Astro AI: Galaxy Zoo

Training a Machine Learning Algorithm to predict specific structures of the galaxies

Ridhesh Goti and Illes Lohay

12 March 2021

Ludwig-Maximilians-Universität München
Universitäts-Sternwarte München

Overview

Objectives

Objective

- Classification of galaxies with regard to their structural features.

Motivation

- Enables further analysis of the processes happening within the galaxy.
- Determination of object properties within the galaxy (e.g. age and dynamics of stars) is possible.

Basic method

- Analysis of the galaxy recordings using machine learning algorithms.

Main galaxy types

- Galaxies can be roughly divided into three main categories:

Spiral galaxies

Figure: Spiral galaxy NGC1232, [ESO]

Elliptical galaxies

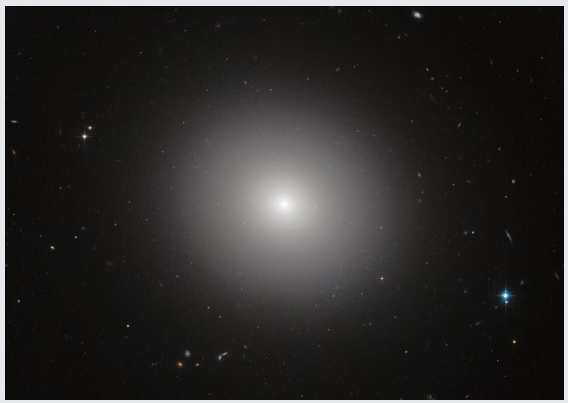


Figure: Elliptical galaxy IC2006, [ESO]

Irregular Galaxies

Figure: Irregular galaxy NGC1427, [ESO]

Hubble-Sequence

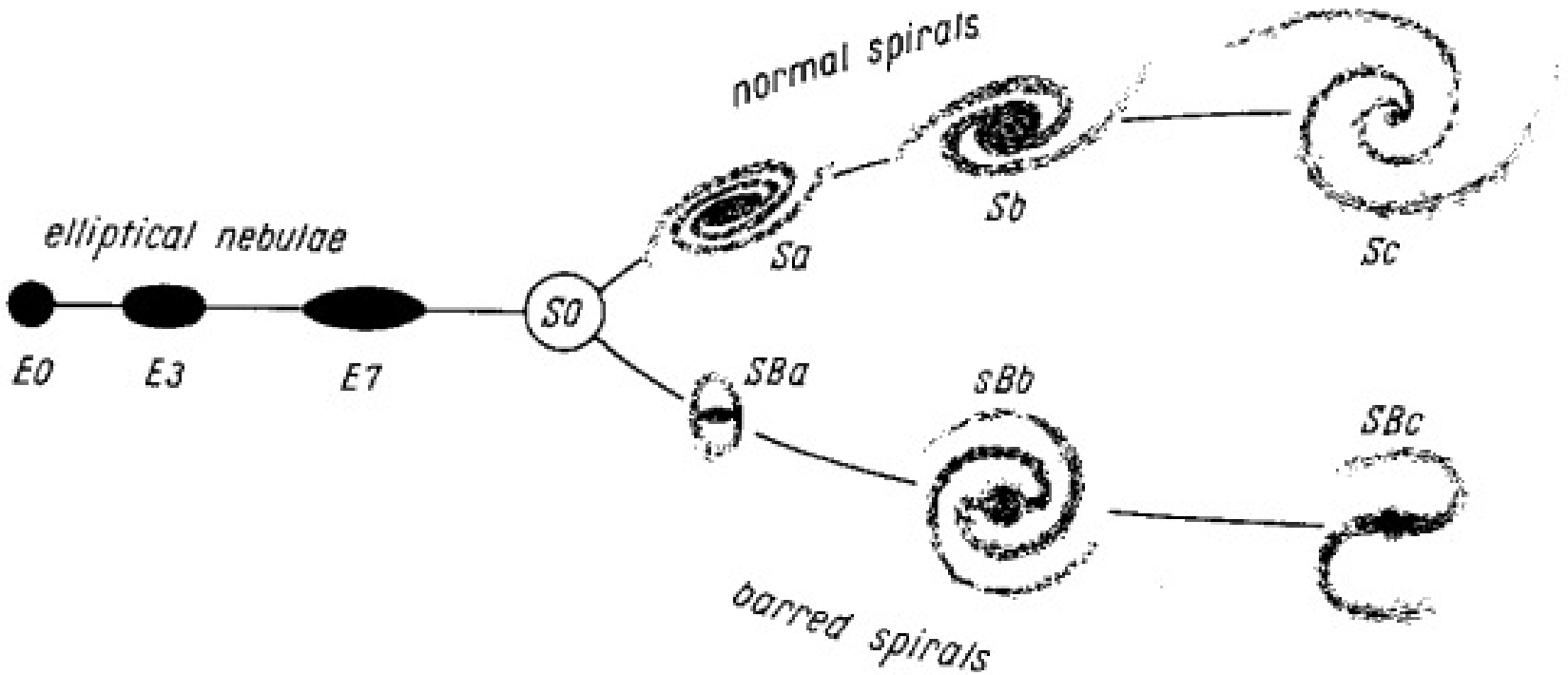


Figure: An original Hubble-Sequence. Also known as the "Tuning Fork".
[https://www.genesisnet.info/schoepfung_evolution/i42461_galaxien.php]

Hubble-Classification

Features of **spiral galaxies**

- Size of the bulge relatively to the disk/ the character of the spiral arms (**a, b, c**);
- Presence/absence of bars (**SA, SB**);
- Ratio of number of stars within HII regions to the number of stars in arms or/and disk.

Transition type SO

- Also known as 'armless spiral'.

Features of **elliptical galaxies**

- The class is described in terms of E_n;
- $n = \frac{(a-b)}{a} \cdot 10$ with a, b - major and minor **axis** of the galaxy;
- n can vary in between 0 (spherical) and 7 (lens).

Vaucouleurs-scheme

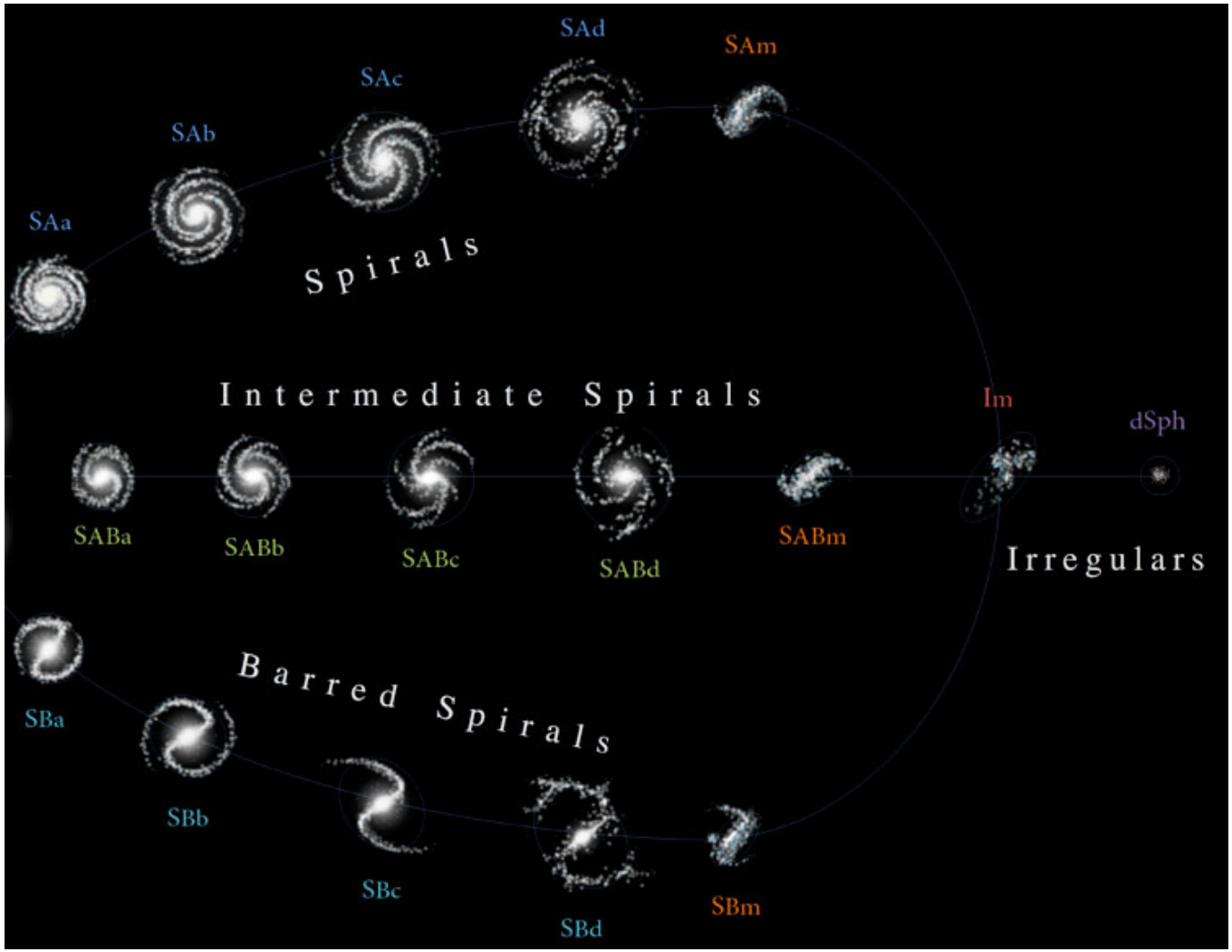


Figure: Vaucouleurs-scheme for spiral galaxies. [https://cseligman.com/text/devaucouleurs.htm]

Vaucouleurs-Classification: Extension of Hubble

Spiral galaxies

Opening of spiral arms and size of the bulge are crucial:

- Sizeable bulge, narrow wicked arms: Sa, Sb;
- Small bulge, open arms: Sc, Sd.

