**JABALPUR ENGINEERING COLLEGE, JABALPUR**

GOKALPUR, JABALPUR – 482011



*Major Report on*

**“ Emoji Classifier ”**

*Submitted in partial fulfilment of the requirements for the award of degree of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

Submitted by:

**TENZIN RABTEN 0201CS151087**

**ANTARA DAS 0201IP151016**

**VIII SEMESTER**

Submitted to:

**Prof. SHIWANGI MISHRA**

**CSE Dept.,**

**JEC, JABALPUR.**

**JABALPUR ENGINEERING COLLEGE,**

**GOKALPUR, JABALPUR. 482011**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

DECLARATION



We hereby declare that the project entitled “Emoji Classifier” which is being submitted in partial fulfillment of the requirement for award of the Degree of Bachelor of Engineering in Computer Science and Engineering to Jabalpur Engineering College, Jabalpur (M.P.) is an authentic record of our own work done under the guidance of Prof. Shiwangi Mishra, Department Computer Science & Engineering, Jabalpur Engineering College, Jabalpur.

The matter reported in this Project has not been submitted earlier for the award of any other degree.

CERTIFICATE



This is to certify that the Major Project entitled “Emoji Classifier” is being successfully completed by Tenzin Rabten & Antara Das.

This project is being submitted for their partial fulfillment of the requirement for award of the Degree of Bachelor of Engineering in Computer Science and Engineering from Jabalpur Engineering College, Jabalpur.

Prof. Shiwangi Mishra Dr. Shailja Shukla

Project Guide Head of Department

Computer Science & Engineering Dept.

CERTIFICATE



This is to certify that the Major Project entitled ”Emoji Classifier” is being successfully completed by Tenzin Rabten & Antara Das.

This project is being submitted for their partial fulfillment of the requirement for award of the Degree of Bachelor of Engineering in Computer Science and Engineering from Jabalpur Engineering College, Jabalpur.

Internal Examiner External Examiner

ACKNOWLEDGEMENT

We sincerely express indebtedness to esteemed and revered guide in department of CSE for her invaluable guidance, supervision and encouragement throughout the work. Without his kind patronage and guidance, the project would not have taken shape.

We would like to express our sincere regards to her for advice and counseling from time to time.

ABSTRACT

The recognition and classification of the diversity of materials that exist in the environment around us are a key visual competence that computer vision systems focus on in recent years. Understanding the identification of materials in distinct images involves a deep process that has made usage of the recent progress in neural networks which has brought the potential to train architectures to extract features for this challenging task. This project uses state-of-the-art Convolutional Neural Network (CNN) techniques in order to classify materials and analyze the results. Building on dataset collected, a selection of CNN architectures is evaluated to extract features in order to achieve results for the task. By limiting the amount of information extracted from the layer before the last fully connected layer, transfer learning aims at analyzing the contribution of shading information and reflectance to identify which main characteristics decide the material category the image belongs to. The results of the comparison emphasize the fact that the accuracy and performance of the system improves, especially if the datasets consist of a large number of images.

**CONTENTS**

Declaration…………………………………………………...

Certificate…………………………………………………….

Certificate…………………………………………………….

Acknowledgement…………………………………………..

Abstract………………………………………………………

1. Introduction…………………………………………………..

1.1 Problem Description…………………………………….

2.Team Formation…………………………………………….

Software Requirement Specification………………………..

3. Software Interfaces……………………………………….

4. Data Collection …………………………………………….

5. Methodology………………………………………………...

Classification Using Neural Network……………………..

6. Implementation………………………………………………

7. User Interface………………………………………………..

8. Result……………………………………………………….

9. Testing…………………………………………………...

10. Appendix…………………………………………………

8.1.1 Source Code Screenshot…………………….

8.1.2 References……………………………………...

1. **INTRODUCTION**

Emojis are ideograms and smileys used in electronic messages and web pages. Emoji exist in various genres, including facial expressions, common objects, places and types of weather, and animals. They are much like emoticons, but emoji are actual pictures instead of typographics.

Emojis can be used by any one that makes them incredible useful tools for analysis.

95% of internet users have sent an emoji at some point. Over 10 billion emojis are sent each day. But we still know very little about how the world uses emojis and what it all means.

Emoji faces provide a glimpse into the emotion and sentiment people have.

Emoji might now be the internet’s most popular language, which is why we’ve built Emoji Classifier.

1.1 PROBLEM DESCRIPTION

The problem herein is that we want to be able to identify whether the emoji we see on the Internet is positive, neutral, or negative.

We have decided to build software to perform several tasks in accomplishing this goal:

1. We need a way to "take a picture" or photograph something within our computer (i.e. something we see on the Internet). We could do this by using our phone or a screenshot software.
2. This picture should then be plopped into some machine-learning algorithm that will tell us, which if our emoji is looking positive, negative or somewhere in between.
3. **SOFTWARE PROCESS MODEL**

The system proposed above was designed using the incremental designing principles.

Incremental development is a process of partial implementation of the entire system and a slow build-up of functionality. This approach allows you to reduce the costs incurred before reaching the level of initial productivity. With the help of this model, the process of creating a functioning system is accelerated. This is facilitated by the applied principle of layout from standard blocks, through which it provides control over the process of developing changing requirements.

The incremental model operates on the principle of a cascade model with overlapping, so that the functionality of the product, suitable for operation, is formed earlier. This may require a complete pre-established set of requirements that are implemented in the form of consecutive, small-sized projects, or the project can begin with the formulation of common goals, which are then refined and implemented by the development teams.

The incremental model describes a process in which priority attention is paid to system requirements, and then to their implementation in development teams. As a rule, with time, the increments decrease and each time less requirements are realized. Each subsequent version of the system adds to the previous defined functionality until all the planned features are implemented. In this case, you can reduce costs, monitor the impact of changing requirements, and accelerate the creation of a functional system through the use of the standard block layout method.

**The Process**

Each increment then passes through the remaining phases of the life cycle: coding, testing, and delivery.

First, we design, test and implement a set of functions that form the basis of the product, or the requirements of paramount importance, which play the main role for the successful implementation of the project or reduce the degree of risk. The subsequent iterations extend to the core of the system, gradually improving its functionality or performance. The addition of functions is carried out by performing significant increments in order to fully satisfy the user’s needs. Each additional function is certified in accordance with a set of requirements.

Why the Iterative Model?

* You do not need to spend in advance the funds necessary for the development of the

entire project (since the development and implementation of the main function or function from the high-risk group is first carried out);

* As a result of each increment, a functional product is obtained;
* The customer has the opportunity to comment on each developed version of the system;
* The rule of “divide and conquer” allows you to break up the problem on managed parts, thereby preventing the formation of cumbersome lists of requirements put forward to the development team;
* There is an opportunity to maintain constant progress during the implementation of the project;
* Costs for the initial delivery of the software product are reduced;
* The initial delivery schedule is accelerated (which allows to meet the increased market requirements);
* The risk of failure and changing requirements is reduced;
* Customers can recognize the most important and useful functional capabilities of the product at earlier stages of development;
* The risk is divided into several smaller increments (not comprehensible in one large development project);
* Requirements are stabilized (by inclusion in the user process) at the time of the creation of a certain increment, since no significant changes are moved back to the time of the creation of subsequent increments;
* Incremental functionality is more useful and easier to test than middle-tier products with level-by-level “top-down”
* Improved understanding of the requirements for later increments (which is ensured by the user’s ability to get an idea of ​​the earlier received increments at a practical level);
* At the end of each incremental delivery, it is possible to review the risks associated with costs and compliance with the established schedule;
* The use of successive increments makes it possible to combine the experience gained by users in the form of an improved product, while at the same time spending much less money than is required to carry out a re-development;
* In the development process, we can limit the number of personnel in such a way that the same team works consistently on the delivery of each increment and all the teams involved in the development process did not stop working on the project (the schedule of the workforce distribution can be equalized by the time distribution of the amount of work on the project );
* The possibility to start building the next version of the project in the transitional phase of the previous version smooths out the changes caused by the personnel change;
* At the end of each incremental delivery, it is possible to review the risks associated with costs and compliance with the established schedule;
* Customer needs are better manageable, since the development time of each increment is very small;
* Because the transition from the present to the future does not happen instantly, the customer can get used to the new technology gradually;
* Significant signs of progress in the implementation of the project help to maintain a “pressure” at the managed level caused by compliance with the schedule.

1. **TEAM FORMATION**

Team structure addresses the issue of organization of the individual project teams. There are some possible ways in which the individual project teams can be organized. On the basis of our knowledge and skills we divided the team as following

|  |  |  |
| --- | --- | --- |
| **MODULES** | **STUDENT NAME** | **ROLL NO** |
| **Requirement Analysis**  **(Mini specification)** | Tenzin Rabten  Antara Das | 0201CS15087  0201IP151016 |
| **Coding and Testing** | Tenzin Rabten  Antara Das | 0201CS15087  0201IP151016 |
| **Documentation** | Tenzin Rabten  Antara Das | 0201CS151087  0201PI151016 |

**4. SOFTWARE REQUIREMENT SPECIFICATION**

**1. Overall Description**

This section is about the requirements, constraints and the interfaces included in the project. A map of functions are also supplied. The document follows the IEEE standards, yet some of the sections are discarded as they are not compatible for this project.

