

**Big Data Content Analytics**

**Multiclass Classification Problem with Convolutional Neural Networks**

Dimitra Afrati: F2821801

Harper Bush: F2821804

Nikolatou Maria: F2821821

Table Of Contents

[**Abstract**](#_i8jt05bia4fm) **3**

[**Introduction**](#_yt60zkttvnht) **4**

[Convolutional Neural Networks](#_yvh78ym8dv4e) 4

[How Convolutional Neural Networks work:](#_85ov2db5oxrw) 5

[Convolutional layer](#_wvrhpipkka1b) 5

[Kernel](#_dg4jzf69vpdy) 6

[Pooling Layer](#_1tqmtcjmgguf) 6

[Activation functions](#_syvdro3ubmem) 6

[ReLU function](#_whuzcsseiayv) 6

[Sigmoid function](#_vejf8gheopmm) 7

[**Data Preparation**](#_lcyriljw053v) **7**

[The Initial Dataset](#_4djnvnq55t3r) 7

[One hot encoding](#_o1golurjtshc) 8

[The Final Dataset](#_5e6c89ze0trb) 8

[**Modeling**](#_qgvz4cume4m7) **9**

[**Results and Predictions**](#_cybxgben0eq2) **11**

[Confusion Matrix](#_pj54xrfbskb9) 12

[Sample Predictions:](#_txhonyp1e9e2) 14

[Image #25, category: Travel](#_i1fpju9yjhj2) 14

[Image #20 , category: Romance](#_8v7oprgbd7iz) 15

[Image #3, category: Business](#_etot8lp8ukun) 15

# Abstract

This paper presents a multiclass classification problem of book covers images. Each book is assigned to one and only one label. Goal of this paper is to identify the most well-fitted model to predict the category that a book belongs given only its book cover. The categories are referred to the type of content each book addresses. The [multiclass classification](https://www.sciencedirect.com/topics/computer-science/multiclass-classification) problem was reduced to a set of [binary classification](https://www.sciencedirect.com/topics/computer-science/binary-classification) problems. Using Convolutional Neural Networks, a Deep Learning algorithm which reading an input image, assign learnable weights and biases to various objects in the image, a model was constructed for multiclass prediction.

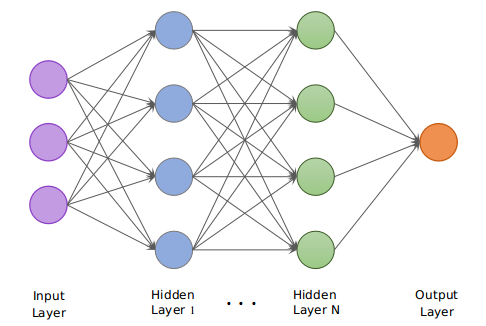
# 

# 

# Introduction

## Convolutional Neural Networks

A Convolutional Neural Network (CNN) is a type of Artificial Neural Network (ANN), mostly popular for analysing images. Like the human brain consists of interconnected neurons that constantly transmit signals, an artificial neural network also has interconnected artificial neurons that transmit data among each other. As shown in image 1, a typical neural network has an input layer, an output layer and various hidden layers. The input layer receives and input, while the output layer gives out the predictions of the model. The layers are fully connected to the layers before and after them. Each node in the network consists of certain random weights and each layers has a bias attached to it that influence the output of every node. Certain activation functions are applied to each layer to decide which nodes to fire.