Also introduced different ring structures:

- S(s): without inner ring;
- S(r): with inner ring;
- RS: with outer ring.

Transition SO

- Varying from rather spiral to rather elliptical: $S0^{+}$, $S0^{0}$, $S0^{-}$.

Elliptical galaxies

- The classification follows the Hubble rules.

Images of galaxies

Data

All 62000 recordings of the galaxies are saved within the .zip-archive and are accessible on demand.

Figure: An example of the provided galaxy pictures.

List of theoretical classes

Features list

The following list of the galactic features is used to divide and classify the galaxies.

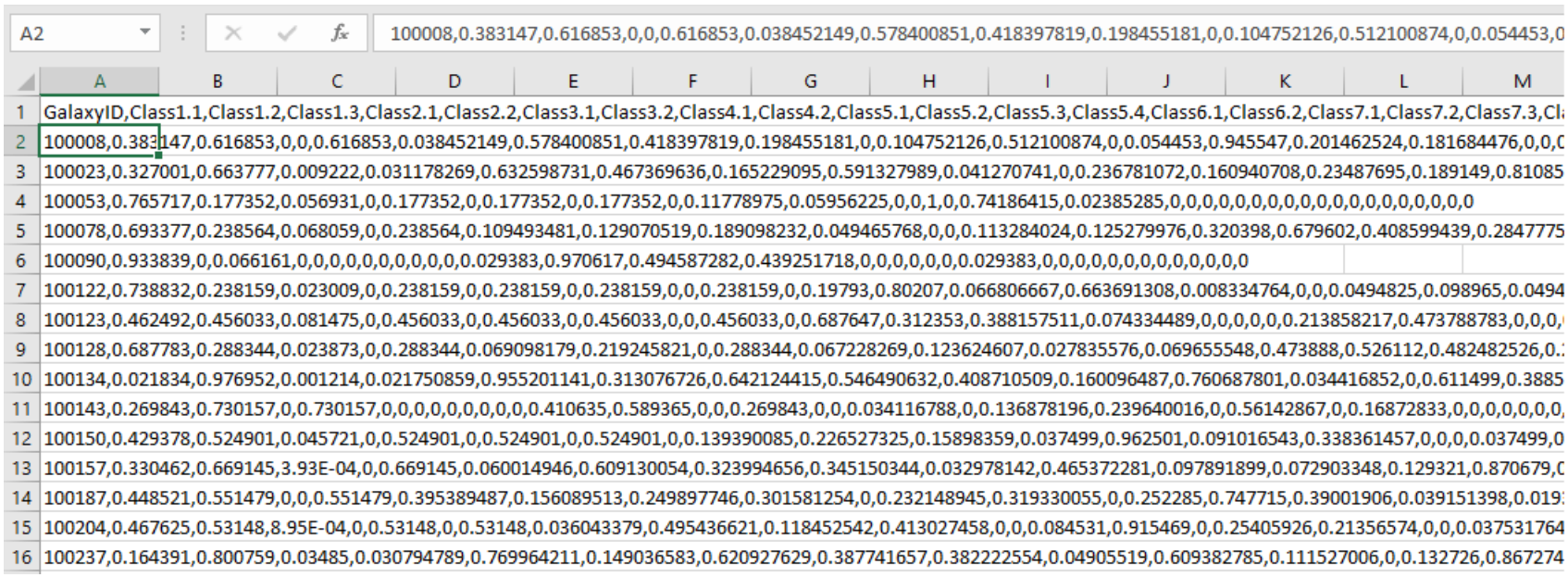
Task	Question	Responses	Next
01	*Is the galaxy simply smooth and rounded, with no sign of a disk?*	smooth features or disk star or artifact	07 02 **end**
02	*Could this be a disk viewed edge-on?*	yes no	09 03
03	*Is there a sign of a bar feature through the centre of the galaxy?*	yes no	04 04
04	*Is there any sign of a spiral arm pattern?*	yes no	10 05
05	*How prominent is the central bulge, compared with the rest of the galaxy?*	no bulge just noticeable obvious dominant	06 06 06 06
06	*Is there anything odd?*	yes no	08 **end**
07	*How rounded is it?*	completely round in between cigar-shaped	06 06 06
08	*Is the odd feature a ring, or is the galaxy disturbed or irregular?*	ring lens or arc disturbed irregular other merger dust lane	**end** **end** **end** **end** **end** **end** **end**
09	*Does the galaxy have a bulge at its centre? If so, what shape?*	rounded boxy no bulge	06 06 06
10	*How tightly wound do the spiral arms appear?*	tight medium loose	11 11 11
11	*How many spiral arms are there?*	1 2 3 4 more than four can't tell	05 05 05 05 05 05

Figure: The theoretical list of galactic features.

List of object labels

Determined classes

The final piece of provided data is the .csv-file (e.g. Excel) including the catalogue number and the corresponding class probability of the galaxy.



A2 | 100008,0.383147,0.616853,0,0,0.616853,0.038452149,0.578400851,0.418397819,0.198455181,0,0.104752126,0.512100874,0,0.054453,0

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	GalaxyID,Class1.1,Class1.2,Class1.3,Class2.1,Class2.2,Class3.1,Class3.2,Class4.1,Class4.2,Class5.1,Class5.2,Class5.3,Class5.4,Class6.1,Class6.2,Class7.1,Class7.2,Class7.3,Cl												
2	100008,0.383147,0.616853,0,0,0.616853,0.038452149,0.578400851,0.418397819,0.198455181,0,0.104752126,0.512100874,0,0.054453,0.945547,0.201462524,0.181684476,0,0,0												
3	100023,0.327001,0.663777,0.009222,0.031178269,0.632598731,0.467369636,0.165229095,0.591327989,0.041270741,0,0.236781072,0.160940708,0.23487695,0.189149,0.81085												
4	100053,0.765717,0.177352,0.056931,0,0.177352,0,0.177352,0,0.177352,0,0.11778975,0.05956225,0,0,1,0,0.74186415,0.02385285,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0												
5	100078,0.693377,0.238564,0.068059,0,0.238564,0.109493481,0.129070519,0.189098232,0.049465768,0,0,0.113284024,0.125279976,0.320398,0.679602,0.408599439,0.2847775												
6	100090,0.933839,0,0.066161,0,0,0,0,0,0,0,0,0,0.029383,0.970617,0.494587282,0.439251718,0,0,0,0,0,0.029383,0,0,0,0,0,0,0,0,0,0,0												
7	100122,0.738832,0.238159,0.023009,0,0.238159,0,0.238159,0,0.238159,0,0,0.238159,0,0.19793,0.80207,0.066806667,0.663691308,0.008334764,0,0,0.0494825,0.098965,0.0494												
8	100123,0.462492,0.456033,0.081475,0,0.456033,0,0.456033,0,0.456033,0,0,0.456033,0,0.687647,0.312353,0.388157511,0.074334489,0,0,0,0,0.213858217,0.473788783,0,0,0,												
9	100128,0.687783,0.288344,0.023873,0,0.288344,0.069098179,0.219245821,0,0.288344,0.067228269,0.123624607,0.027835576,0.069655548,0.473888,0.526112,0.482482526,0.:												
10	100134,0.021834,0.976952,0.001214,0.021750859,0.955201141,0.313076726,0.642124415,0.546490632,0.408710509,0.160096487,0.760687801,0.034416852,0,0.611499,0.3885												
11	100143,0.269843,0.730157,0,0.730157,0,0,0,0,0,0,0,0,0.410635,0.589365,0,0,0.269843,0,0,0.034116788,0,0.136878196,0.239640016,0,0.56142867,0,0.16872833,0,0,0,0,0,0,												
12	100150,0.429378,0.524901,0.045721,0,0.524901,0,0.524901,0,0.524901,0,0.139390085,0.226527325,0.15898359,0.037499,0.962501,0.091016543,0.338361457,0,0,0,0.037499,0												
13	100157,0.330462,0.669145,3.93E-04,0,0.669145,0.060014946,0.609130054,0.323994656,0.345150344,0.032978142,0.465372281,0.097891899,0.072903348,0.129321,0.870679,0												
14	100187,0.448521,0.551479,0,0,0.551479,0.395389487,0.156089513,0.249897746,0.301581254,0,0.232148945,0.319330055,0,0.252285,0.747715,0.39001906,0.039151398,0.019:												
15	100204,0.467625,0.53148,8.95E-04,0,0.53148,0,0.53148,0.036043379,0.495436621,0.118452542,0.413027458,0,0,0.084531,0.915469,0,0.25405926,0.21356574,0,0,0.037531764												
16	100237,0.164391,0.800759,0.03485,0.030794789,0.769964211,0.149036583,0.620927629,0.387741657,0.382222554,0.04905519,0.609382785,0.111527006,0,0.132726,0.867274												

Figure: An example of an object class specification.

Redefining the images and labels

Procedure

The science notebook of Deepnote have been used to write and execute the code.

- Cell #1: Imported multiple packages;
- Cell #2: Loaded the labels and the images;
- Cell #3: Converted the labels into the classes (i.e. if the label has three classes 7.1, 7.2, 7.3 then it would assign the most probable class as 1 and rest of them as 0.);
- Cell #4: Split data into the train, test, and validation set;
- Cell #5: Reshaped the image (from 200 X 200 to 40000)
- Cell #6: Principal component analysis for the dimensionality reduction (optional).