**2. Product Functions**

Major functions of the product and brief descriptions of these functions can be found in this section. Also detailed diagrams and descriptions can be found in subsections of this section.

|  |  |  |
| --- | --- | --- |
| ID | Function | Description |
| 1 | Train System | Training the system’s machine learning part for better result |
| 2 | Image Capture | Getting the image from the browser for prediction |
| 3 | Predict Class | Predicting the class – positive, neutral or negative to which the emoji belongs to. |

2.1.1 Use­ Case Model Survey

This section includes use case diagrams and their detailed descriptions of the functions that mentioned in section 2.1.

**3. Specific Requirements**

**3.1 Functional Requirements**

As mentioned at section 2.1.1, these requirements are categorized by use cases. For any specific use case, there are specific requirements which are detailed below.

3.1.1 Train System Requirements

● The system should provide a method for admin.

● The system should provide taking new data from admin to train classifiers to improve reliability.

3.1.2 Capture Image Requirement

* The system should provide an interface to capture image from the browser to be used for prediction.
* A method should be provided to get the captured image as an input for prediction.

3.1.3 Emoji Classifier Requirements

● The system should provide functions which can take the image and predict the percentage likelihood of it belonging to a particular class.

● The system should provide a well-­trained classifier to generate better inputs for classifier.

**3.2 Non­-Functional Requirements**

**3.2.1 Usability**

The system should be easy to use. The user should reach the summarized text with one button press if possible. Because one of the software’s features is timesaving. The system also should be user friendly for admins because anyone can be admin instead of programmers.

**3.2.2 Reliability**

This software will be developed with machine learning, computer vision and deep learning techniques. So, in this step there is no certain reliable percentage that is measurable. Also, user provided data will be used to compare with result and measure reliability. With recent machine learning techniques, user gained data should be enough for reliability if enough data is obtained. The maintenance period should not be a matter because the reliable version is always run on the server which allow users to access summarization. When admins want to update, it take long as upload and update time of executable on server. The users can be reach and use program at any time, so maintenance should not be a big issue.

**3.2.3 Performance Calculation**

Time and response time should be as little as possible, because one of the software’s features is timesaving. Whole cycle of classifying an image should not be more than 30 seconds. Calculation and response times are very low, and this comes with that there can be so many sessions at the same times.

**11 3.2.4 Supportability**

The system should require Python knowledge to maintenance. If any problem arise deep learning methods, it requires code knowledge and deep learning background to solve. Problems should be fixed with an update and it also require code knowledge and network knowledge.

**4. Data Model and Description**

This section includes components and data objects of the system.

4.1 Components

Our system will consist of 2 main components:

1. **SOFTWARE INTERFACES**

1. Take picture of image seen in internet:

-- OpenCV

Open Source Computer Vision Library is an open source computer vision and machine learning software library.

We used OpenCV library functions to capture the screenshots of the monitor and display it in a window.

2. Building the actual model:

-- Keras:Keras is an open-source neural-network library written in Python.

Keras is a high-level neural networks API, written in Python and capable of running on

top of TensorFlow.

Keras is being used in this project to build the neural network.

-- TensorFlow:

TensorFlow is an open source software library for numerical computation using

data flow graphs. The graph nodes represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) that flow between them.

TensorFlow provides a collection of workflows to develop and train models using Python.

1. **DATA COLLECTION**

The data set used for this project is being built by collecting different images of face emojis from the internet and resizing each to a size of 200 x 200 pixels.

The Dataset is being divided into two subsets: Train and Test set.

Each of the subsets contain images belonging to three classes: Positive, Neutral and Negative.

The train set contains a total of 45 images, with 15 images belonging to each of the three classes.

Data Augmentation is applied over the data collected to increase the amount of data and improve the accuracy of the prediction.

**Data Augmentation:**

For images, we can increase the size/variety of our data set using generators, using ImageDataGenerator class

Generate batches of tensor image data with real-time data augmentation. The data will be looped over (in batches).

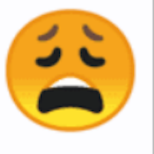
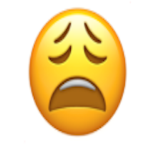
**TRAIN SET**

**POSITIVE IMAGES**

**NEUTRAL IMAGES**



**NEGATIVE IMAGES**



1. **METHODOLOGY**

**Classification using neural network**

Machine learning algorithms are often used for classification purposes. In general, a training set is constructed from known samples and used to train a classifier. The classifier can then be used to predict unseen samples with the most probable class, often done with a testing set to evaluate performance. It is of importance that these sets are different and that the testing set is not used during development to avoid over-training which occurs when the algorithm performs good on testing data but generalizes badly.

Emoji Classes

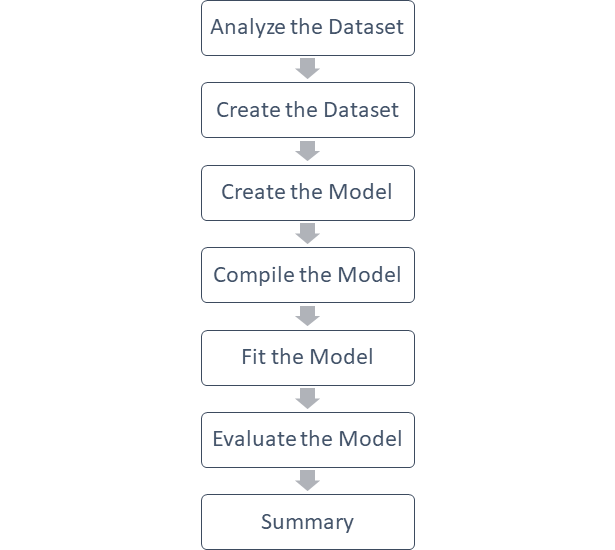
* Positive
* Negative
* Neutral

Convolution Neural Network for Predicting Class of Emojis

This project uses deep learning to classify the input image of an emoji into the above three classes.

For any input image the likelihood of it being positive, negative and neutral is being mentioned.

**BUILDING THE MODEL**



CNN Architecture

Layers in CNN

1. Input layer- Consist of Height x Width x Depth (R,G,B)
2. Convolutional Layer- Connected to small part of the input
3. Activation- RELU activation used in the CNN Layer
4. Pooling Layer- Pooling used for downsampling on the width, height
5. Fully connected layer



**IMPLEMENTATION**

The two tasks to be accomplished in this project were:

**1.Capture picture of image seen in internet:**

For this we created a file “im\_capture” that runs the image capture and lets save the screenshot.

**Workflow**

create a window to hold the feed, and set up the mouse callback

- infinitely monitor the screen

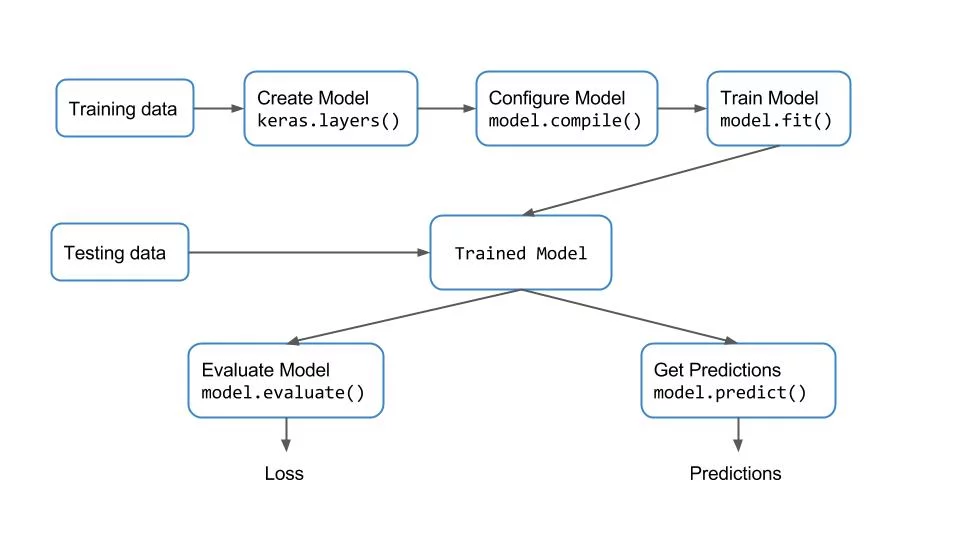
- grabs a screenshot of the entire screen

- converts that screenshot to a NumPy array

- show the image on the display window we created earlier

the user can quit by pressing 'q'

**2. Machine Learning Model:**

**Workflow for Training the Network**

**Model Used:** SEQUENTIAL MODEL

# The Sequential model is a linear stack of layers.

**Functions to build the model**

--mid layers of the model:

model.add(MaxPooling2D(pool\_size=(2, 2)))

model.add(Dropout(0.5))

model.add(Flatten())

model.add(Dense(16, activation='relu'))

model.add(Dropout(0.5))

-- model.add(MaxPooling2D(pool\_size=(2, 2)))

-----Max Pooling : helps condense data

taking 2 by 2 sections of image and pooling them together to take the maximum-considering it the determining feature.

-- model.add(Dropout(0.5))

-----plays a role in overfitting,

don’t give the model the same data every time--- 50% data dropped out every time---different subset of data given to model every time.

-- model.add(MaxPooling2D(pool\_size=(2, 2)))

-----Max Pooling : helps condense data

taking 2 by 2 sections of image and pooling them together to take the maximum-considering it the determining feature.