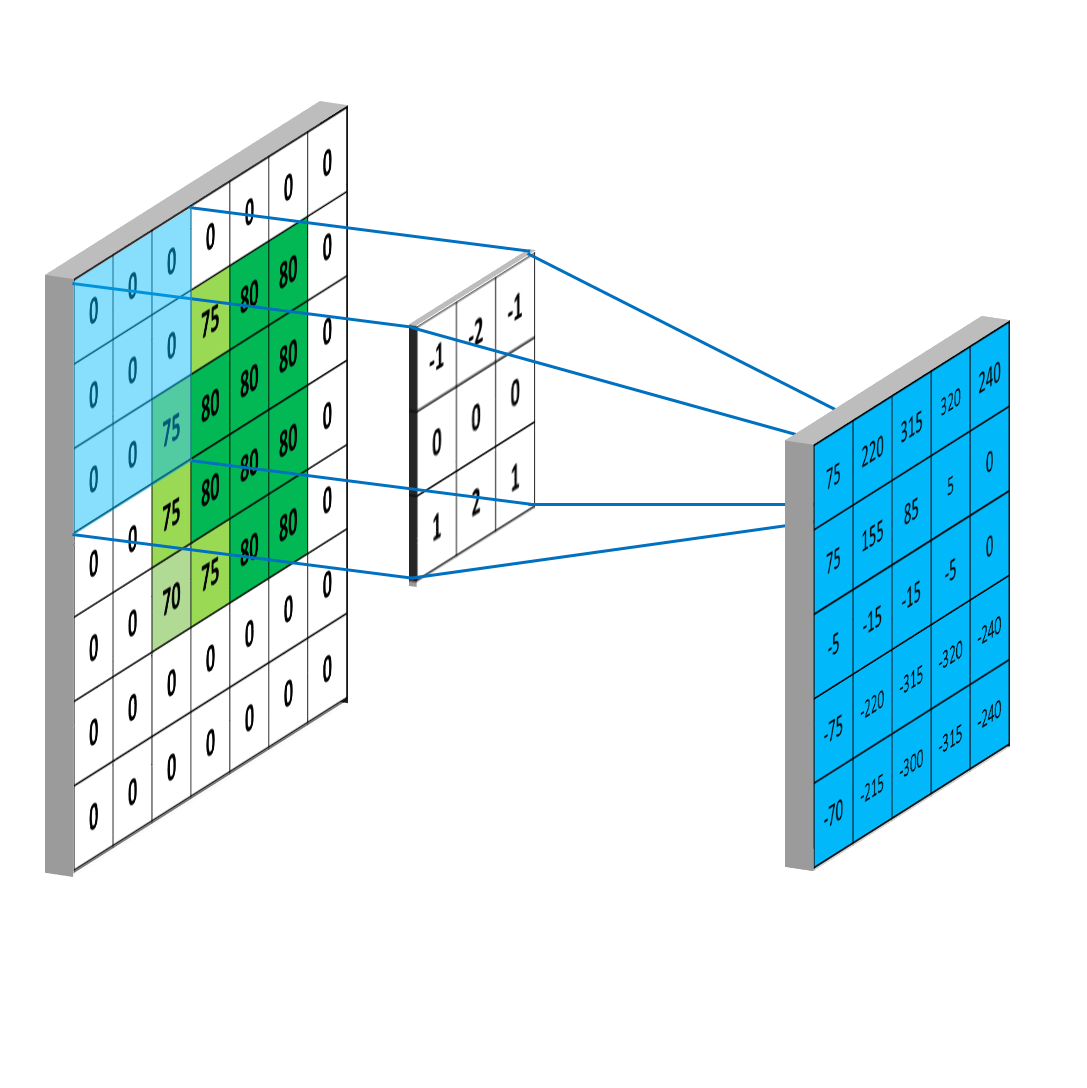
*Image 1*

On the other hand, Convolutional Networks, apart from fully connected layers, include other types of layers, commonly convolutional and pooling layers. Convolutional neural networks are inspired by the visual cortex system, in humans and other animals. As the name suggests, this layer applies the convolution with a learnable filter and as a result the network learns the patterns in the images: edges, corners, arcs, then more complex figures. They can detect patterns and make sense of them. This is the main reason that are useful for image analysis.

## How Convolutional Neural Networks work:

### Convolutional layer

Same as Regular Neural Networks, CNN take as an input an image, which is stored as a two dimensional array of pixels. The dimensions of each array are the image resolution. Each pixel of the image stores its resolution ranging from 0 to 255. RGB images store 3 channels, one for green, one for red, and one for blue. However, regular neural networks can not identify a certain object in different places of the image, for example if an object appears in the bottom left corner of an image it will not be recognised by the Neural Network. CNN consider the context/shared information in the small neighborhoods, meaning that they match parts of the image, instead of the whole image.



*Image 2*

#### 

#### Kernel

The element that is involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter and it selects an initial matrix. For example, in image 2 it selects a K as a 3x3x1 matrix. The Kernel shifts, every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering. The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed. Finally, all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output. The first Convolutional Layer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

### Pooling Layer

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Pooling layers consider a block of input data and simply pass on the maximum value. Doing this reduces the size of the output and requires no added parameters to learn, so pooling layers are often used to regulate the size of the network and keep the system below a computational limit.

### Activation functions

Activation function is a node that is put at the end of or in between Neural Networks. In an artificial neural network, the activation function defines the output of the neuron based on a set of inputs. The purpose is to make our output non-linear. In the current project ReLU and Sigmoid functions were used.

#### ReLU function

The ReLU (Rectified Linear Unit) function is one of the most widely used activation functions. It transforms the input to the maximum of either zero or the input itself. If the input is less than or equal to zero, then the activation will transform this input to zero. If the input is greater than zero, then the activation will output the input itself. In the first case the neuron will not activated, while in the second case it will be activated.

#### Sigmoid function

Sigmoid function exists between 0 to 1. Therefore, it is especially used for models that have to predict the probability as an output. Since probability of anything exists only between the range of 0 and 1, sigmoid is the right choice.