Code

```
from sklearn.ensemble import RandomForestClassifier
from sklearn import svm
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from keras.layers import Dense, Dropout, Conv2D, MaxPool2D
from sklearn.linear_model import SGDClassifier
```

```
img = ioi.imread('/content/images_training_rev2/images_training_rev1/'+i)
im = img[110:310,110:310]
img_gray = rgb2gray(im)
imgs.append(img_gray)
....

data = pd.read_csv('/content/galaxy_zoo_labels.csv')
```

```
test_ratio = 0.4
train_set, test_set, img_train, img_test = train_test_split(class11,imgs, test_size=test_ratio,
                                                            random_state=160982)
train_set, valid_set, img_train, img_valid = train_test_split(train_set,img_train, test_size=test_ratio,
                                                              random_state=160982)
```

```
c5 = list(data.iloc[i][['Class5.1','Class5.2','Class5.3','Class5.4']])
c6 = list(data.iloc[i][['Class6.1','Class6.2']])
.......
```

```
imtrain = img_train.reshape(img_train.shape[0],40000)
imtest = img_test.reshape(img_test.shape[0],40000)
imvalid = img_valid.reshape(img_valid.shape[0],40000)
```

```
pca = de.PCA(0.94)
imtrainp = pca.fit_transform(imtrain)
imtestp  = pca.transform(imtest)
imvalidp  = pca.transform(imvalid)
```

Figure: The code for redefining the images and the labels.

Analyzed/Depicted Features I

Regularisation

Regularisation - the process of constraining the model in order to simplify it, increase accuracy and prevent overfitting.

- For SVM: readjusting hyperparameter c;
- For SGD: the maximum number of passes over the training data *max_iter*;
- For RF: the number of trees in the forest *n_estimators*.

Learning Curve

Learning Curve is a plot depicting the validation and training score for varyying number of training samples.

Analyzed/Depicted Features II

Hyperparameter Tuning

Hyperparameter Tuning - the process of choosing an optimal set of hyperparameters for the learning algorithm.

- For SVM: C, *gamma*, *kernel*;
- For SGD: *max_iter*, *penalty*, *loss*;
- For RF: *n_estimators*, *depth*.

Precision-Recall Tradeoff

Precision-Recall Tradeoff - the process where precision and recall are varyying on cost of each other.

- Precision: fraction of true positive numbers (TP) and total predicted positive numbers (TP+FP);
- Recall: fraction of positive numbers (TP) and total positive numbers (TP+FN).

Problems: noise; non-separable data.

Support Vector Machines-Intro

Art of learning

- Supervised learning - Classification, Regression.

Advantages

- Effective in high dimensional spaces;
- Effective for clear differences between the classes;
- Different kernel functions: Linear, Polynomial, rbf, sigmoid;
- Each of the kernels has parameters (Regularization and Hyper-parameters tuning).

Disadvantages

- Condition: Number of dimensions $\leq$ Number of samples

Support Vector Machines-Code-Structure

```
clf = svm.SVC(kernel='rbf', gamma=0.0001, C=10)
#Train the model using the training sets
clf.fit(imtrainp, train_set)
#Predict the response for test dataset
y_pred = clf.predict(imtestp)
```

Figure: The code for for support vector machine (SVM).

Support Vector Machines-Regularization

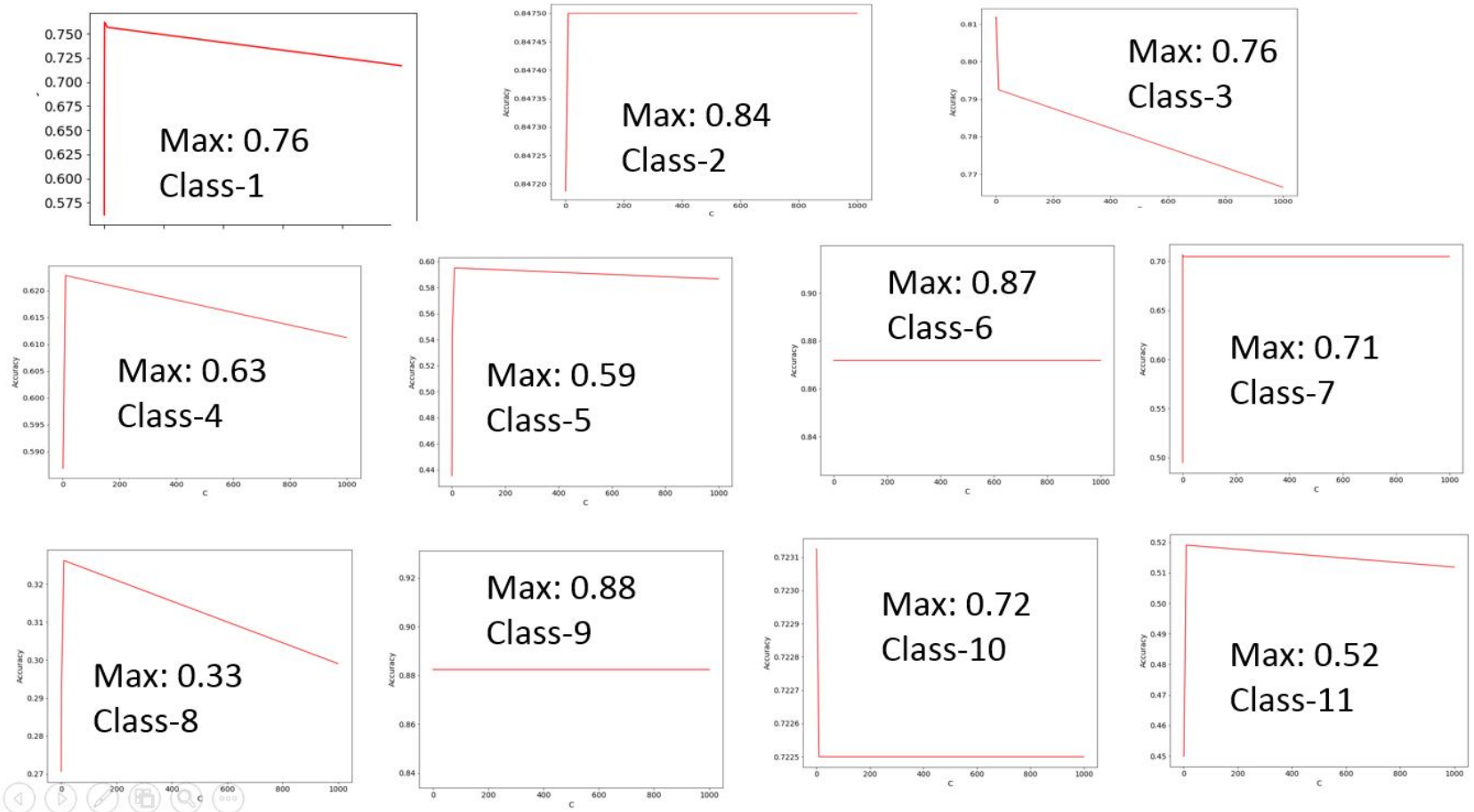


Figure: The final results for the regularization of SVM.

Support Vector Machines-Learning Curve

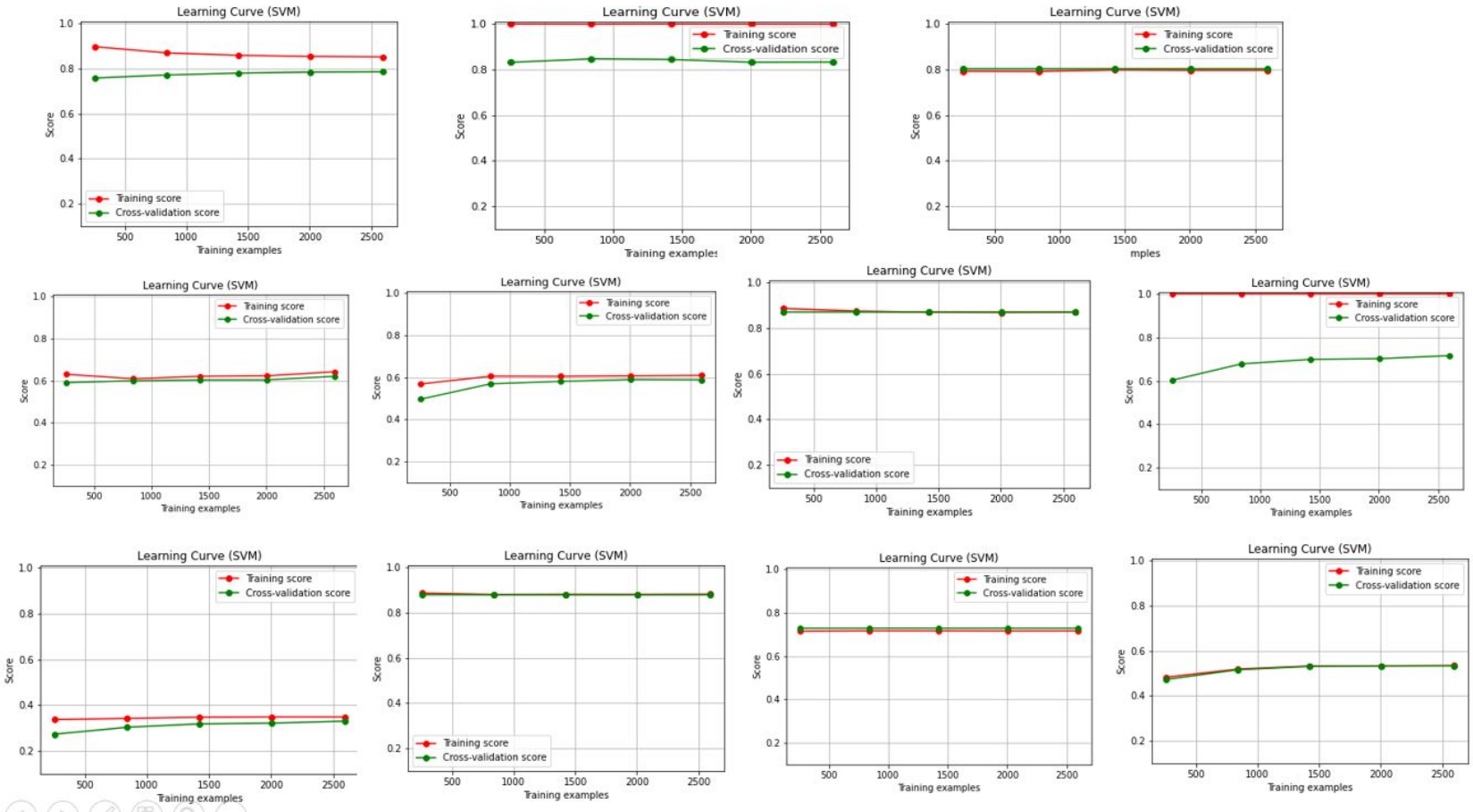


Figure: Learning curve of the SWM method.