-- model.add(Dropout(0.5))

-----plays a role in overfitting,

don’t give the model the same data every time--- 50% data dropped out every time---different subset of data given to model every time.

Overfitting occurs mainly because the network parameters are getting too biased towards the training data. We can add a dropout layer to overcome this problem to a certain extent. In case of dropout, a fraction of neurons is randomly turned off during the training process, reducing the dependency on the training set by some amount.

-- model.add(Flatten())

----- Flattening a tensor means to remove all of the dimensions except for one.

A Flatten layer in Keras reshapes the tensor to have a shape that is equal to the number of elements contained in the tensor.

This is the same thing as making a 1d-array of elements.

-- model.add(Dense(16, activation='relu’))

----- tells model the value of decision it makes,

activation tells how things are being pre- weight to a degree.

-----Dense implements the operation:

output = activation(dot(input, kernel) + bias)

where activation is the element-wise activation function passed as the activation argument,

kernel is a weights matrix created by the layer,

and bias is a bias vector created by the layer (only applicable if use\_bias is True).

1. **RESULTS**
   * + 1. **POSITIVE IMAGES AS INPUT**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMOJI | PREDICTION (Likelihood In %age) | | | PREDICTED CLASS |
| Positive | Neutral | Negative |
|  | 35.27 | 29.51 | 35.20 | POSITIVE |
|  | 40.79 | 27.65 | 31.54 | POSITIVE |
|  | 100 | 0 | 0 | POSITIVE |
|  | 0 | 0 | 100 | NEGATIVE |
|  | 100 | 0 | 0 | POSITIVE |

* + - 1. **NEUTRAL IMAGES AS INPUT**

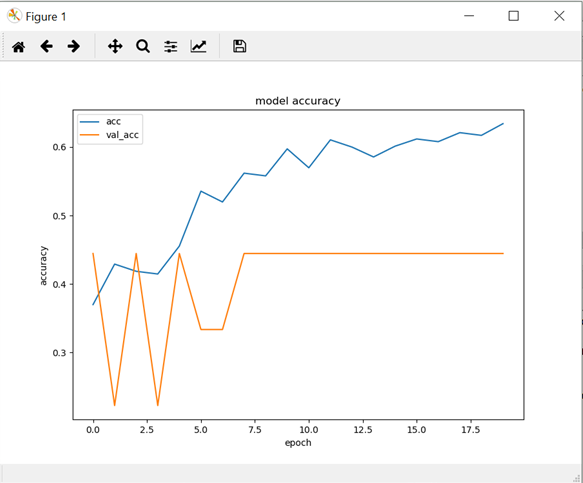
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMOJI | PREDICTION (Likelihood In %age) | | | PREDICTED CLASS |
| Positive | Neutral | Negative |
|  | 35.20 | 35.27 | 29.51 | NEUTRAL |
|  | 27.91 | 40.79 | 31.54 | NEUTRAL |
|  | 0 | 100 | 0 | NEUTRAL |
|  | 20.90 | 50.20 | 30.25 | NEUTRAL |
|  | 0 | 100 | 0 | NEUTRAL |

1. **NEGATIVE IMAGES AS INPUT**

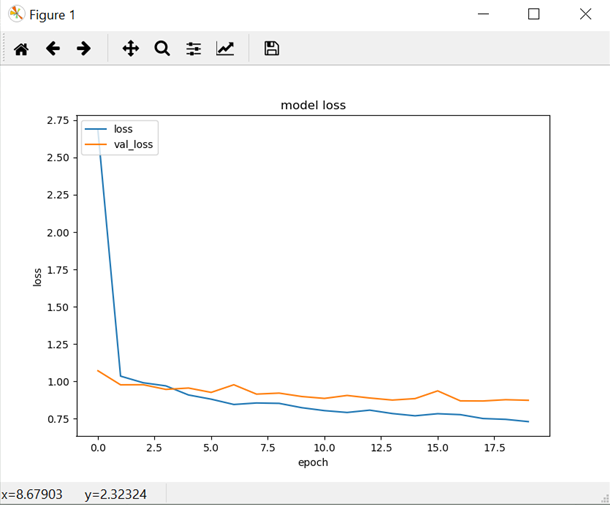
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EMOJI | PREDICTION (Likelihood In %age) | | | PREDICTED CLASS |
| Positive | Neutral | Negative |
| Image result for emoji negative | 15.30 | 40.20 | 45.55 | NEGATIVE |
| Image result for emoji negative | 0 | 0 | 100 | NEGATIVE |
| Image result for emoji negative | 31.76 | 29.31 | 39.21 | NEGATIVE |
|  | 0 | 0 | 100 | NEGATIVE |
|  | 0 | 0 | 100 | NEGATIVE |