# Data Preparation

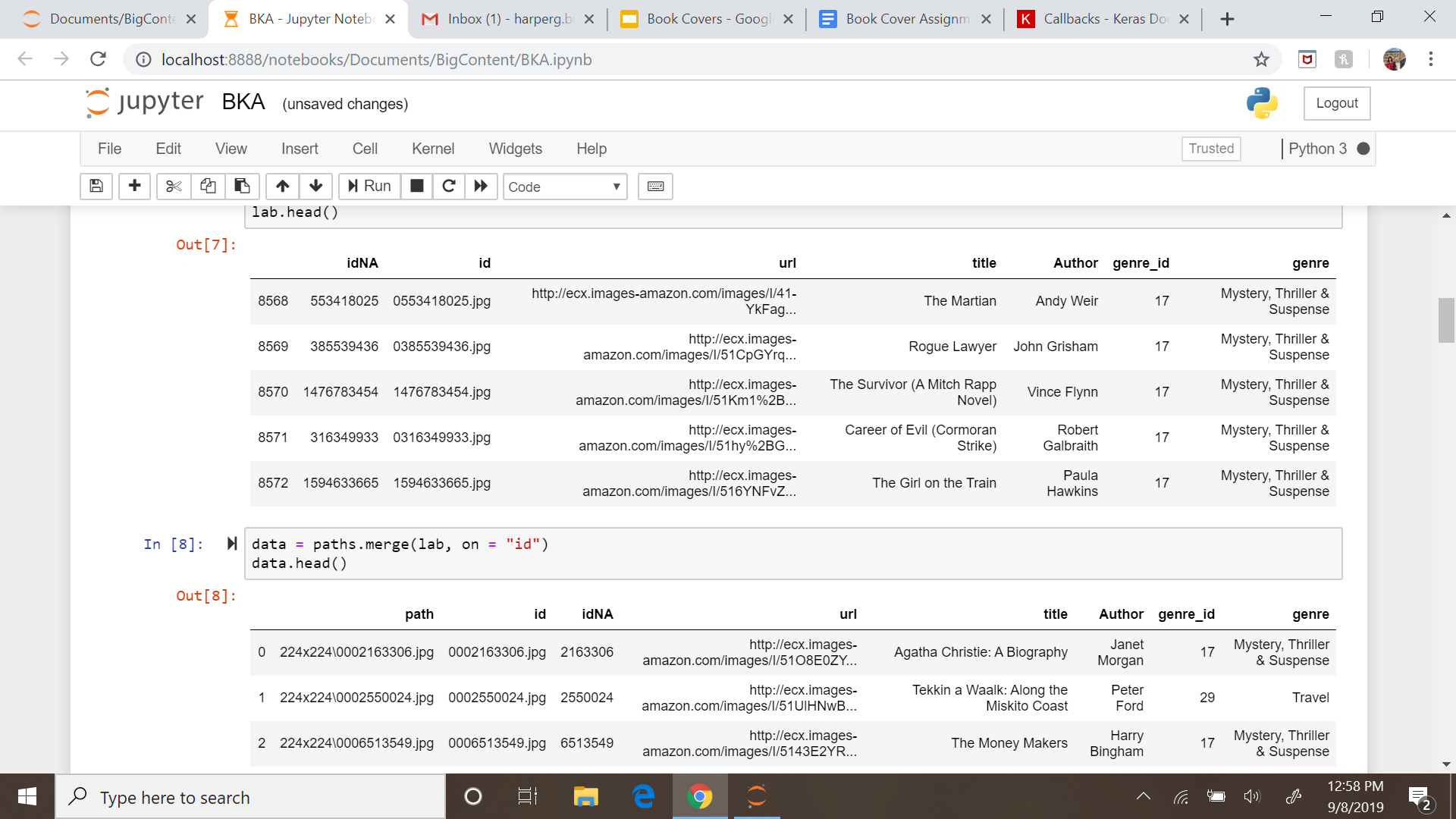
Dimitra Afrati- ETL

## The Initial Dataset

The dataset used for the project was obtained from the website below:

<https://github.com/uchidalab/book-dataset>

The initial dataset consists of book cover images. It contains 57000 colored images of size of size 224 x 224, all stored in one single folder, in JPG format. The file that contains the labels of the books, was then imported in a variable called *labs*. It consists of 6 columns: the image ID, the url of the actual image, the title of the book, the author, the genre ID and the book genre.



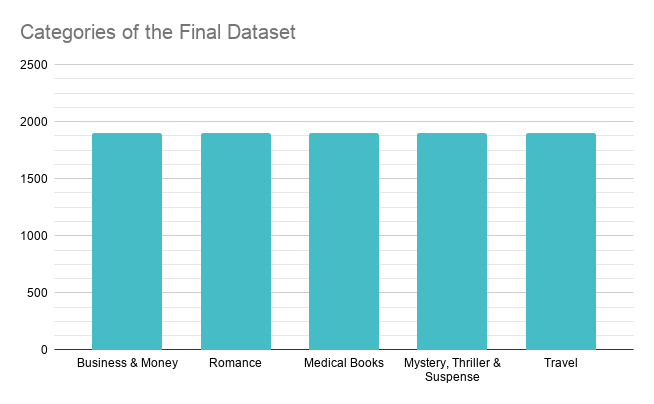
In order to match the images with the corresponding labels the following procedure was performed: The paths of all the images of the initial dataset were stored in a variable named *paths*. Using a function in python, the last part of each image path, that is also the ID of each image, was stored in a second column of the data frame with the column name ‘ID’. Then the dataframe that contains the path names was merged with the dataset that contains the label information, on the column ID of the image. Due to memory errors, only 5 categories were kept from the initial dataset. As you can see in plot 1, those categories were equally split and consist of 1900 rows each.