Support Vector Machines-Hyperparameters tuning

Using Validation set	[0.1,1000]	[0.0001,1]	[rbf,polynomial,linear]
	C	gamma	kernel
class-1	10	0.001	rbf
class-2	0.1	0.1	poly
class-3	0.1	0.01	rbf
class-4	10	0.0001	rbf
class-5	10	0.0001	rbf
class-6	0.1	0.1	rbf
class-7	10	0.1	poly
class-8	10	0.0001	rbf
class-9	0.1	0.1	rbf
class-10	0.1	0.1	rbf
class-11	10	0.0001	rbf

Figure: Hyperparameter tuning of the SWM method.

Support Vector Machines-Recall

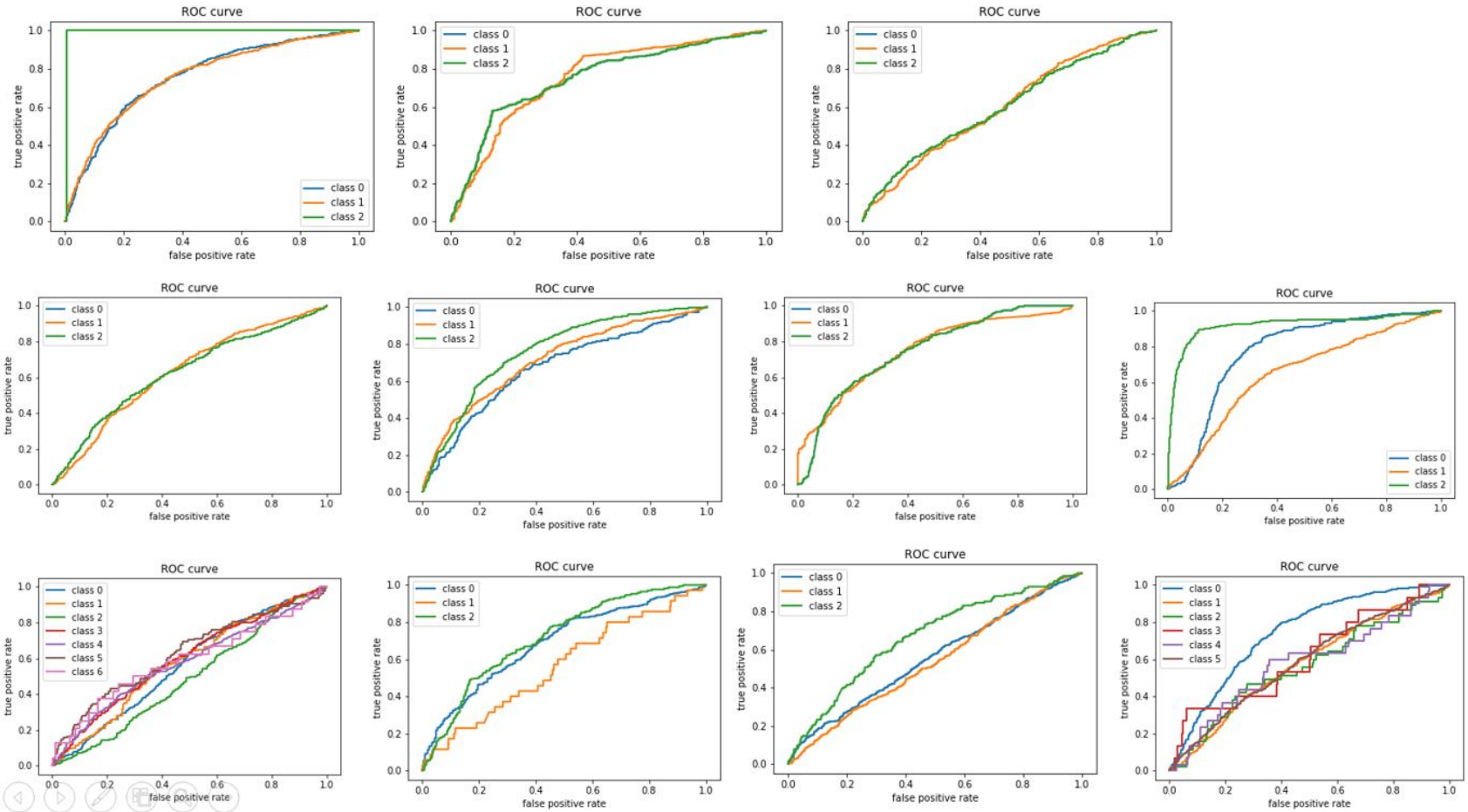


Figure: Recall of the SVM

Support Vector Machines-Precision-Recall

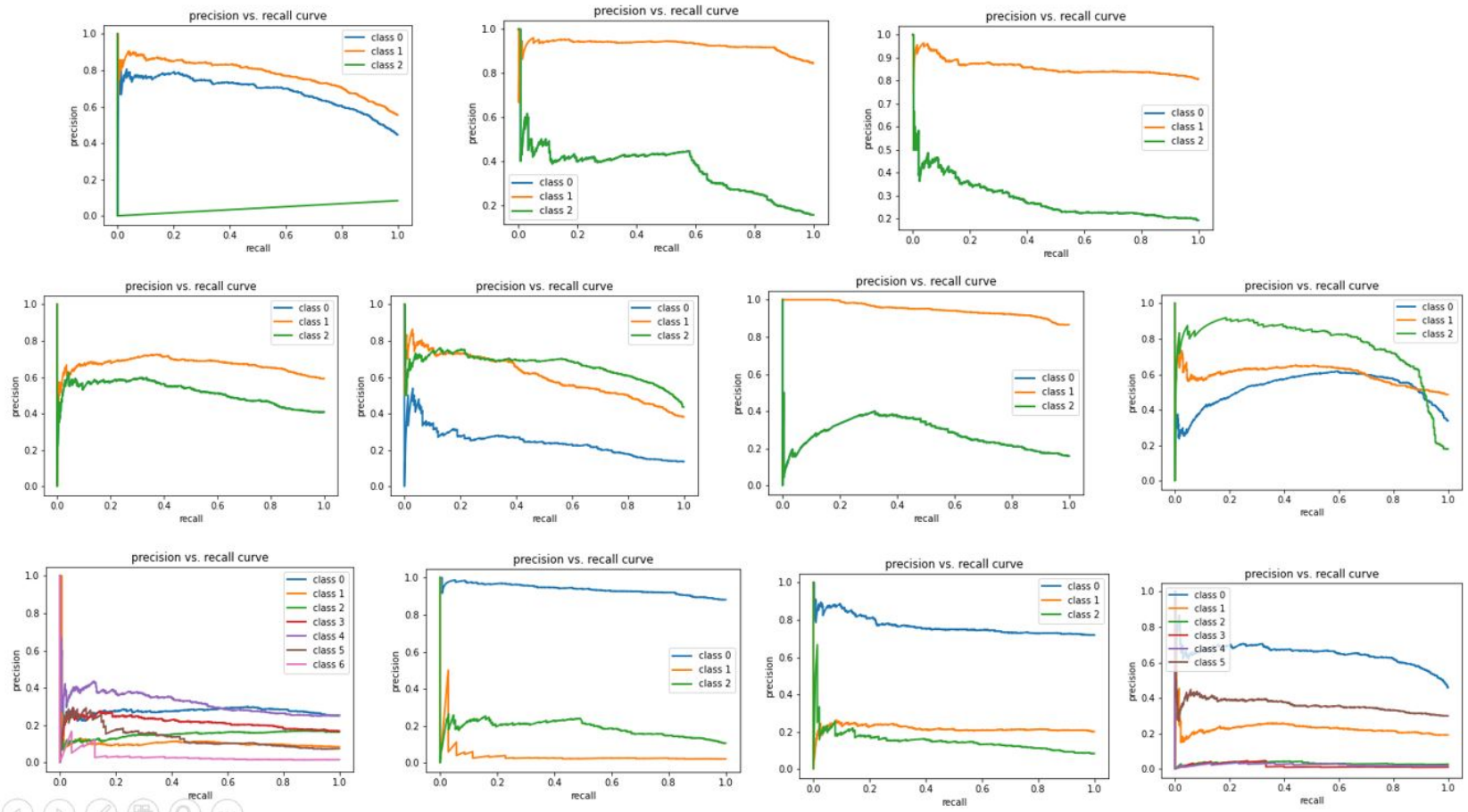


Figure: Precision of the SWM

Support Vector Machines-Final Score

```
The final score is : 0.7172697143090985
Accuracy: 0.716875
Precision: 0.7180507307581849
Recall: 0.716875
```
Class-1

```
The final score is : 0.777857332780828
Accuracy: 0.8475
Precision: 0.8707635198499531
Recall: 0.8475
```
Class-2

```
The final score is : 0.7271443203381059
Accuracy: 0.8115625
Precision: 0.6586336914062499
Recall: 0.8115625
```
Class-3

```
The final score is : 0.5844333676079692
Accuracy: 0.6228125
Precision: 0.6145984034150475
Recall: 0.6228125
```
Class-4

```
The final score is : 0.533853671256137
Accuracy: 0.595
Precision: 0.4886422566620533
Recall: 0.595
```
Class-5

```
The final score is : 0.8121974123539232
Accuracy: 0.871875
Precision: 0.760166015625
Recall: 0.871875
```
Class-6

```
The final score is : 0.7057077573987166
Accuracy: 0.7046875
Precision: 0.7137105970777696
Recall: 0.7046875
```
Class-7

```
The final score is : 0.2503517562657631
Accuracy: 0.32625
Precision: 0.2296869724861058
Recall: 0.32625
```
Class-8

```
The final score is : 0.8274169986719788
Accuracy: 0.8825
Precision: 0.7788062499999999
Recall: 0.8825
```
Class-9

```
The final score is : 0.6069319006166123
Accuracy: 0.723125
Precision: 0.522909765625
Recall: 0.723125
```
Class-10

```
The final score is : 0.4292537501040175
Accuracy: 0.5190625
Precision: 0.3766086274638301
Recall: 0.5190625
```
Class-11