1. **Testing**

EPOCH VS ACCURACY GRAPH



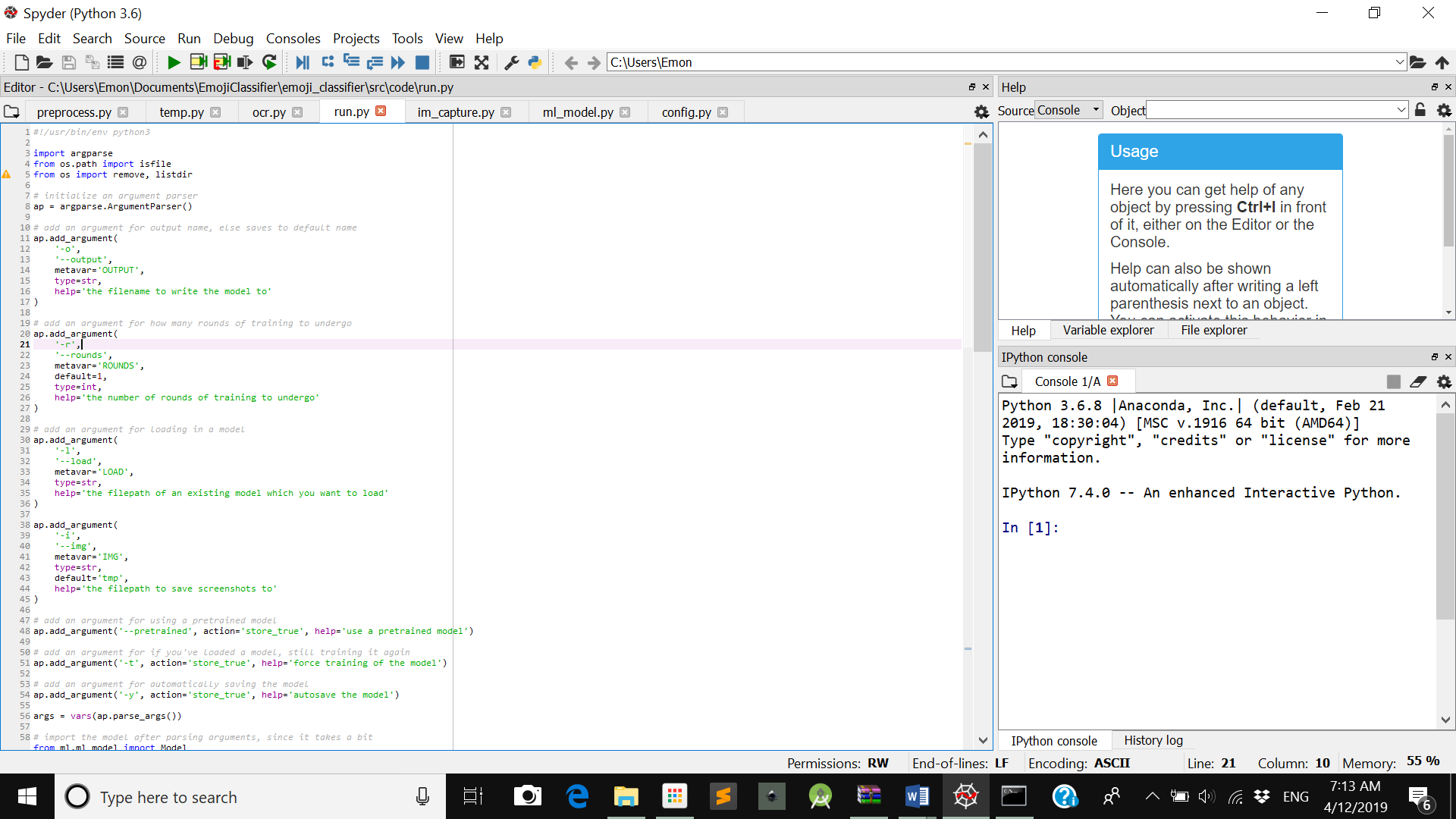
EPOCH VS LOSS GRAPH

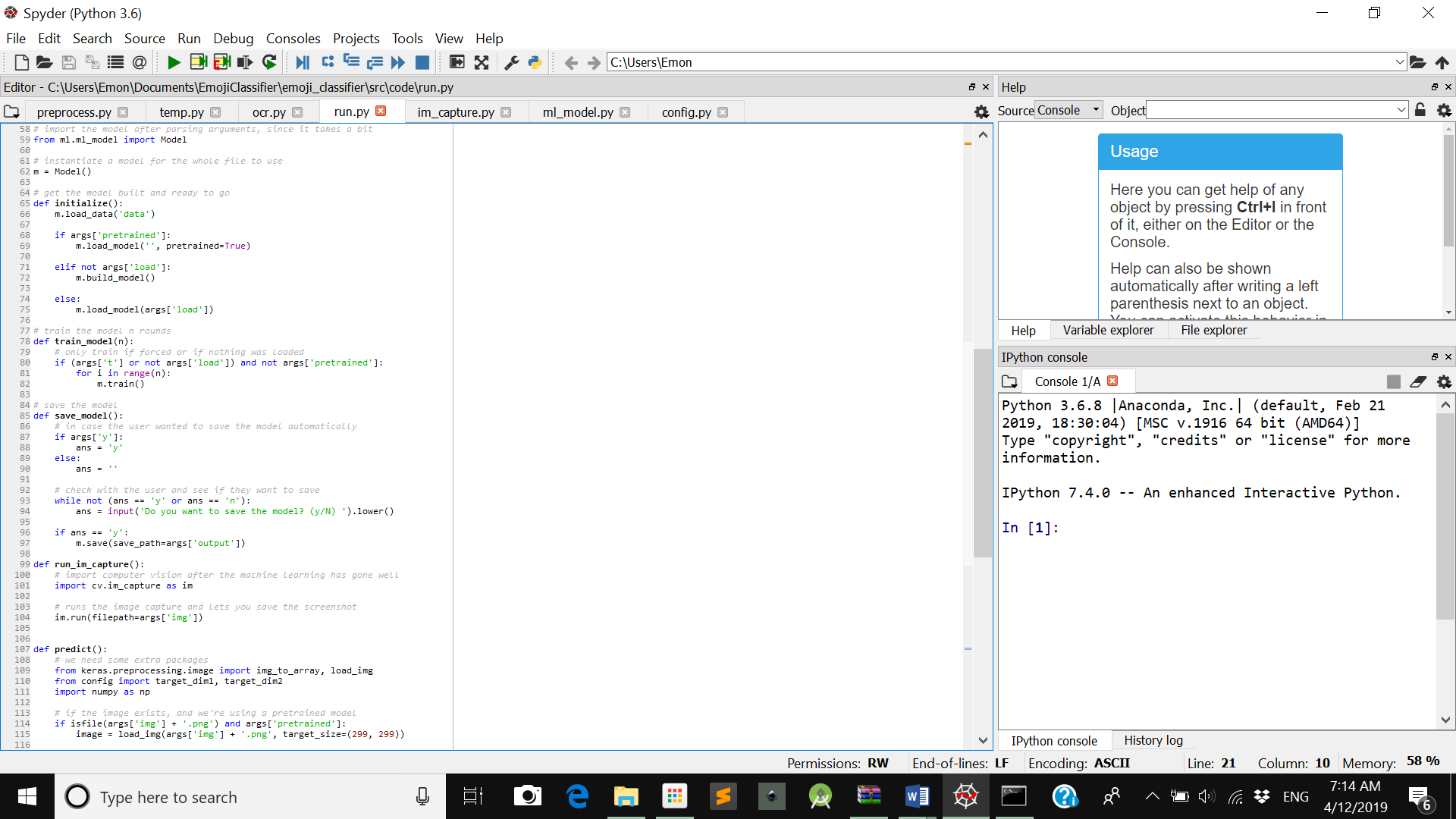


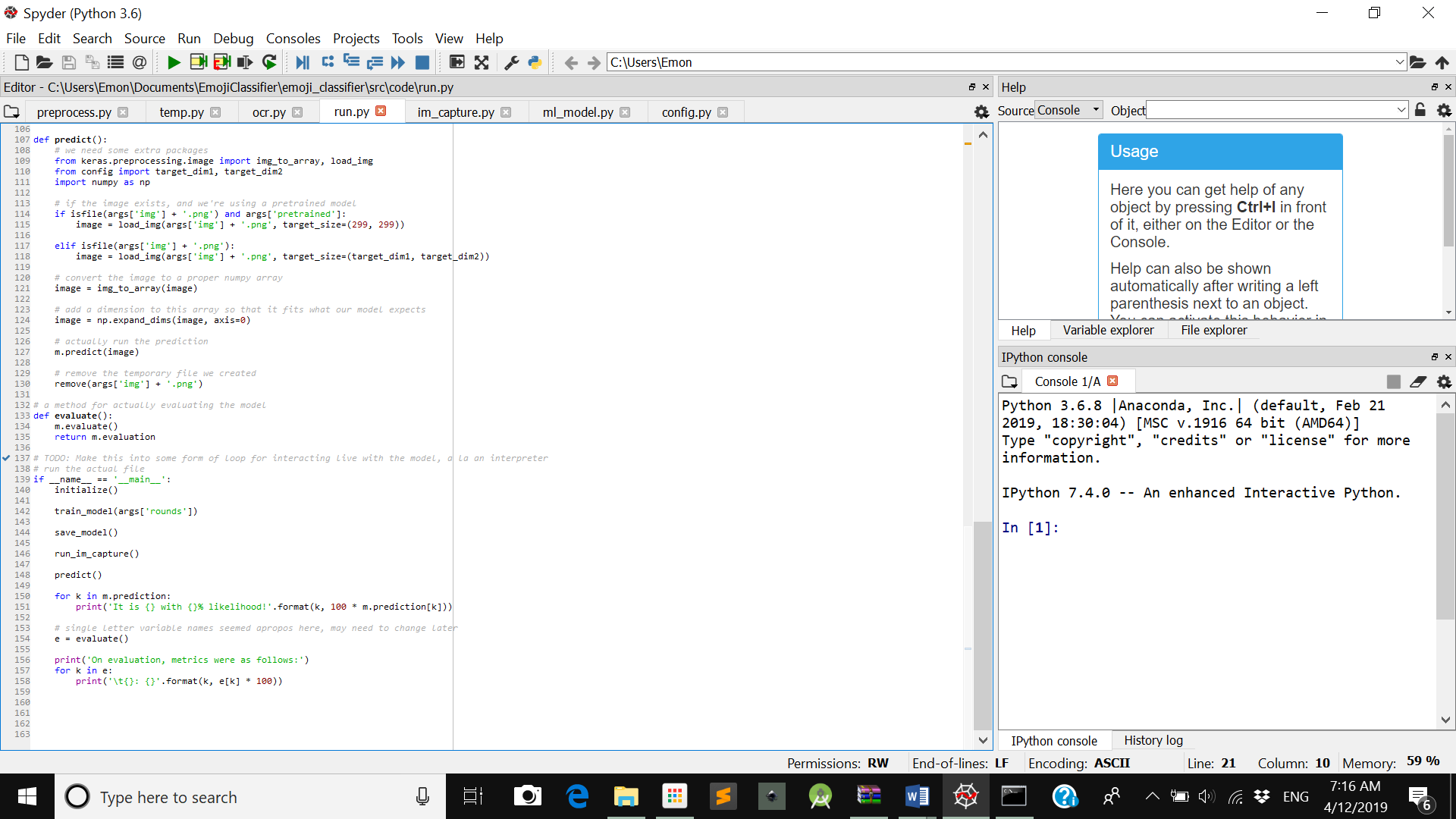