### One hot encoding

Sometimes it is easier to deal with binary data, rather than categorical ones. This transformation can be done using *One hot encoding*. In the initial labels dataset, the category was stored in one column with the integer value of the ID category. By using one hot encoding the categories can be split into different columns, while the integer values are replaced by binary ones. Thus, each book in this dataset, instead of having one column of integer type that describes the category ID, it now contains the values 0 or 1, depending on if it belongs to the category of the corresponding column.

## The Final Dataset

Finally, using *imread* and the *map* function the images of the dataset were imported and matched with the corresponding labels. The final dataset consists of 9500 images of 5 different categories.



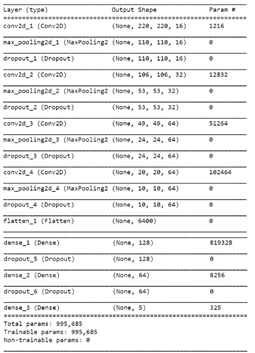
*Plot 1*

# Modeling

Harper Bush - Data Analyst

Because we are attempting to classify images into a set of categories, we found it best to use a Convolutional Neural Network sequential model. Below is the code of the layers of the first model we used. This is referred to in the code as “model”.

This model’s parameters are used frequently in classic examples for CNNs and in image classification. However, a few specific inputs were required in order for it to work with the book cover images. The first was in the first Conv2D layer.

We want 16 filters, a kernel size of (5,5), relu activation, and an input shape of (224,224,3). It is common practice to start models off with a lower number of filters, then increase by doubling that number as we go. 16 is somewhat small to start for a CNN.

The kernel size hyperparameter will determine the size of the area that the network will look at at one time. We chose a size of 5x5. That is considered larger than most. It will look at larger factors of the image rather than tiny details.

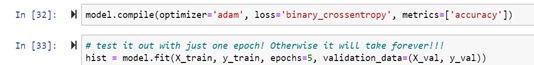
Our activation function is ReLU even though our results will be binary using the one hot encoded categories. This function is widely accepted as the best function to have within the layers of most models because it looks like a linear function, but performs much better than one. In our case, it can identify true zeros, not just exponentially decreasing towards zero as sigmoid does.

One of the most important parameters to be sure to set ourselves is the input shape, because this must match the shape of the images in our dataset: 224x224 RGB (224,224,3). Otherwise, the network won’t run.

To prevent overfitting, we add in a dropout layer, which we have set at .25. Some layer units/neurons will be ‘dropped’ or randomly ignored when running through the data. Our value is somewhat low in comparison to the most popular choices for optimization (somewhere around .5). Another additional layer we place in the model is the max pooling layer. Basically, it takes the highest number from the existing groups of neurons and puts them in 2x2 bins. This decreases dimensionality and allows the model to take a look in separate aspects of the image at once.