Figure: Final score of the SWM

Random Forest Classifier

- Fits a number of decision tree classifiers
- Parameters to define a tree structure: max_depth, n_estimators
- Classifying through 'yes'-'no' questions (i.e. Is the galaxy rounded? If yes then does the galaxy have ring, irregular, etc?)
- Time-consuming with n-estimators

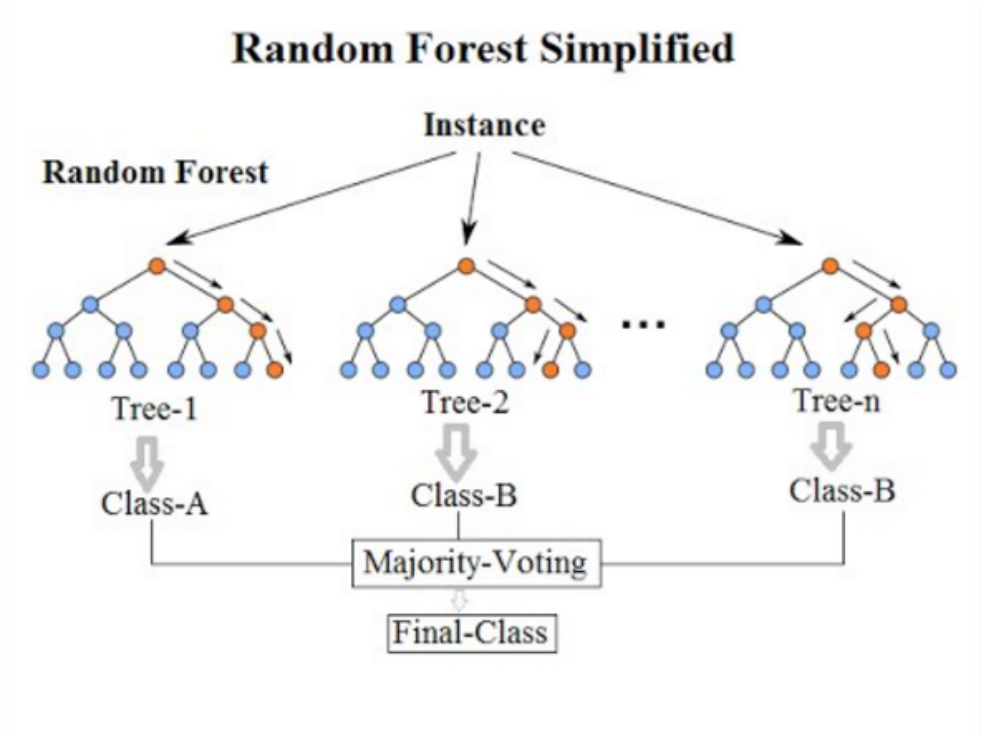


Figure: Random forest tree structure

Random Forest Classifier-Code-Structure

```
clf = RandomForestClassifier(n_estimators=10,max_depth=5,random_state=0)

clf.fit(imtrainp, train_set)
y_pred = clf.predict(imtestp)
```

Figure: The code for for random forest (rf).

Random Forest Classifier-Regularization

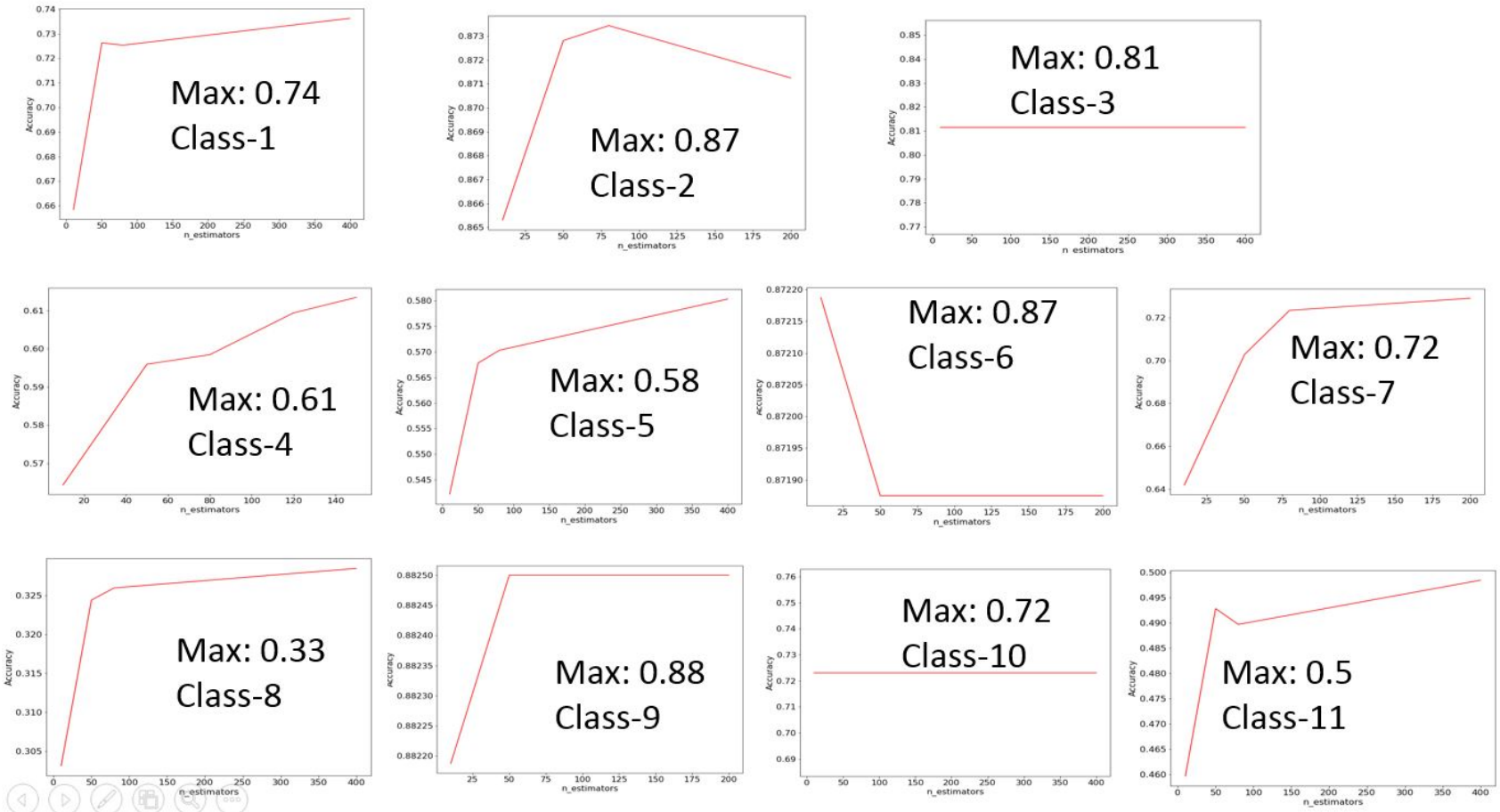


Figure: The final results for the regularization of RF.

