R1. A table of content with hyperlinkable page to each relevant section (Hint: Use Word->Reference->Table of content and make each chapter/section description as Heading 1 or 2 under Styles first)

R2. Describe in your report the software and libraries used as well as how to set up, configure, and run your code.

**Libraries**: Numpy, Pandas, Keras, Matplotlib, Sklearn

**Selected Machine Learning Algorithms:**

(Reference: <https://colab.research.google.com/drive/1Dvu-f4xFOLgGP-eHbUNuH6hgmxj7E7rI?authuser=1>,

<https://colab.research.google.com/drive/1yYMH9UDw8BI_qLukFLsqsGmRjjhpw5CC?authuser=1#scrollTo=scdIaOalsMa7>) v- WE CANT CITE IT SINCE THERE IS NO FREE ACCESS

During our data preparation, the data split into train and test data, the ratio is 5 to 1.

The table of the ML algorithms we tried to run several times, got the best prediction score and the time running, which was shown below:

|  |  |  |
| --- | --- | --- |
| Models | Score (mean accuracy on the given test data and labels) | Time(s) |
| DecisionTreeClassifier | **58.5%** | **15.559** |
| BaggingClassifier | **61.6%** | **104.556** |
| ExtraTreesClassifier | **62.6%** | **0.925** |
| RandomForestClassifier | **64.2%** | **2.008** |
| AdaBoost classifier | **58.3%** | **66.003** |
| GradientBoostingClassifier | **65.8%** | **122.632** |
| SVM kernel = “rbf” | **63.2%** | **116.786** |
| SVM kernel = “poly” | **59.4%** | **101.656** |
| SVM kernel = “linear” | **58.5%** | **263.394** |
| SVM kernel = “sigmoid” | **59.4%** | **101.656** |
| LogisticRegressionClassifier | **59.6%** | **29.912** |
| KNeighborsClassifier | **61.3%** | **62.323** |

From these algorithms, most we use the default value, then we tried to tune the different parameters in some selected algorithms, which show better accuracy, including extra decision tree, random forest, gradient boosting, SVM with ‘rbf’ kernel. From the accuracy, the following we will emphasize on the ExtraDecisonTree Classifier and the SVM with the kernel “rbf”, which shows a better performance.

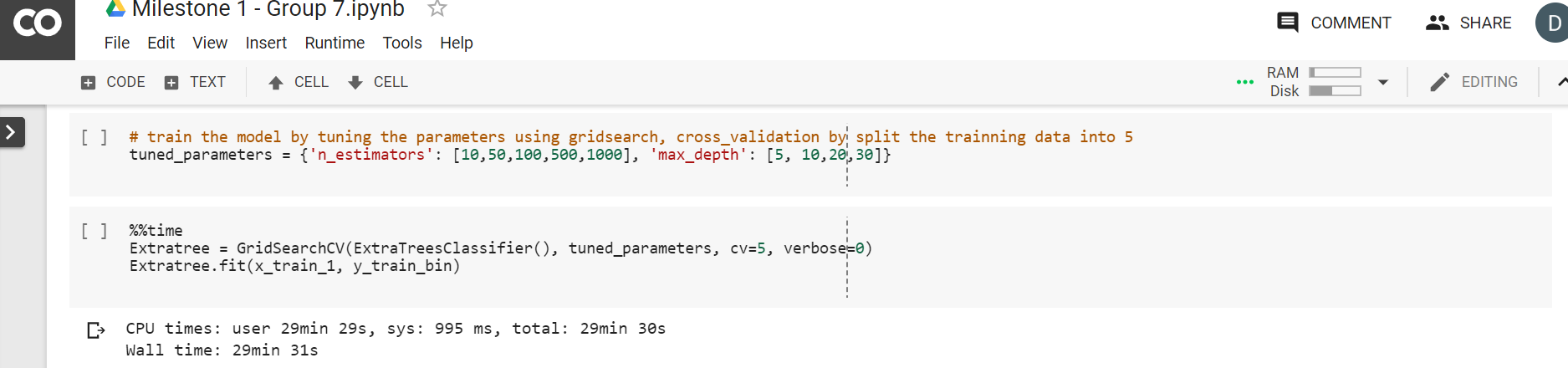
ExtraDecisonTree Classifier:

Data setup:

Reshape the data, the ratio of the train data to the test data is 5 to 1, and use the 5 fold split to do the cross-validation