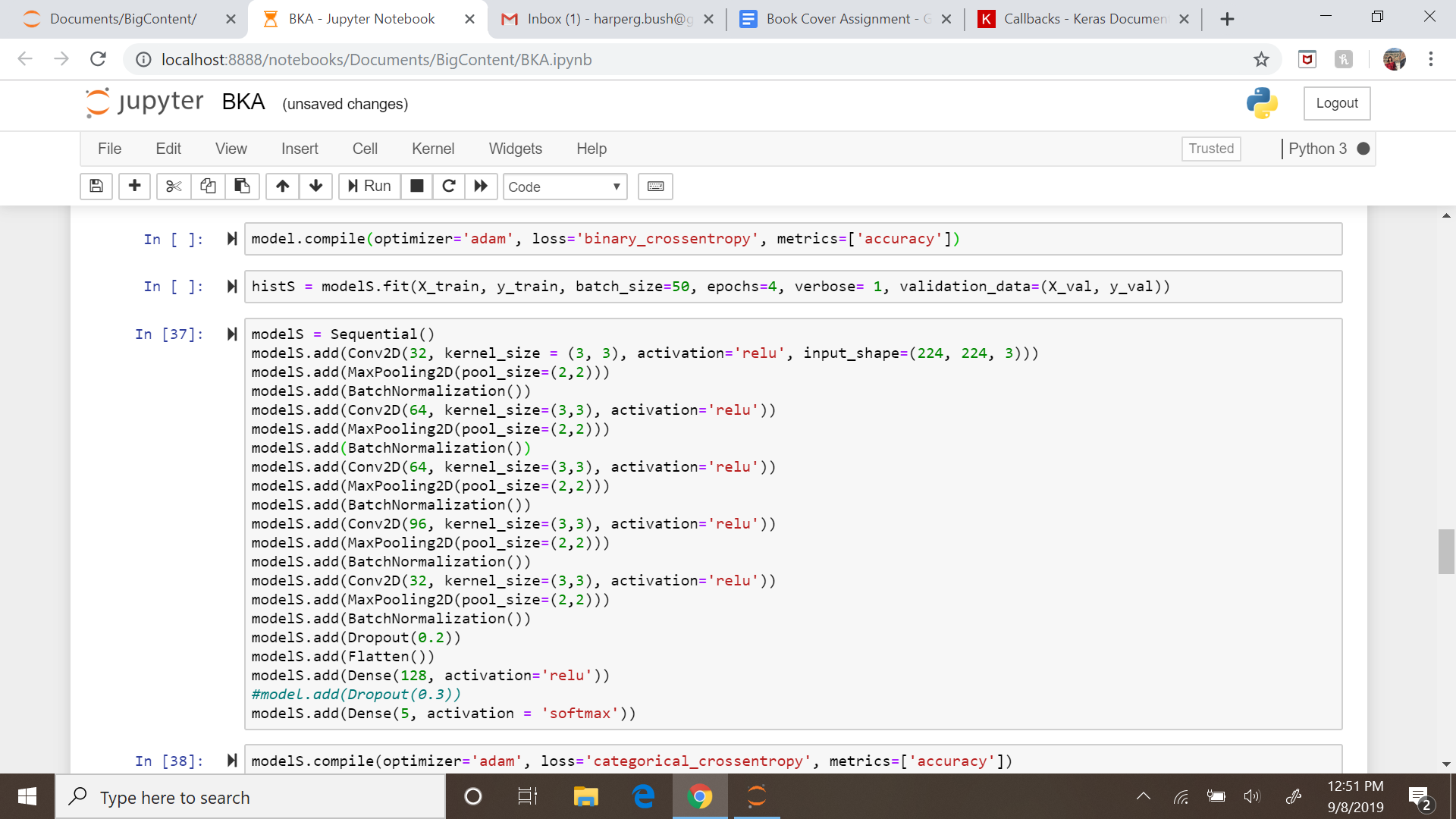
Our model contains multiple iterations of these layers, but the final layer is our fully connected ‘dense’ layer. This is flattened to a single dimension, and it is important to specify the number of categories we have in this part (5 categories). In addition, we have used the sigmoid activation function in this final section because this is the format we want our results to be returned in.

Below is a line of code for compiling the model. We define our optimizer, loss identifier, and metric for evaluating our results. These results will be explained after running the model. The adam optimizer is a popular optimizer because it works like stochastic gradient descent, but is more efficient and works better with each additional layer. Binary crossentropy is the generally accepted loss method for our image classification models. Finally, accuracy will tell us the percentage of correct prediction our model reports. We want low loss and high accuracy.



In addition, we have the model.fit command, which will begin running the model. We chose 5 epochs to reduce the time for computation (which takes between 5 and 10 minutes per epoch).

