, based on this network structure, which of these webpages is most relevant to the person who made the search. As such, we're going to use the concept of Procrastinating Pat who is an imaginary person who goes on the Internet and just randomly click links to avoid doing their work. By mapping all the possible links, we can build a model to estimate the amount of time we would expect Pat to spend on each webpage. We can describe the links present on page A as a vector, where each row is either a one or a zero based on whether there is a link to the corresponding page. And then normalise the vector by the total number of the links, such that they can be used to describe a probability for that page. For example, the vector of links from page A will be 0 1 1 1. Because vector A has links to sites B, to C, and to D, but it doesn't have a link to itself. Also, because there are three links in this page in total, we would normalize by a factor of a third. So the total click probability sums to one. So we can write, L_A = (0, a third, a third, a third). Following the same logic, the link vectors in the next two sites are shown here. And finally, for page D, we can write L_D is going to equal, so D is connected to B and C, but not A, two sides in total, (0, a half, a half, 0). We can now build our link matrix L by using each of our linked vectors as a column, which you can see will form a square matrix. What we're trying to represent here with our matrix L is the probability of ending up on each of the pages. For example, the only way to get to A is by being at B. So you then need to know the probability of being at B, which you could've got to from either A or D. As you can see, this problem is self-referential, as the ranks on all the pages depend on all the others. Although we built our matrix from columns of outward links, we can see that the rows describe inward links normalized with respect to their page of origin.