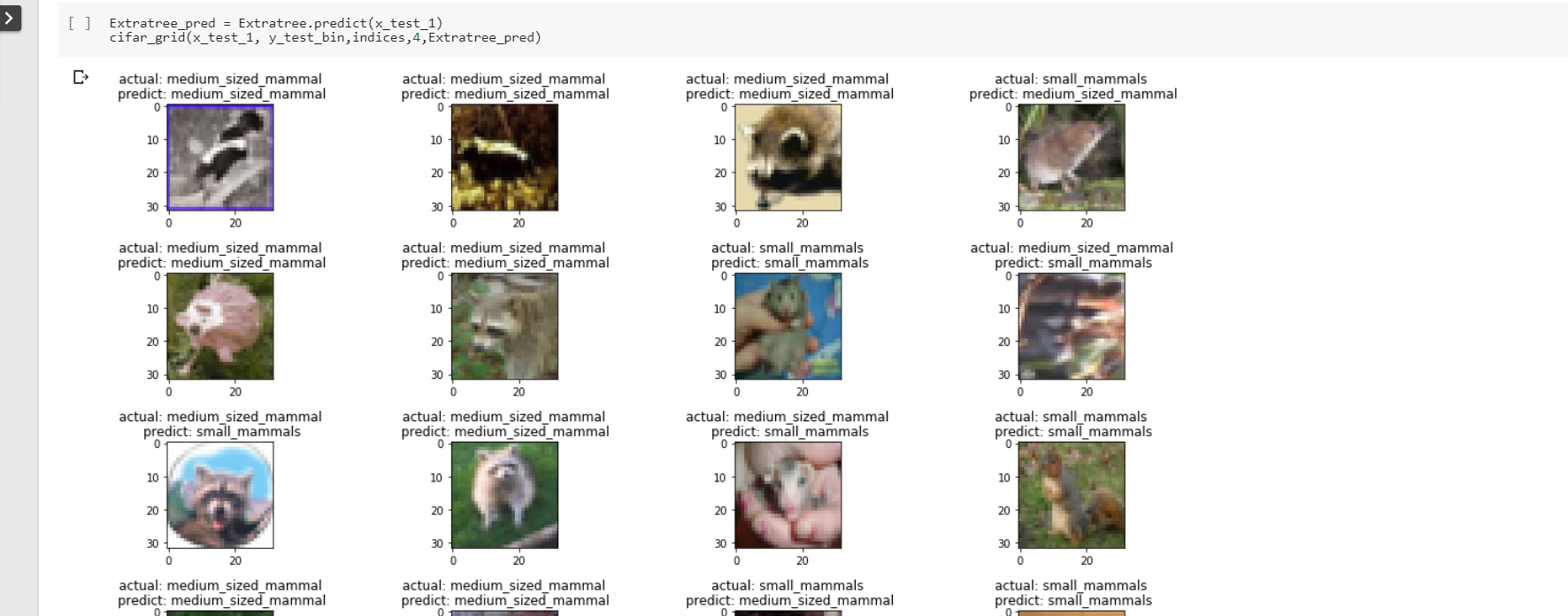
Setup the model:

Define and build the model, fit the model using the training data and generate the prediction, finally, evaluate the model’s performance. We also use the gridsearch to find out the best parameter:



Prediction result:

We generate the prediction by the .predict(), the complete result please refer to the collaboratory, the following just the part of the prediction images:



Model evaluation:

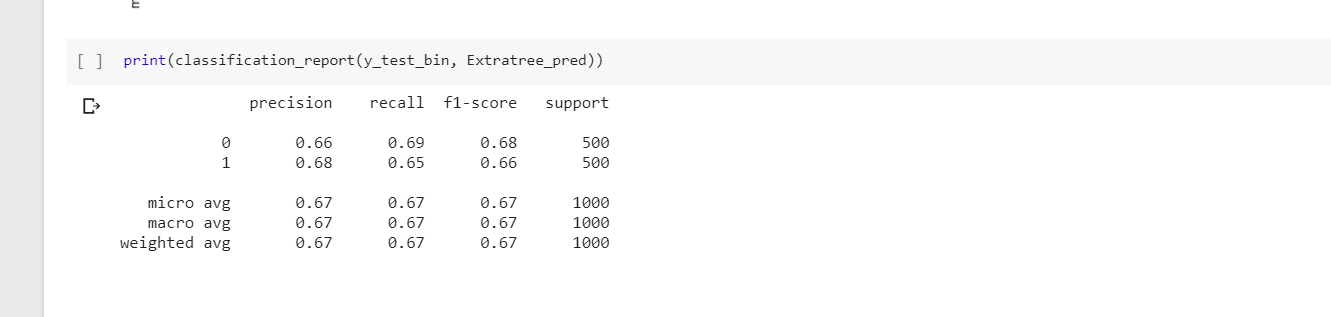
(Reference: <https://colab.research.google.com/drive/1IVWXPZHwRqyDzj5Y3xo3Q69A_9Q_1I8x?authuser=1#scrollTo=3rEDDdHr7kxb>)