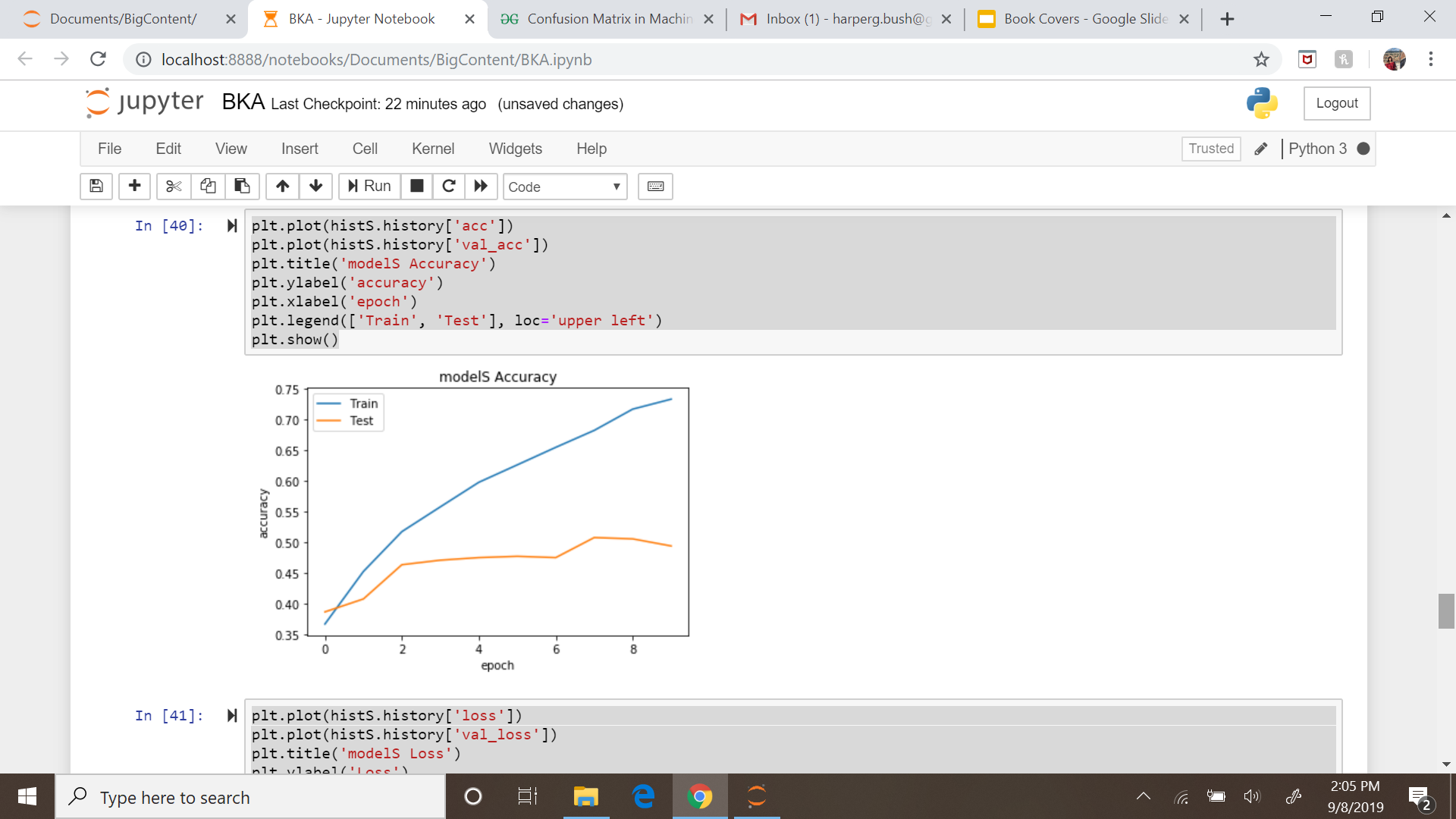
Because hyperparameters must be tested and adjusted to improve performance, we created a second model to compare with our first and named it “modelS”. The code for building this model is below.