1. **APPENDIX**

**8.1 SOURCE CODE SCREENSHOTS**

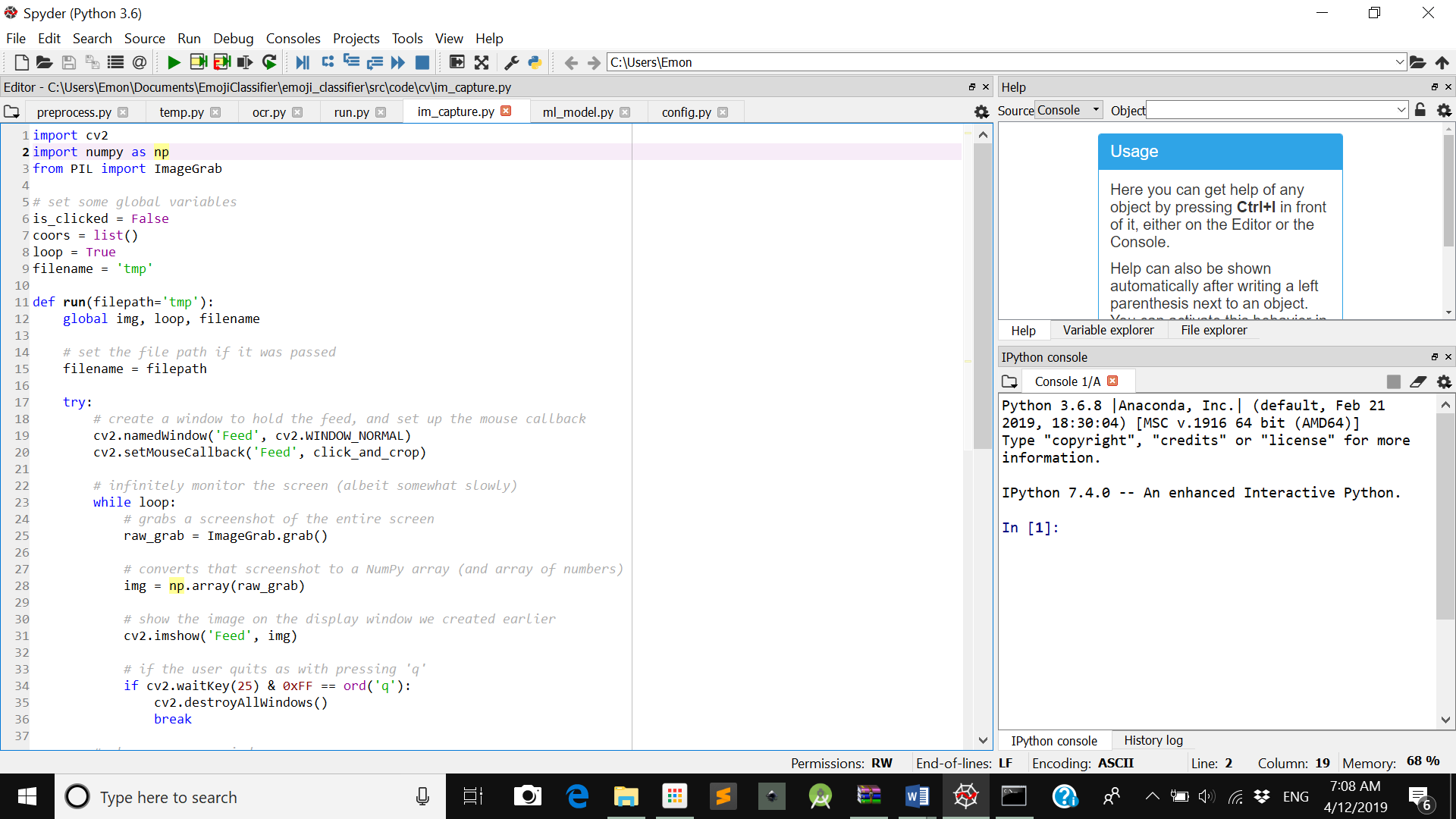
**8.1.1 run.py**

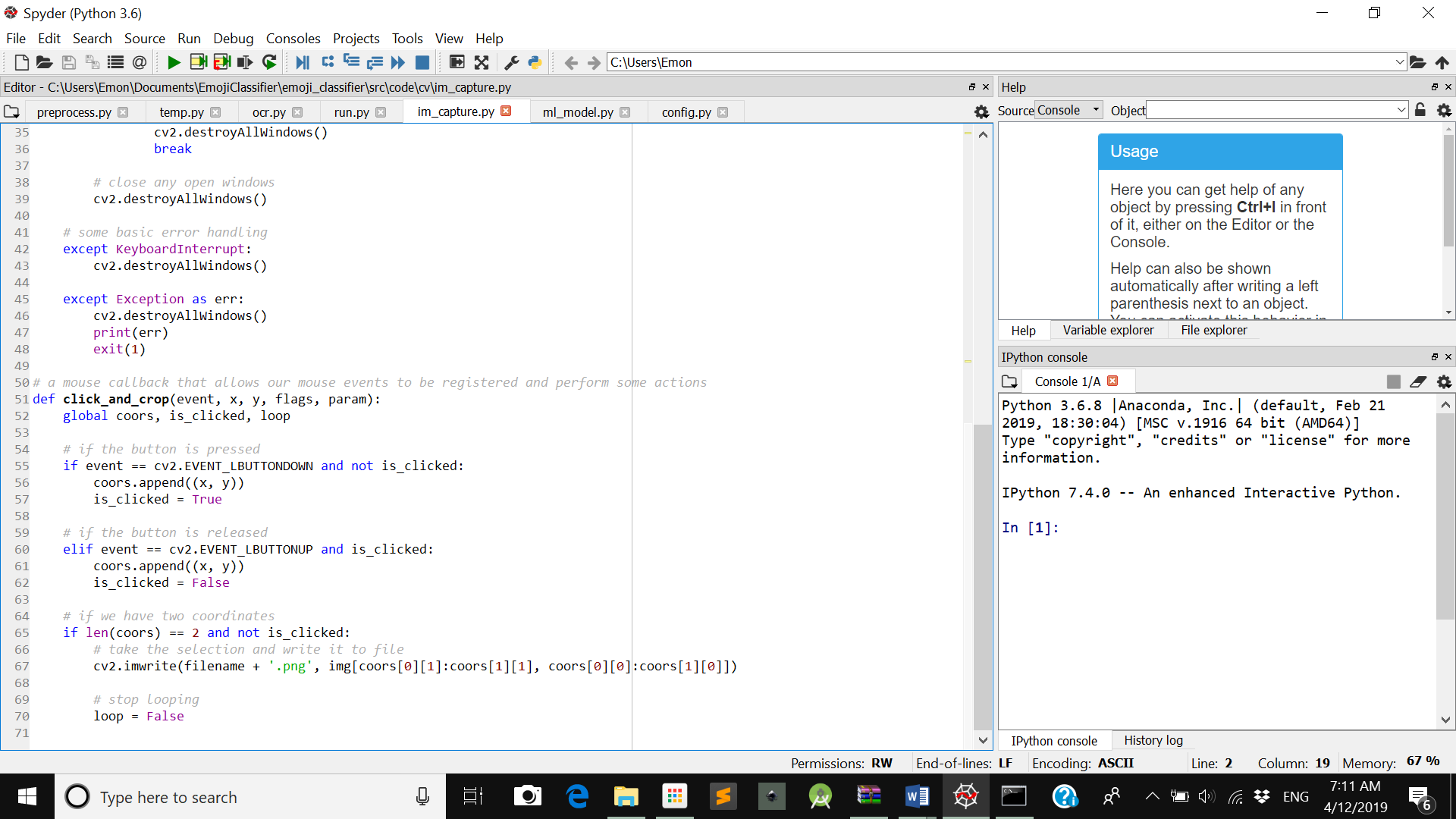