Random Forest Classifier-Learning Curve

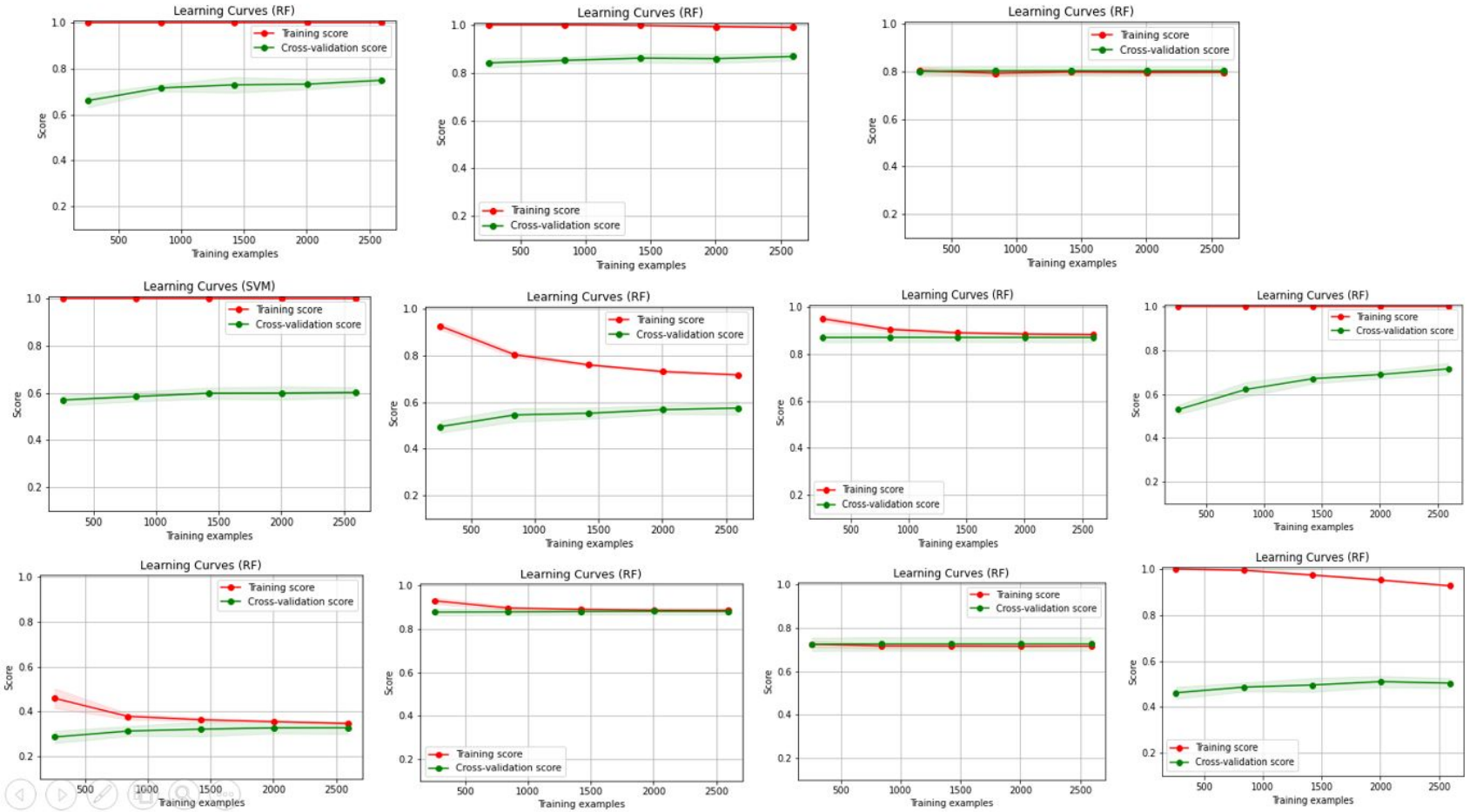


Figure: The final results for the learning curve of RF.

Random Forest Classifier-Hyperparameters tuning

Column1	[1,2,4]	[5,15,25]
Hyper-parameters	n_estimators	depth
class-1	300	20
class-2	120	20
class-3	10	2
class-4	150	15
class-5	200	5
class-6	50	5
class-7	120	20
class-8	60	2
class-9	10	5
class-10	10	2
class-11	120	10

Figure: Hyperparameter tuning of the RF method.

Random Forest Classifier-Recall

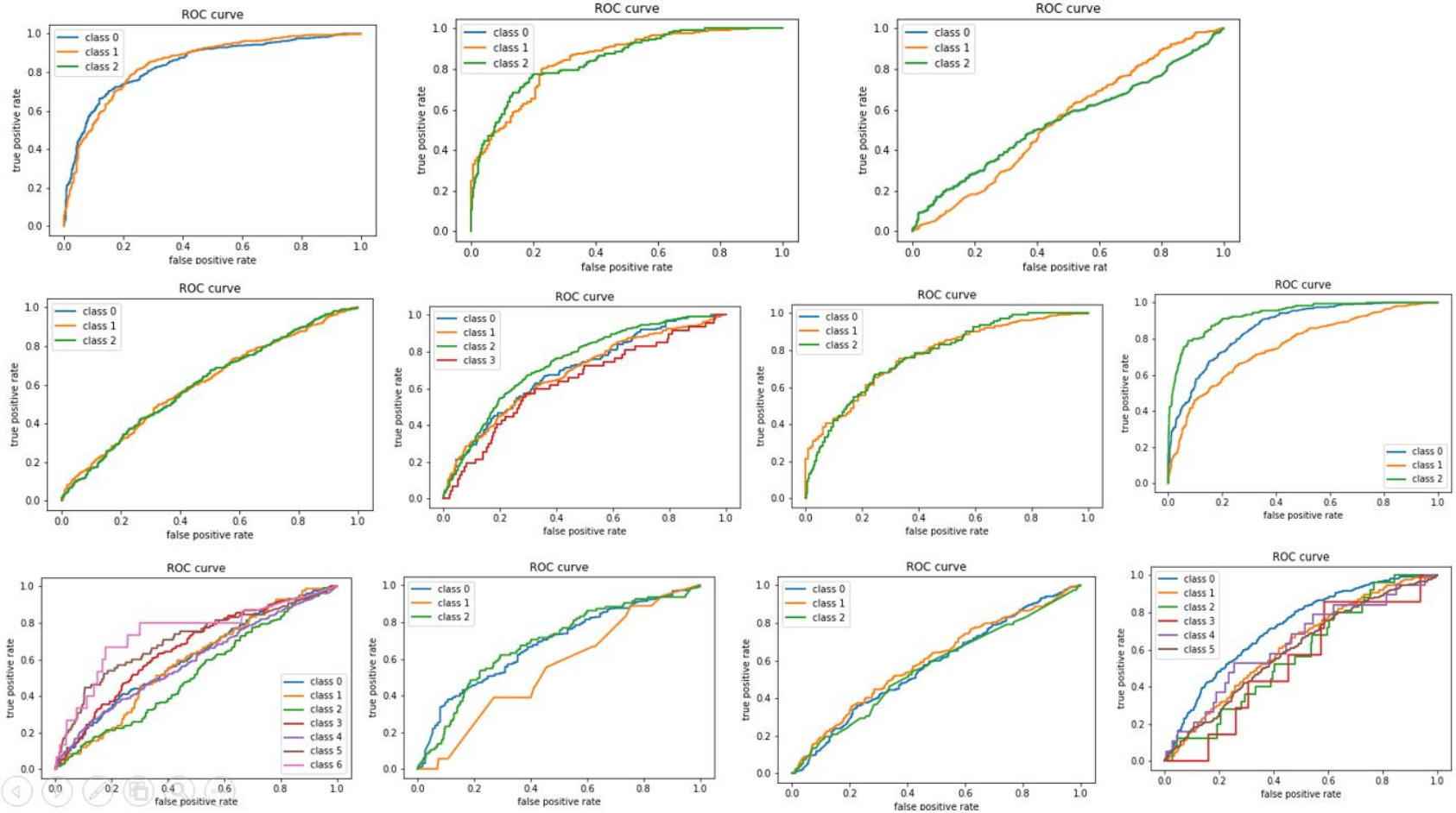


Figure: Recall of the RF

Random Forest Classifier-Precision-Recall

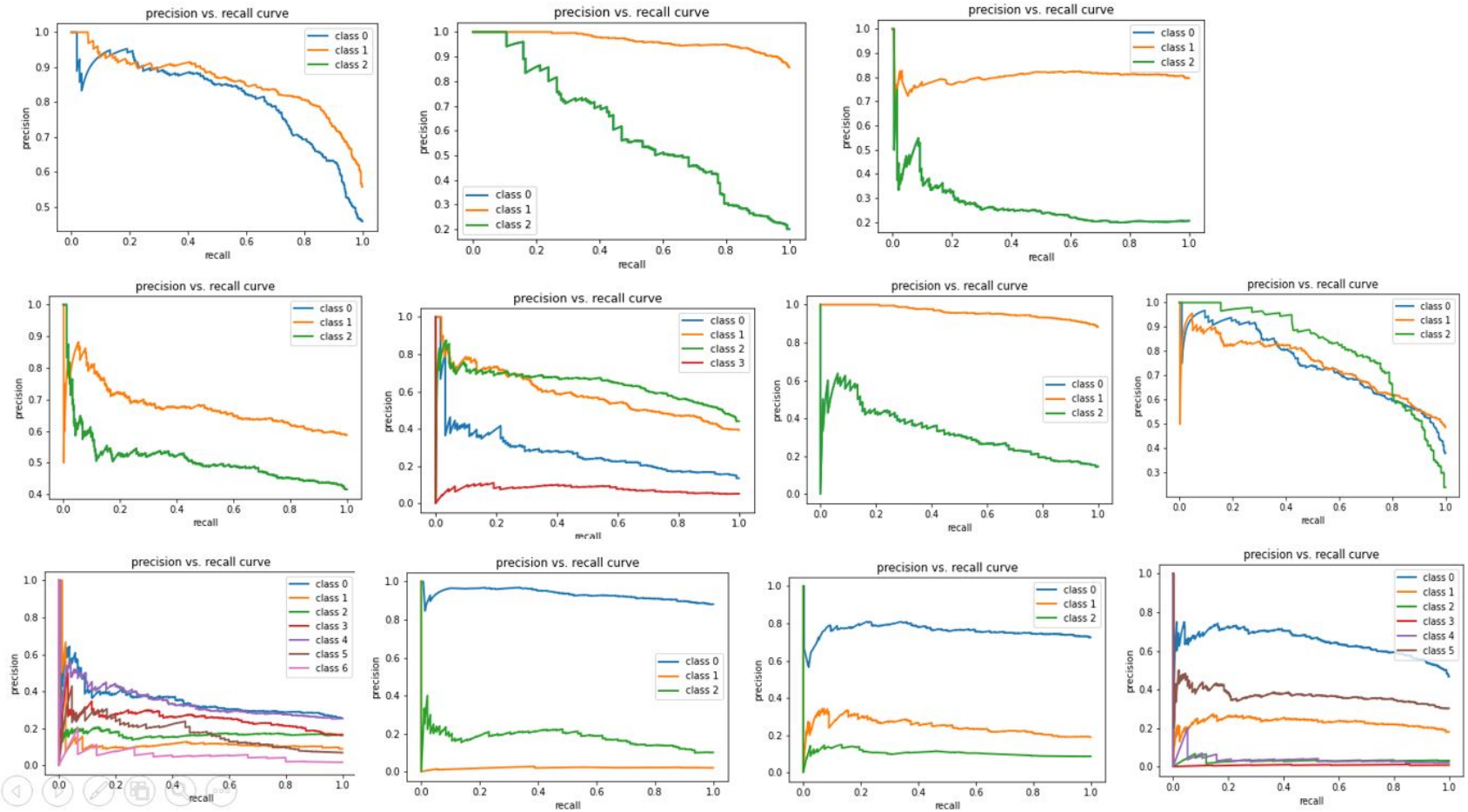


Figure: Precision-Recall of RF.