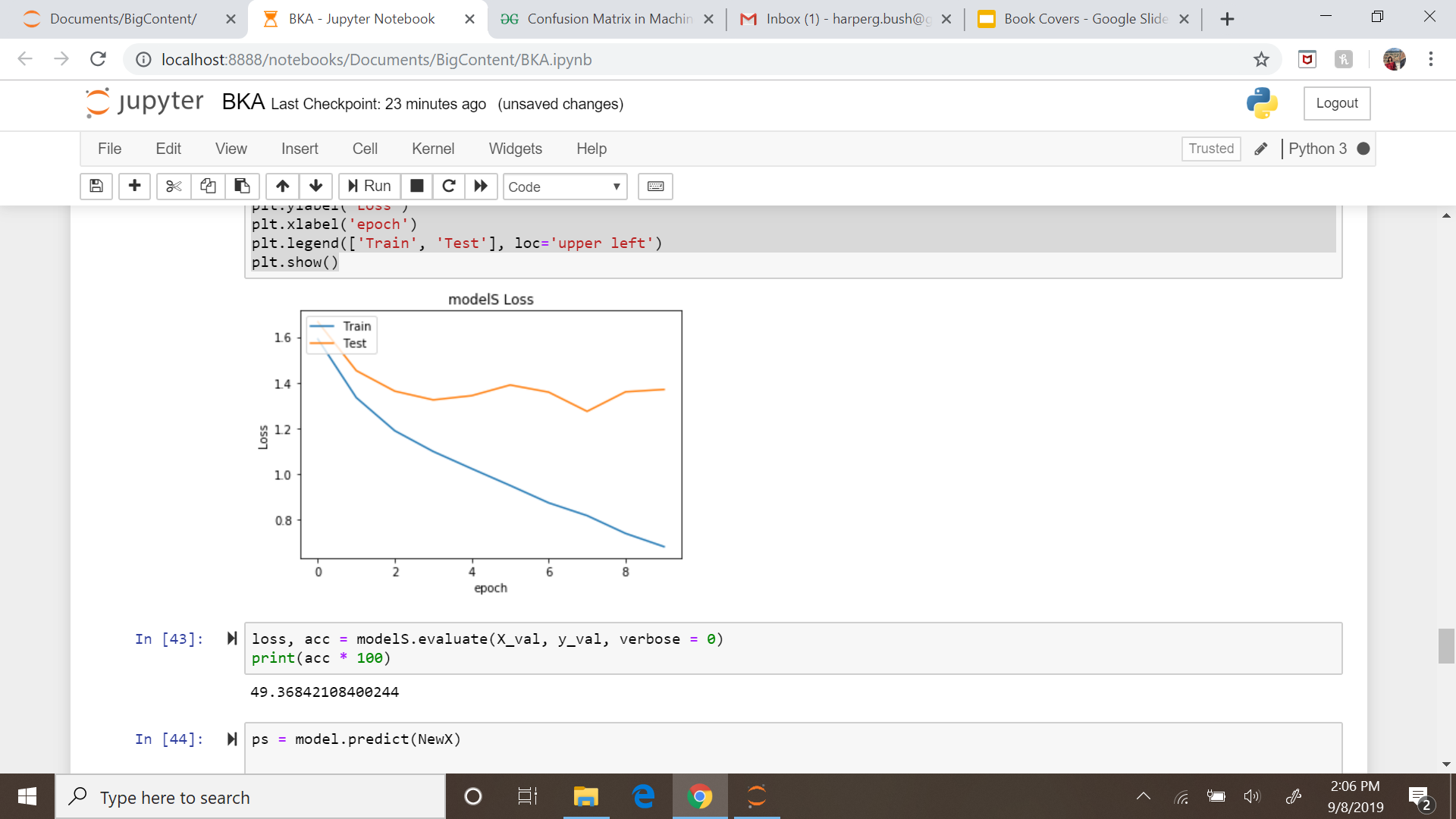


There are a few differences between modelS and the first model. These are the number of filters, the kernel size, and the final dense activation function. Because our images are somewhat small, we thought a smaller kernel size might detect differences between genres better than a larger one. It was reduced from 5,5 to 3,3. Additionally, we changed the initial filter size from 16 to 32, which may improve our accuracy scores. Subsequently, each of the filter sizes in the other convolutional layers have doubled in size.

The most important difference is in the dense layer activation function. We have changed it to the softmax function, which is typically used in such multi category classification. Choosing this option can fix some prediction errors we were seeing from the previous mode. In addition, the loss metric will now be categorical cross entropy instead of binary. Our results will be returned in a new format using this method.

Last, the plots of the accuracy and loss are presented during the training part.