From the result of the confusion matrix and classification report, the number of observations of the labeled classes is balanced, and F1 score of the small\_mammals prediction is a little bit higher than the medium\_sized\_mammals, and the heatmap of the confusion matrix also indicates the prediction of the small\_mammals is better than medium\_sized\_mammals.

confusion matrix:

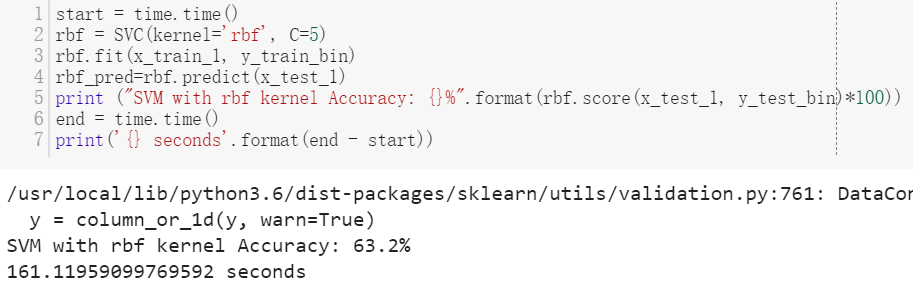


classification report:

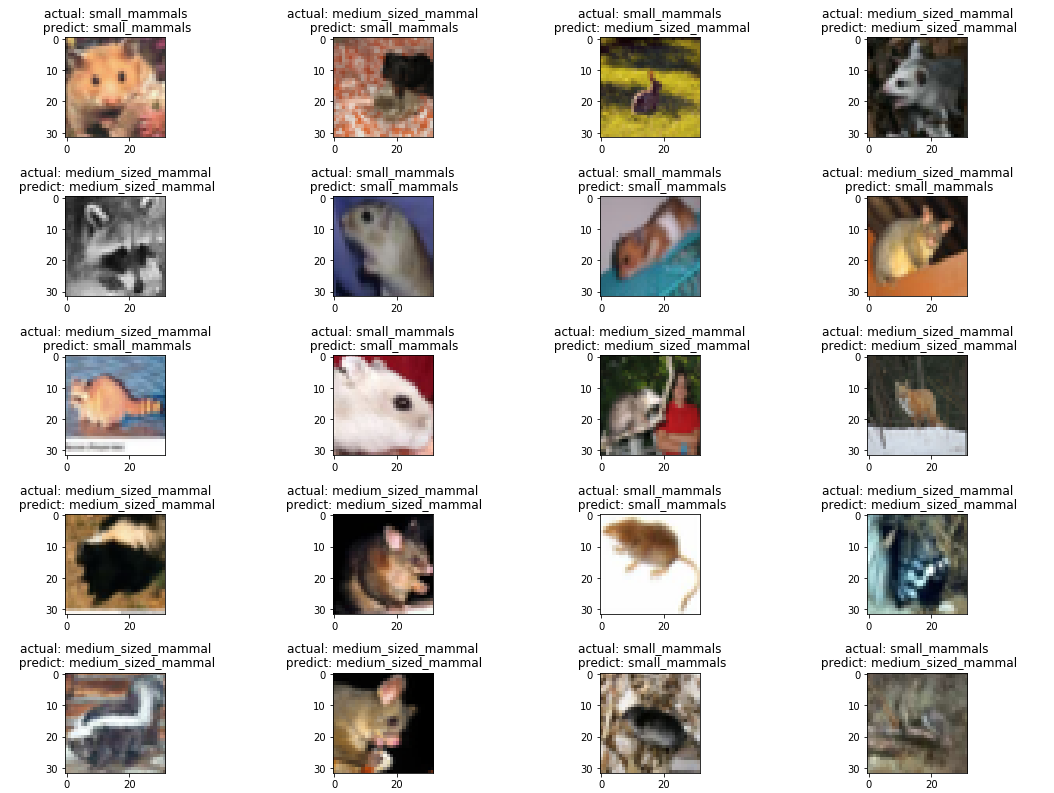


SVM with “rbf” kernel:

Data and the model setup are similar to the RandomForest model:



Prediction result:

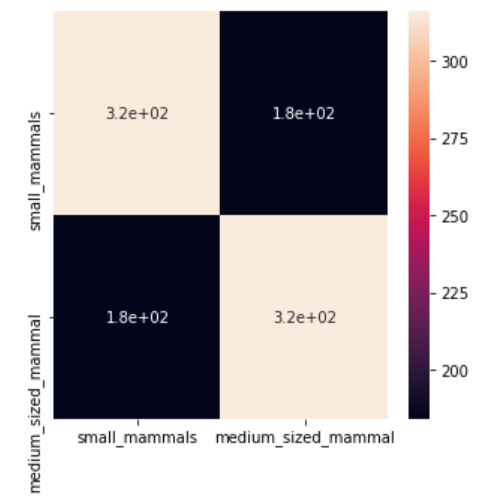




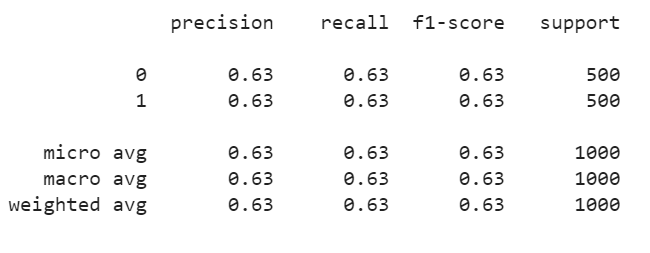
Model evaluation:

From the result of the confusion matrix and classification report, the number of observations of the labeled classes is balanced, and F1 score of the small\_mammals prediction equals to the medium\_sized\_mammals, and the heatmap of confusion matrix also indicates that the prediction of the two classes is same.

confusion matrix:



classification report:



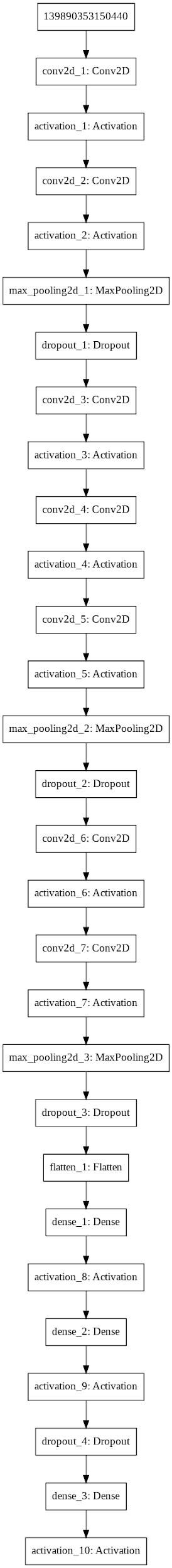
Lesson learned and the work for the milestone2:

The most important lesson we learned is that data is the key, validate you got the right data, then get started.

What we will improve our work is continue to tune the value of parameters try to get the best performance for each ML algorithms we selected on milestone 2.

Qiao’s CNN model:

Model Architecture:

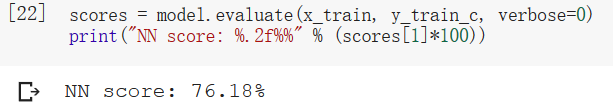


Because we are trying to do feature recognition for small pictures, I tried to use a long and narrow convolutional neural network to detect the different features for these 2 types of mammals.

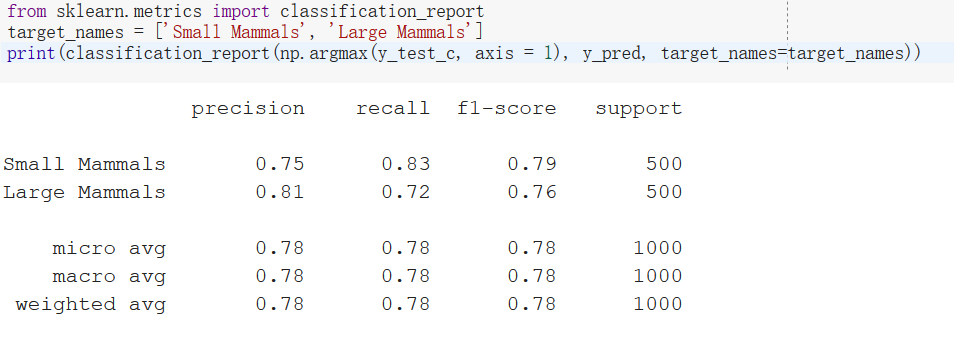
Training time for each epoch is around 56s. I ran a total of 76 epoches to reach the current model.