****



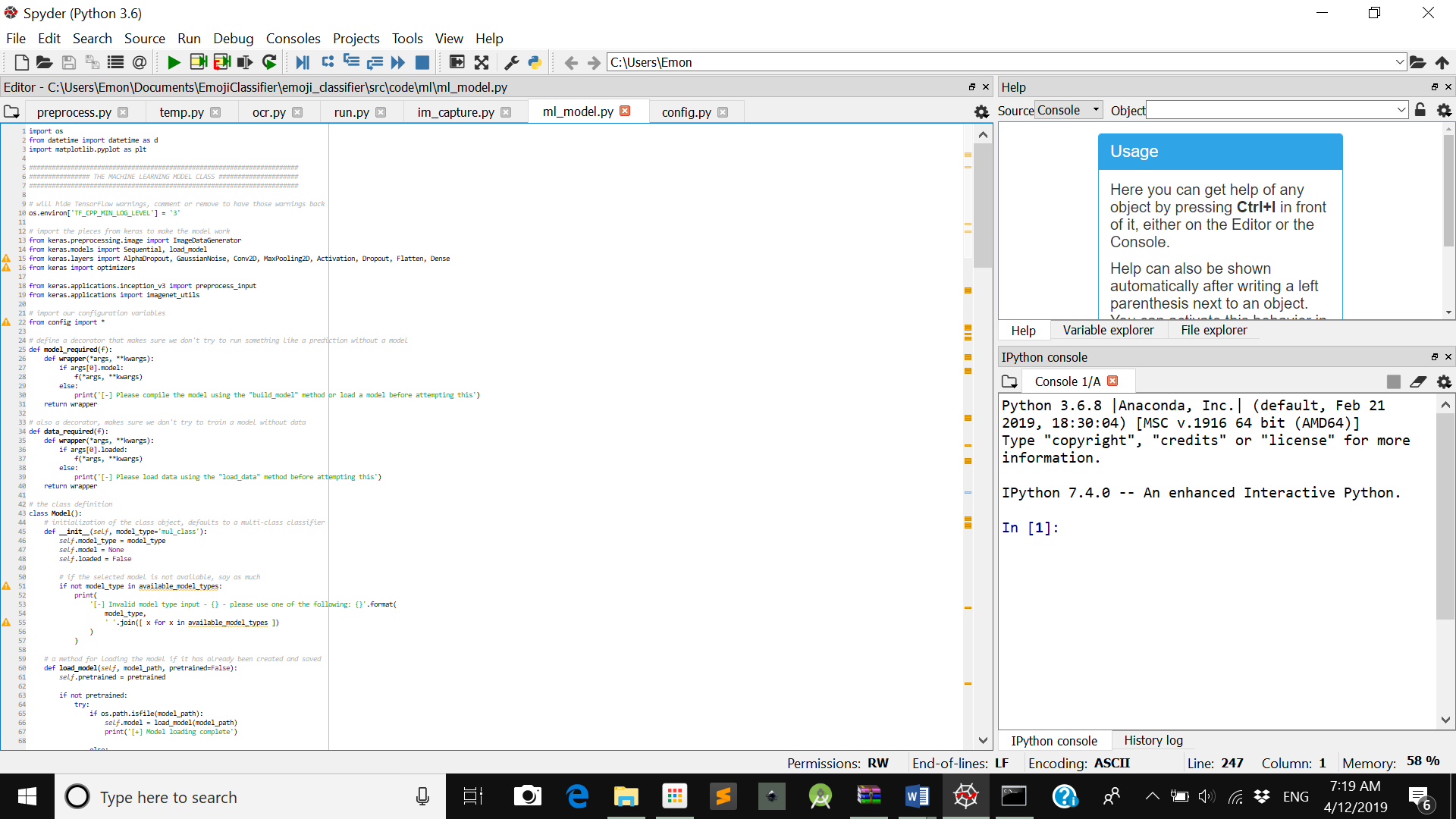


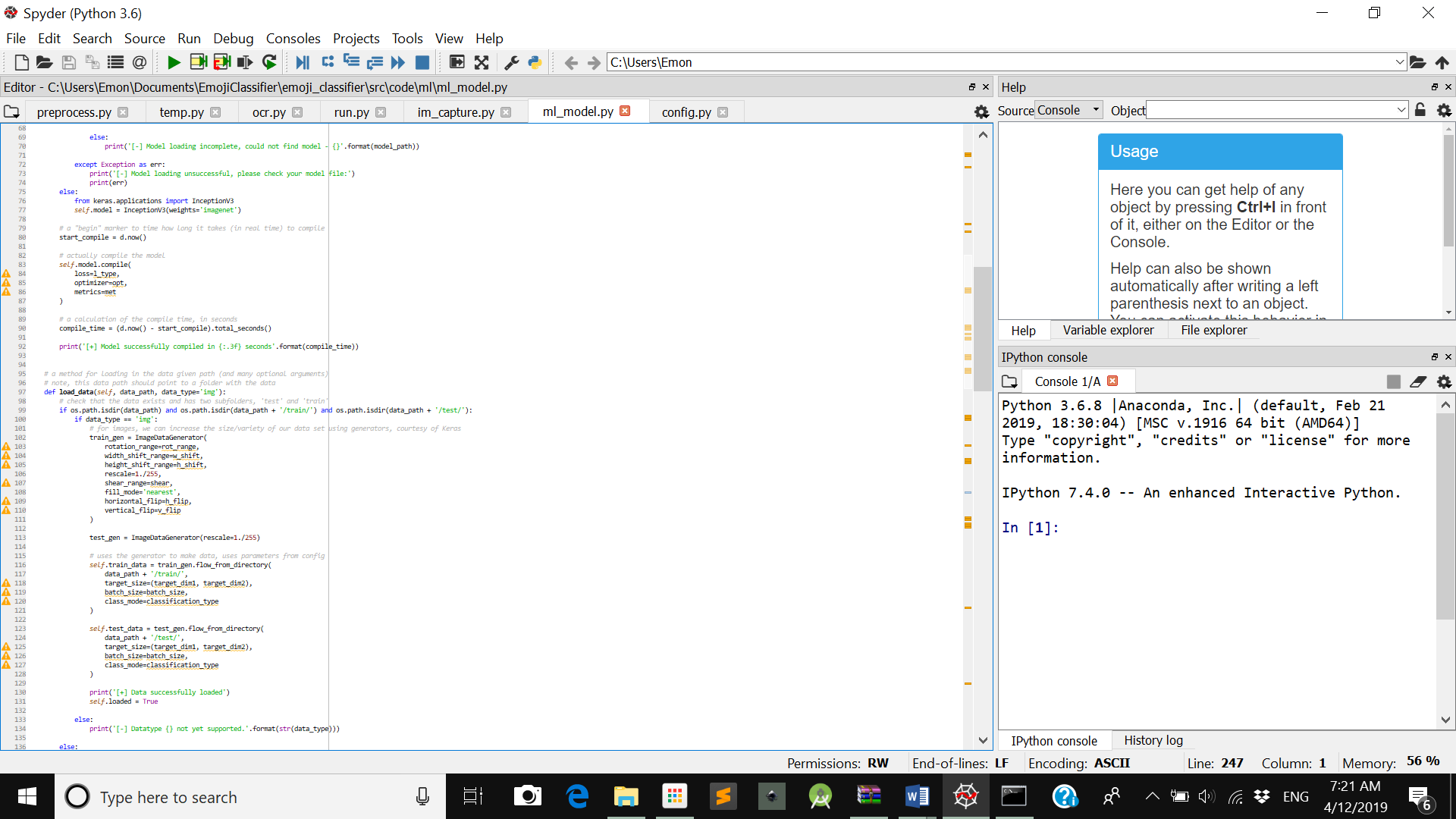
**8.1.2 im\_capture.py**

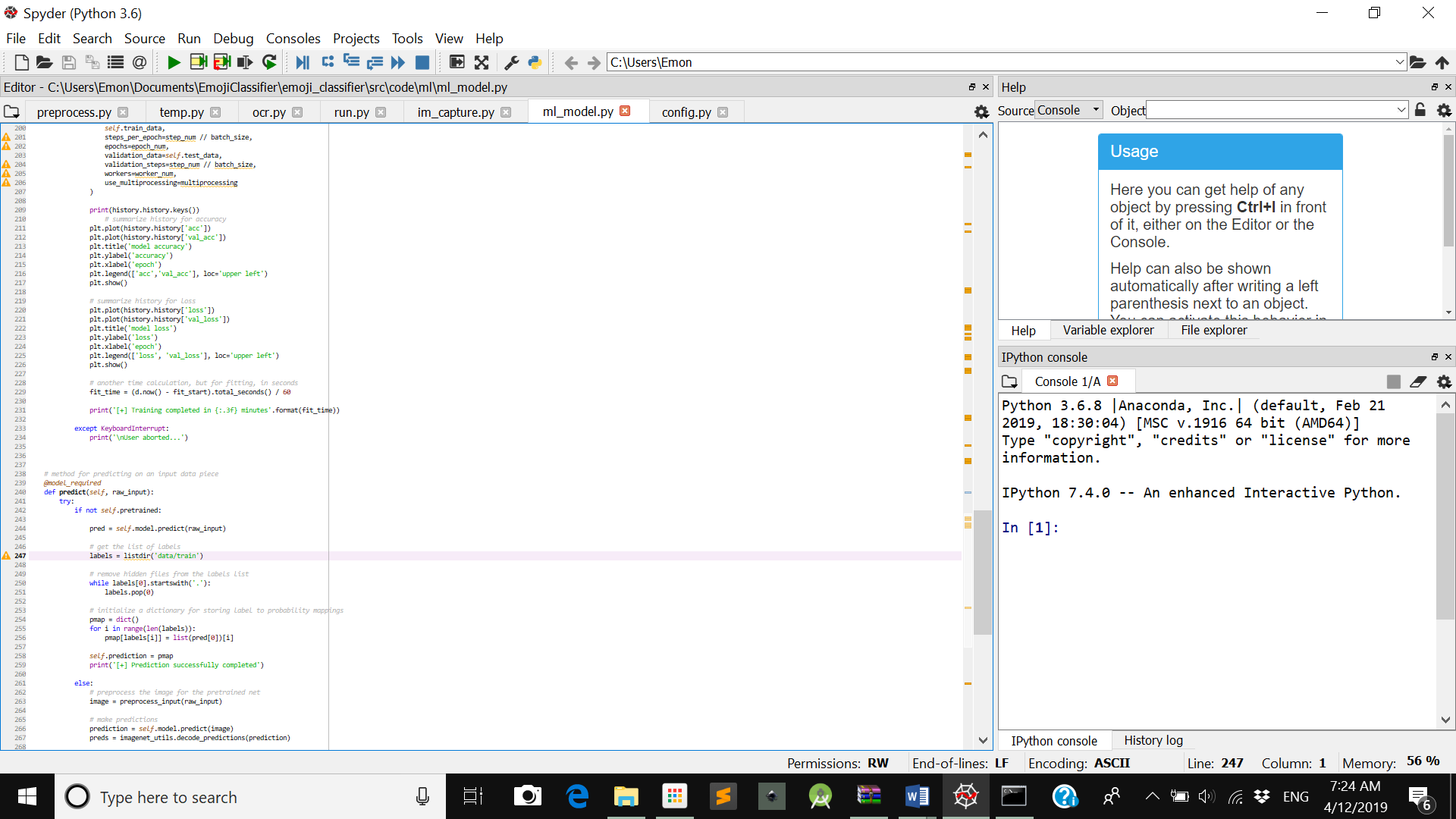
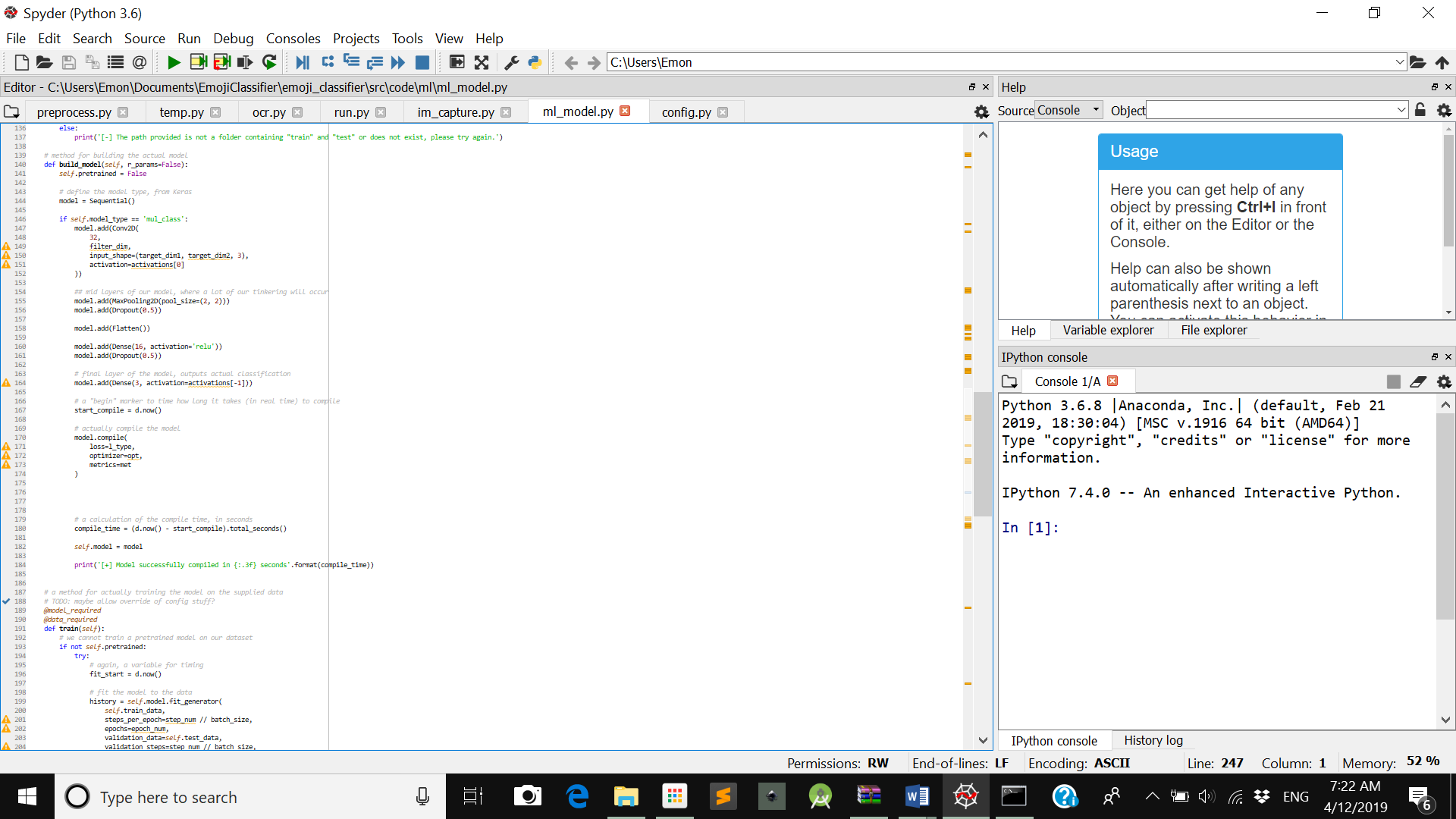