Random Forest Classifier-Final Score

```
The final score is : 0.7247495209821048
Accuracy: 0.73625
Precision: 0.7492049925785363
Recall: 0.73625
```
Class-1

```
The final score is : 0.8308910680954166
Accuracy: 0.8709375
Precision: 0.8775171154876742
Recall: 0.8709375
```
Class-2

```
The final score is : 0.7271443203381059
Accuracy: 0.8115625
Precision: 0.6586336914062499
Recall: 0.8115625
```
Class-3

```
The final score is : 0.5553834667101476
Accuracy: 0.6134375
Precision: 0.6071107983546509
Recall: 0.6134375
```
Class-4

```
The final score is : 0.520929472293134
Accuracy: 0.5803125
Precision: 0.474832192364789
Recall: 0.5803125
```
Class-5

```
The final score is : 0.8121974123539232
Accuracy: 0.871875
Precision: 0.760166015625
Recall: 0.871875
```
Class-6

```
The final score is : 0.7270378326195502
Accuracy: 0.7359375
Precision: 0.7689955722219028
Recall: 0.7359375
```
Class-7

```
The final score is : 0.22108926343542434
Accuracy: 0.3271875
Precision: 0.1719026600559579
Recall: 0.3271875
```
Class-8

```
The final score is : 0.8272613315623444
Accuracy: 0.8821875
Precision: 0.7787738355736167
Recall: 0.8821875
```
Class-9

```
The final score is : 0.6069319006166123
Accuracy: 0.723125
Precision: 0.522909765625
Recall: 0.723125
```
Class-10

```
The final score is : 0.39355228235873313
Accuracy: 0.49625
Precision: 0.43335455504612247
Recall: 0.49625
```
Class-11

Figure: The final score for RF.

Stochastic Gradient Descent

- Uses derivatives for training the sample
- Finds a gradient of the loss by a function: log, hinge, modified_huber
- Penalty: l1, l2, elastic net
- Regularisation: max_iter
- It has many hyper parameters to tune

Stochastic Gradient Descent-Code-Structure

```
clf = SGDClassifier(loss="hinge", penalty="elasticnet", max_iter=10)

clf.fit(imtrainp, train_set)
#Predict the response for test dataset
y_pred = clf.predict(imtestp)
```

Figure: The code for for stochastic gradient descent (sgd).

Stochastic Gradient Descent-Regularization

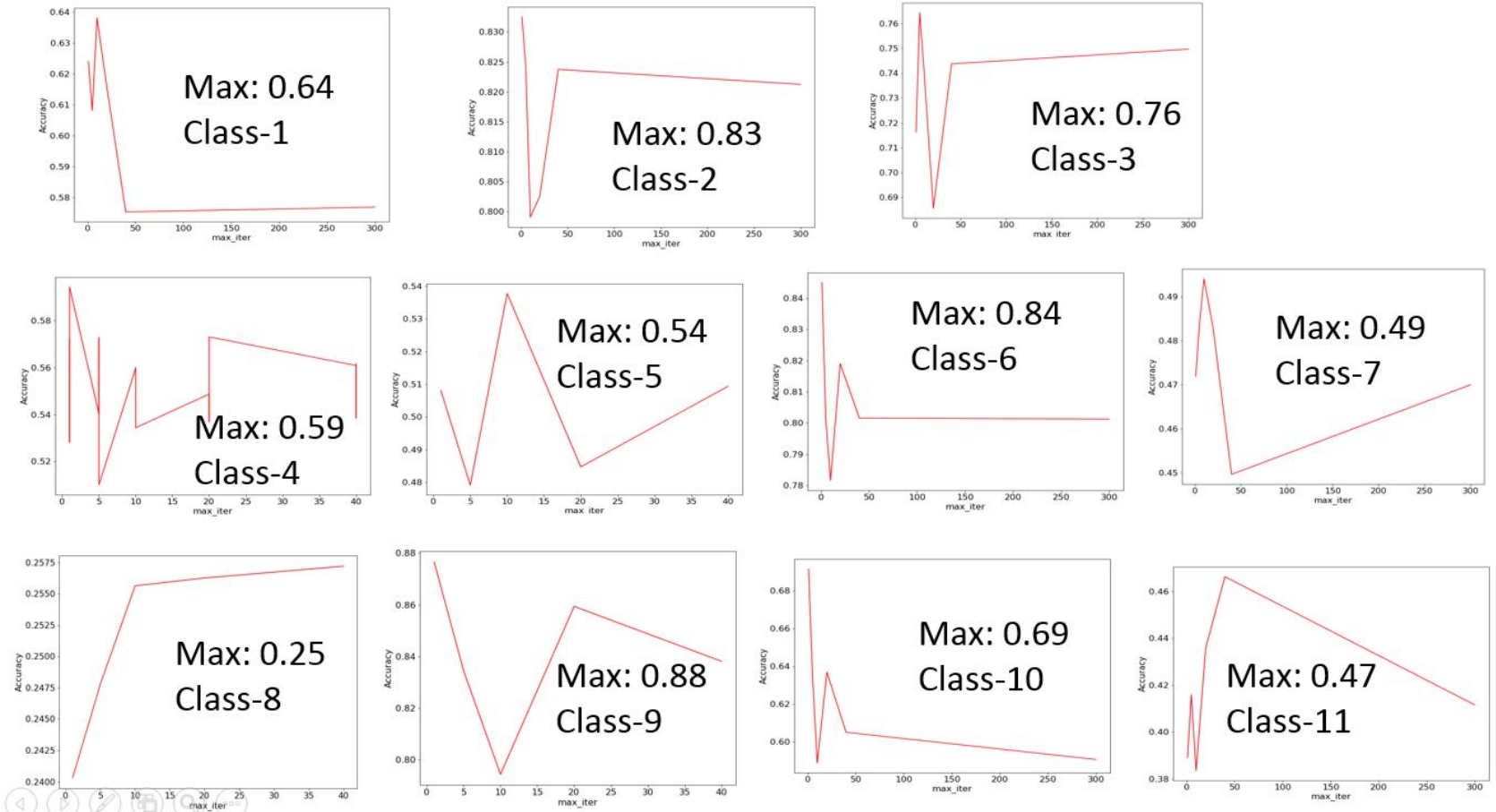


Figure: Regularisation of the stochastic gradient descent.

Stochastic Gradient Descent-Learning Curve

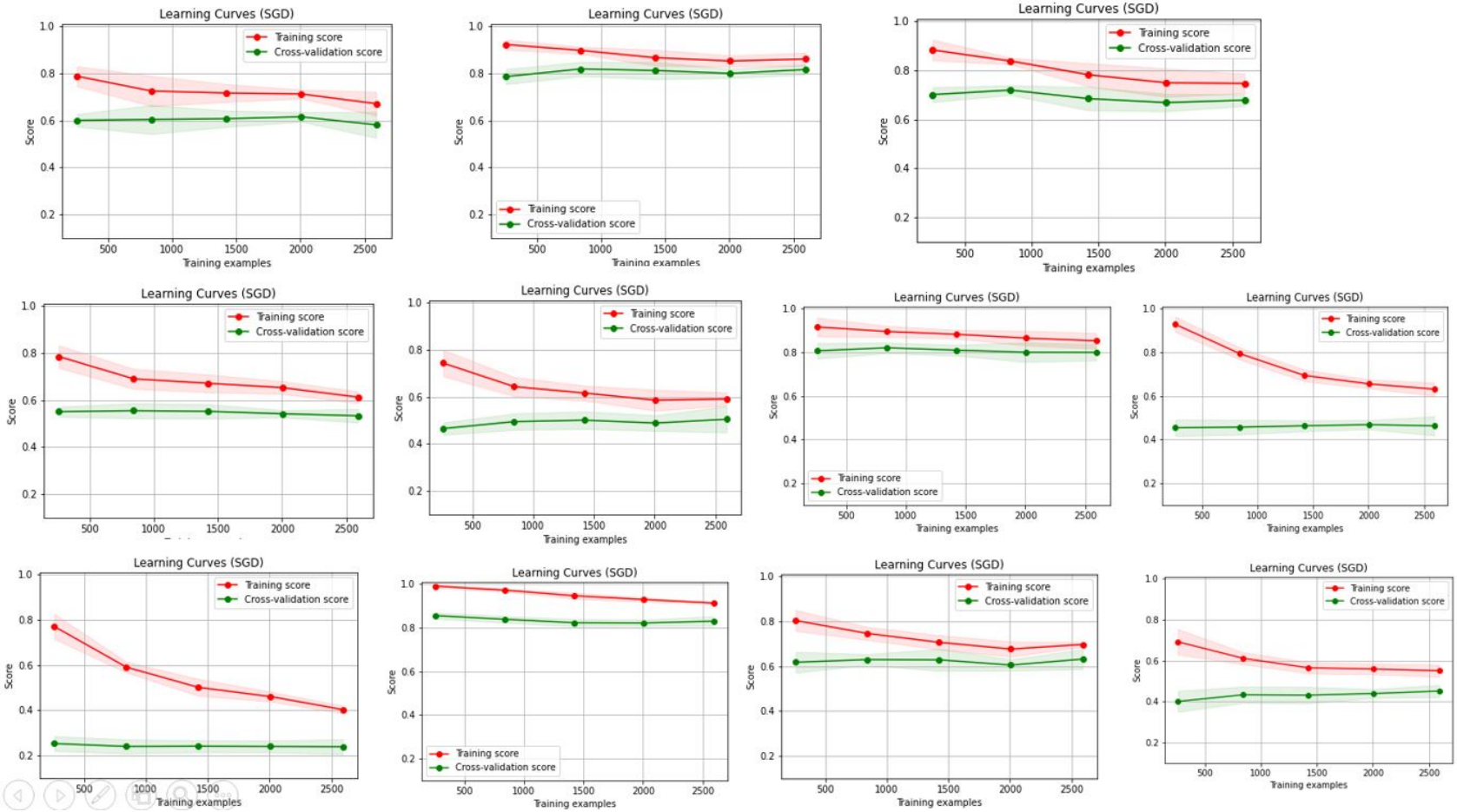


Figure: Learning curve of the stochastic gradient descent.