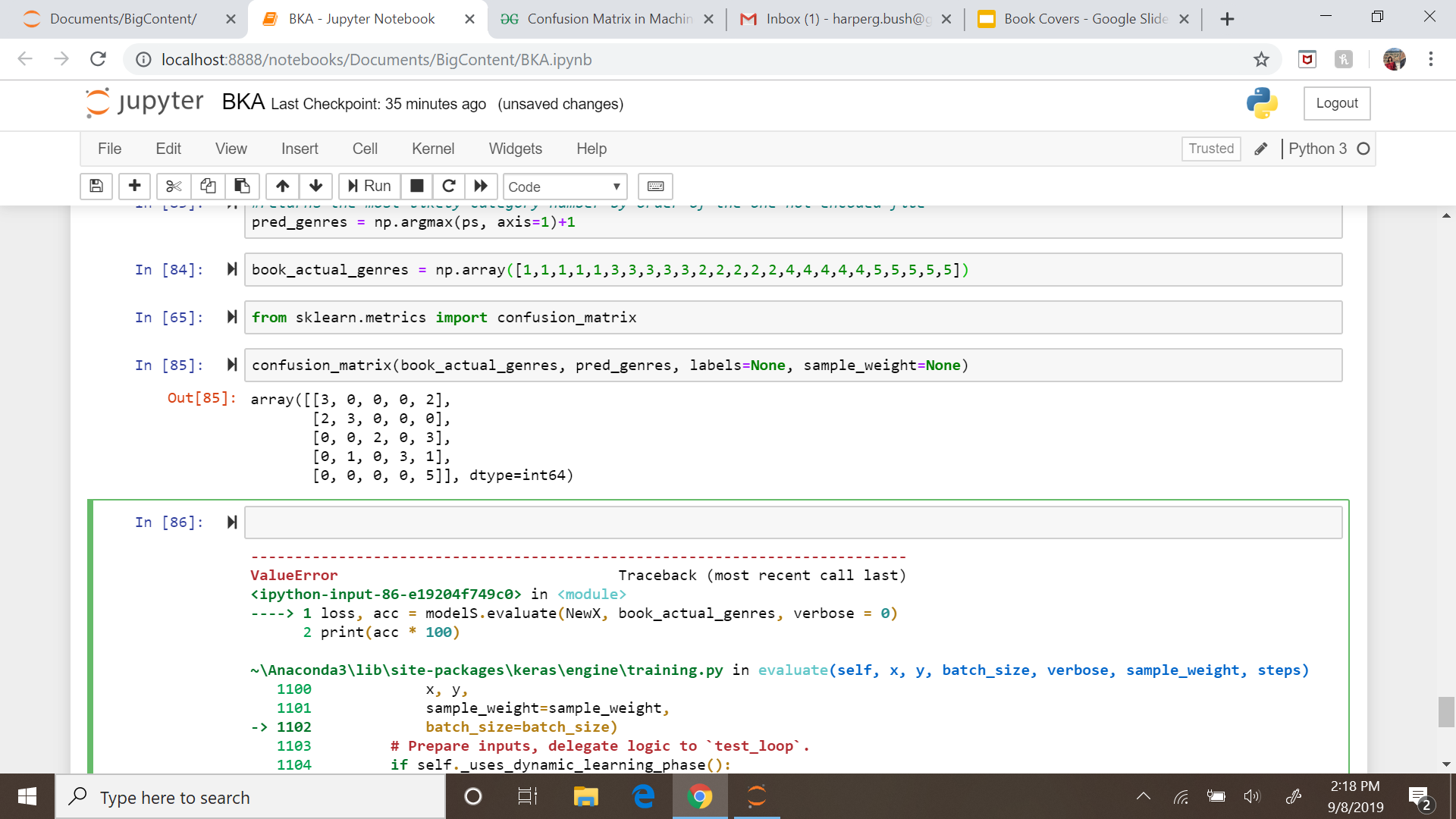
# Results and Predictions

Nikolatou Maria- Business Analyst

After the training was performed, the model was used for predicting the categories of books given book covers photographed by us. Some categories like Medical Books which we did not have in our possession, were found online. Each category was represented by 5 images. Therefore, the total size of the test data was 25 book covers. In order to resize the images into 224\*224 we used an online image editor (<https://www.imgonline.com.ua/eng/resize-image-result.php>).The evaluation of the model is based upon the accuracy metric and the confusion matrix. The accuracy metric indicates how often the classifier is correct. In the confusion matrix, similar metrics to accuracy were presented and were taken into consideration.

## Confusion Matrix

The multiclass classifier was evaluated from the above confusion matrix which shows the classification of the predicted values in regard with the actual values. In the rows they are presented the actual labels and in the columns the predicted labels. The first position in the matrix is referred to Business & Money, second to Medical Books, third to Mystery, Thriller, fourth to Romance and fifth to Travel.



According to the confusion table, the model predicted that:

* 5 of the books belong to Business & Money
* 4 of the books belong to Medical Books:
* 2 of the books belong to Mystery, Thriller:
* 3 of the books belong to Romance:
* 11 of the books belong to Travel:

Based on that, the following claims can be obtained:

* The Accuracy of the model, which is referred to how often the classifier is correct overall, is 60% (15/25)
* The category with the higher occurrence in the predictions is the category Travel.
* The categories Business & Money and Medical Books were classified pretty well.
* Next, we check the True Positive Rate (Sensitivity/Recall) which corresponds to when the image belongs to a specific category how often the model predicts it (TP/(TP+FN)).

For Business & Money: 0.6

For Medical Books: 0.4

For Mystery, Thriller: 0.6

For Romance: 0.6

For Travel: 1

* Last, Precision is referred to when the model predicts that the image belongs to a specific category how often is it correct. (TP/(TP+FP))

For Business & Money: 0.6

For Medical Books: 0.75

For Mystery, Thriller: 1

For Romance: 1

For Travel: 0.45

The Accuracy metric is needed to be evaluated in regard with the No Information Rate which in this case is 0.2. The No information Rate is the possibility to classify correct a new book cover by randomness. Therefore, if no model is constructed and all the new images will be assigned to a random category among these 5, there is a 20% (1/5) probability to classify them correctly. To summarize, for a model to be well-fitted in the problem should have higher accuracy than the No Information Rate. In this case, the selected model which was presented in the previous chapter, produces an accuracy of 60%. Therefore, by using the model we increase the probability to make a correct classification of the book covers by 40%. This percentage can of course be increased with providing more book cover images in the training part and experiment more in the architecture of the model. and Last, some sample prediction of the test data was presented.

## Sample Predictions:

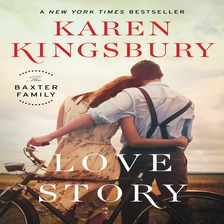
#### Image #25, category: Travel



Prediction: Travel

[1.26505062e-01, 2.50732508e-02, 7.85804093e-02, 5.03052957e-03, 7.64810741e-01]]

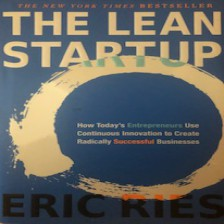
#### Image #20 , category: Romance



Prediction: Romance

[1.71138759e-04, 1.26438317e-04, 2.76215702e-01, 7.23481596e-01, 5.09960591e-06]

#### Image #3, category: Business



Prediction: Travel (incorrect)

[1.70963794e-01, 4.83707264e-02, 2.59321153e-01, 1.56107679e-01, 3.65236670e-01]

**Sources/Links:**

Useful Tutorials

<https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html>

<https://www.analyticsvidhya.com/blog/2019/04/build-first-multi-label-image-classification-model-python/>

<https://keras.io/models/sequential/>

Image Transformation tool

<https://www.imgonline.com.ua/eng/resize-image-result.php>

Datasets Used

<https://www.kaggle.com/astaroth88/book-cover-image-dataset>