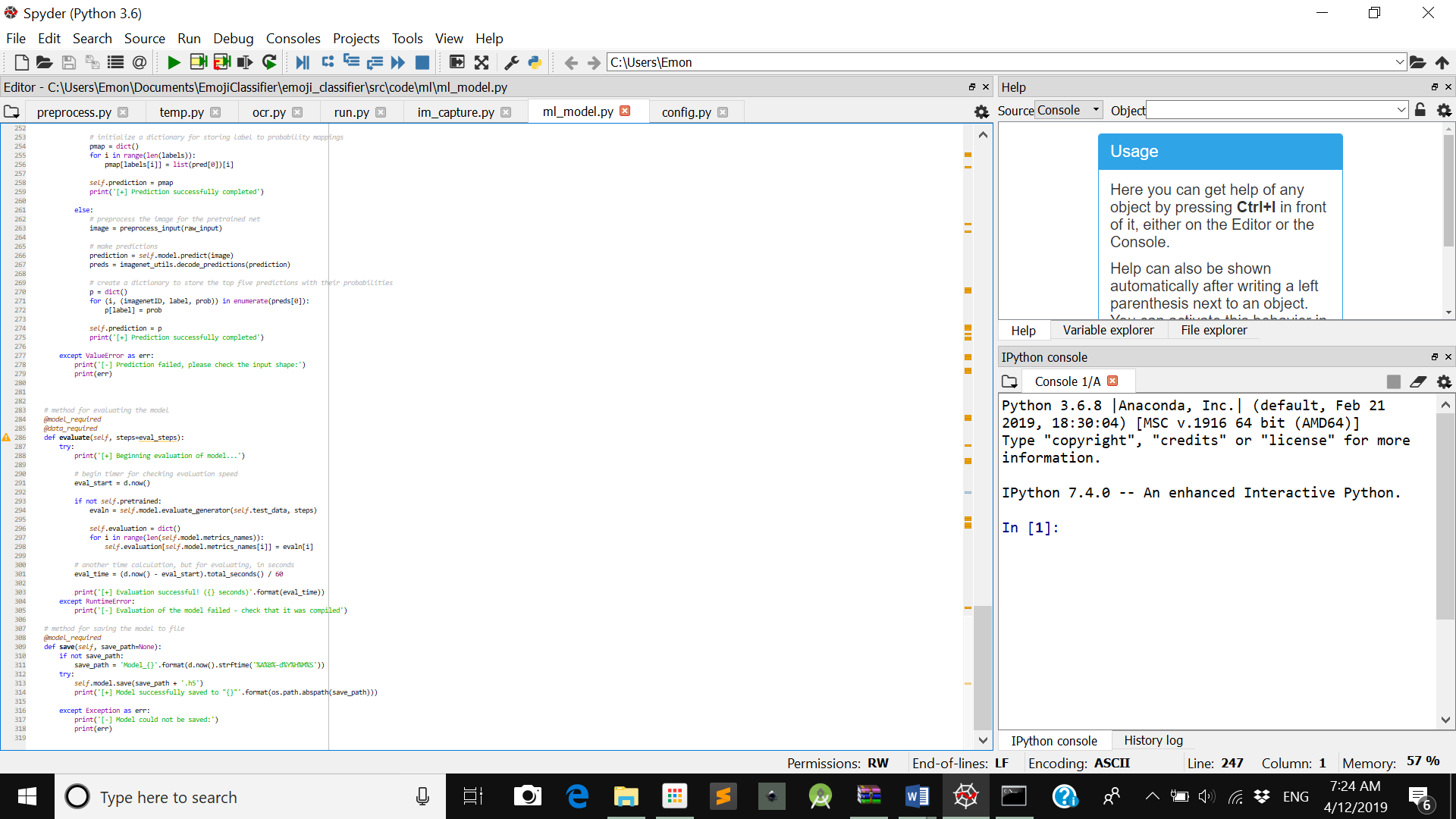


**8.1.3 ml\_model.py**

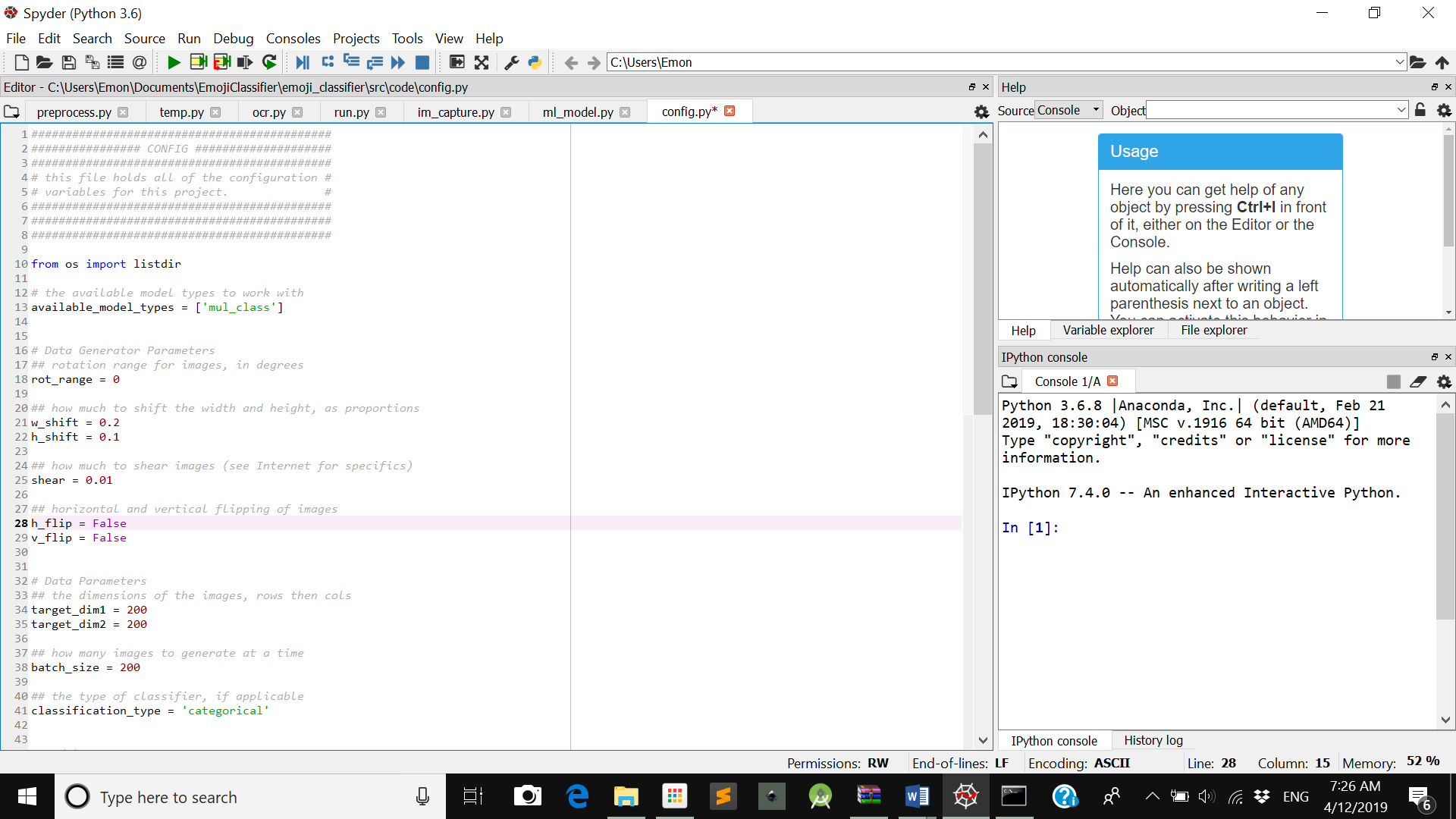


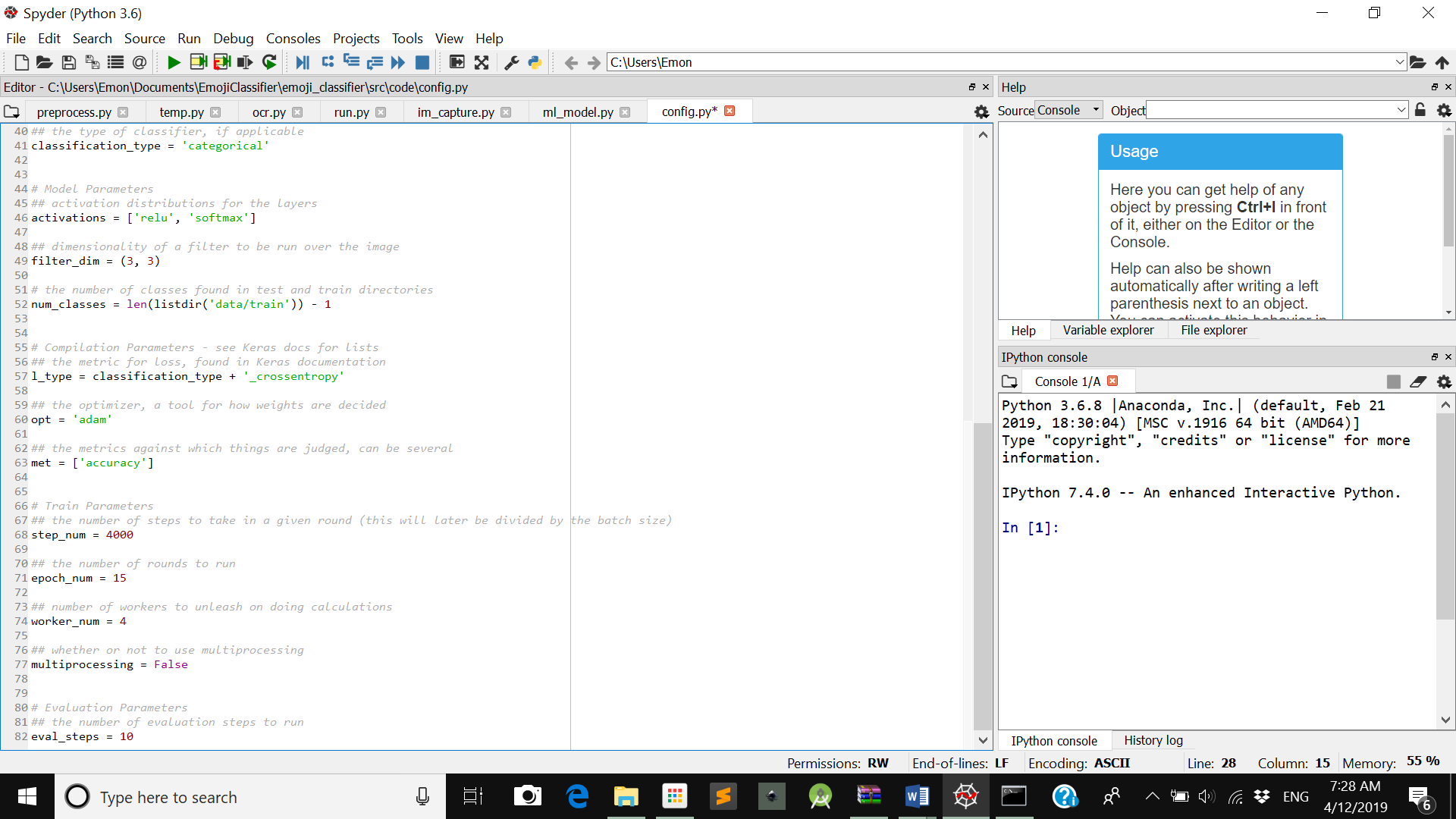




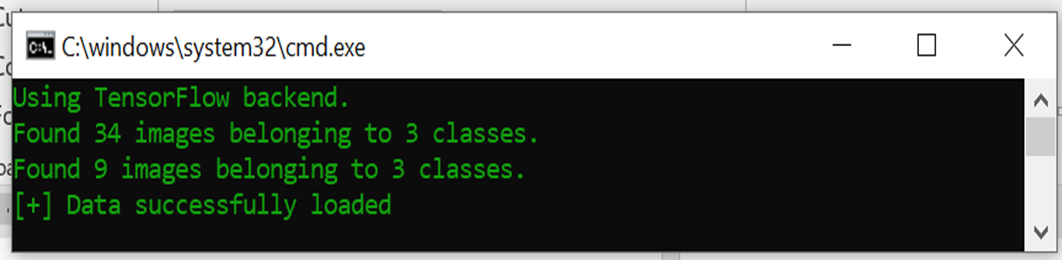