Model Evaluation:

Score:



Classification Report:



Neural network process for image recognition:

* Data set up:
  + Separate dataset so that you have two groups: predictors and target (labels or x)
  + One-hot encode your target set -> in order words, make sure that each category has its own column and is represented in a binary way. Code: *to\_categorical(x)*
  + If working with color images, you need to flatten the image from a RGB format (6000,32,32,3) to (6000, 3072) Code: *x.reshape(x.shape[0], -1)* Reference: <https://datascience-enthusiast.com/DL/Deep-Neural-Network-for-Image-Classification.html>
  + Splice the dataset to filter down to the assigned Superclass pair.
  + Validate the data by visualizing four random images from each sub-class.
  + Reference:
  + <https://medium.com/@birdortyedi_23820/deep-learning-lab-episode-5-cifar-100-a557e19219ba>; <https://nextjournal.com/mpd/image-classification-with-keras>
* Setup the neural network model:
  + Define your model. Code: *model = Sequential()*
  + Add input layer, making sure that that input\_shape is big enough to represent each pixel (in this case 32x32). Select your activation function (we chose ‘relu’, due to its flexibility) and number of neurons (determined through experimentation). Code: *model.add(Dense(500, activation="relu",input\_shape=(3072,)))*
  + Add the hidden layers. We used a formula suggested in stack overflow: # of layers = # of samples / (alfa \*(# neurons of outputs + # neurons of inputs)), where alpha is a arbitrary scaling factor, 2 to 10. Reference: <https://stats.stackexchange.com/questions/181/how-to-choose-the-number-of-hidden-layers-and-nodes-in-a-feedforward-neural-netw>
  + Deal with overfitting issues. Added a dropout layer. Reference: <https://www.tensorflow.org/tutorials/keras/overfit_and_underfit>
* Model compilation
  + Define your optimizer. We choose ‘adam’, given that it adapts it learning rate throughout the process and has a very flexible application
  + Change learning rate to avoid big jumps. Reference: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/
  + Define your loss function. For classification problems the recommended loss function is categorical\_crossentropy.
  + *Code: model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])*
* Model fit
  + Set up an early stop, so that the computer stops computing in case the model is no longer evolving. *Code: early\_stop=EarlyStopping(patience=2)*
  + Fit the model. Define your split for validation, we used 30% test and 70% train and the number of epochs (times the training goes through all the training data). *Code:hist= model.fit(x\_flatten, y\_target,validation\_split=0.3, epochs=20, callbacks=[early\_stop])*
* Evaluate the model and plot
  + Compare the predictions with the actual data. *Code: model.evaluate(x\_flatten, y\_target, verbose=0)*
  + Plot the results
* Salve model
  + To avoid the long training cycles, once we found a model we were comfortable with we saved it.
  + Code: model.save(‘nn\_model.h5’)

R3. List your team members and team name on the first page of both your report

R4. Provide all online links on example code you used/extended in the report.

R.5 Show a clear list on who collaborated with who to accomplish any bonus feature, if any.

R6. Add comments on each major step inside your scripts. Show references to any algorithm or reused code.

R7. Your report should contains at least the following in the order shown.

First page:

Group name and team members

Selected Machine Learning Algorithm

Google Colab link hosing your work

Second page

A who did what table similar to the following.

|  |  |  |
| --- | --- | --- |
| Task | Description | Names |
| Data Preparation | Load, pre-process and visualize data | Megha Rajam Rao, Dandan Zhao |
| ML methods | decision tree, random forest, extra decision tree, SVM with different four kernels, Ada boost classifier, gradient boosting classifier, KNN classifier, logistic classification | Dandan Zhao  CHING-MIN HU |
| Neural networks | Dense(feedforward) | Fernanda Bordin |
| Neural Networks | Convolutional neural network | Qiao Liu  Rajasree Rajendran |
| Powerpoint presentation | Input on Neural network  Input on Data preparation  Input on ML algorithms  Input on CNN | Fernanda Bordin  Dandan Zhao, Megha Rajam Rao  Dandan Zhao, CHING-MIN HU  Qiao Liu |
| Report (contributors) | Input on Neural network  Input on ML algorithms  Input on CNN | Fernanda Bordin  Dandan Zhao, CHING-MIN HU  Rajasree Rajendran, Megha Rajam Rao, Qiao Liu |
|  |  |  |
|  |  |  |
|  |  |  |