**8.1.4 config.py**

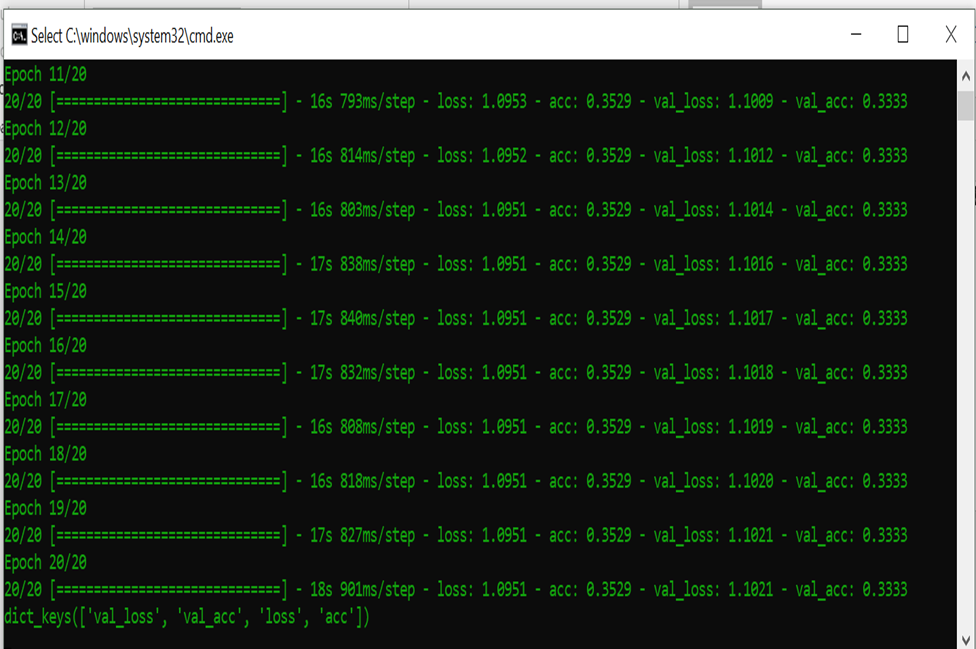




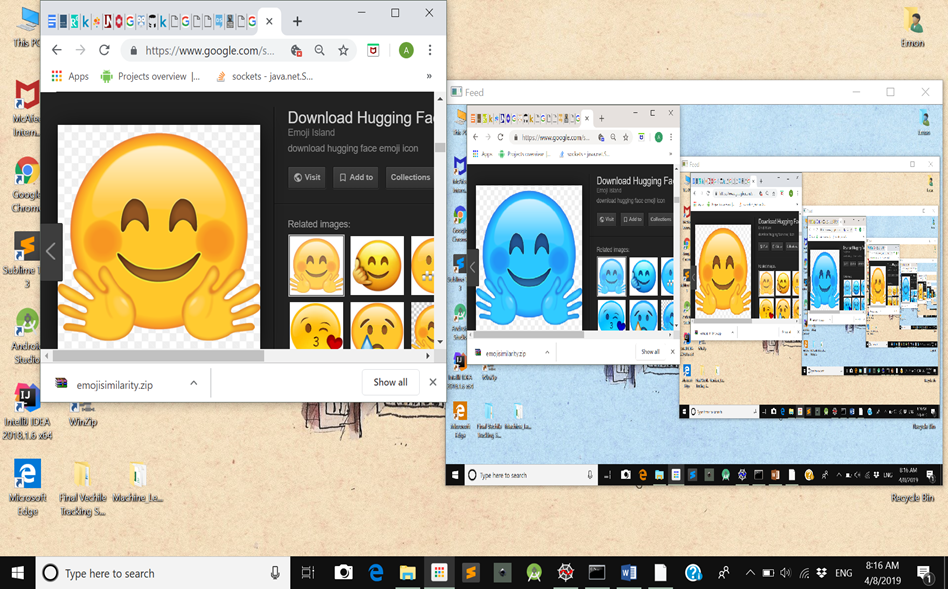
8.2 TRAINING THE MODEL







8.3 LIVE FEED WINDOW



8.4 PREDICTION