Stochastic Gradient Descent-Hyperparameters tuning

Column1	[5,10,20,50]	[l1,l2,elasticnet]	[hinge,log,modified_huber]
Hyper-parameters	max_iter	penalty	loss
class-1	5	elasticnet	hinge
class-2	10	l2	modified_huber
class-3	10	elasticnet	hinge
class-4	10	l2	modified_huber
class-5	5	l2	modified_huber
class-6	5	l1	log
class-7	20	l2	log
class-8	10	l2	hinge
class-9	5	elasticnet	log
class-10	5	l2	modified_huber
class-11	5	l2	log

Figure: Hyperparameter tuning of the SGD method.

Stochastic Gradient Descent-Recall

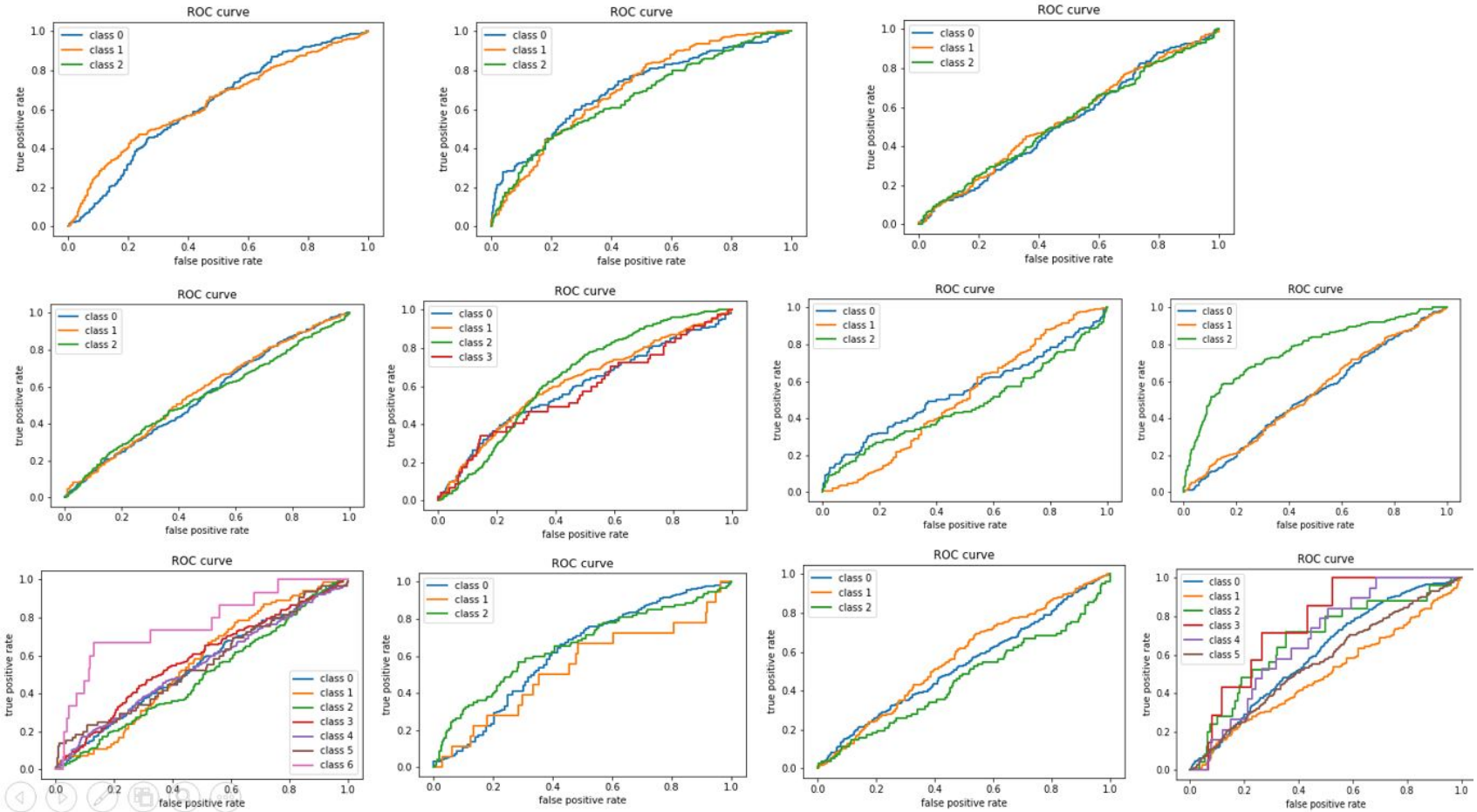


Figure: Recall of the SGD

Stochastic Gradient Descent-Precision-Recall

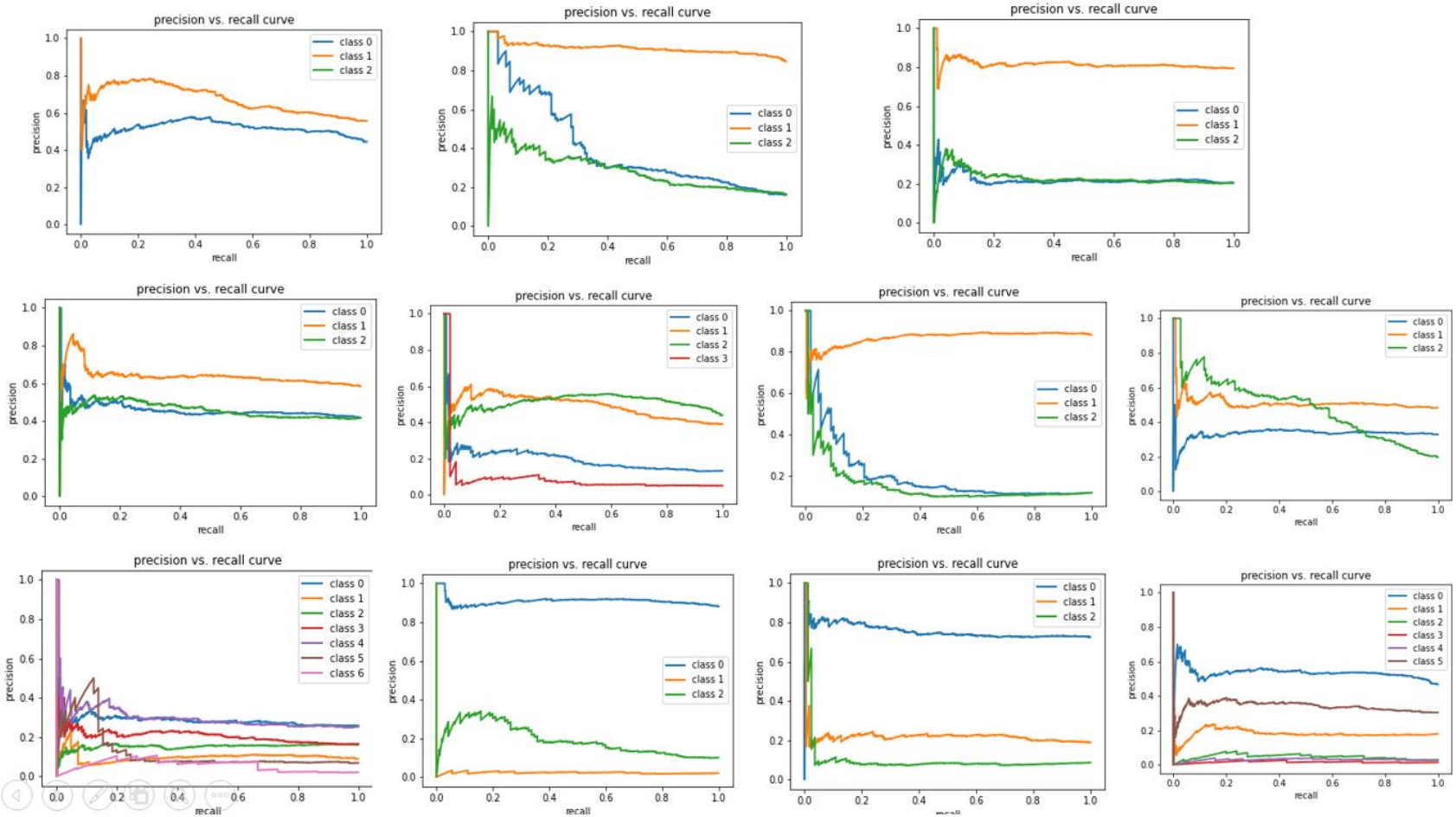


Figure: Precision of the SGD.

Stochastic Gradient Descent-Final Score

```
The final score is : 0.6220259261104093
Accuracy: 0.625
Precision: 0.6216359408046825
Recall: 0.625
```
Class-1

```
The final score is : 0.8142156043234335
Accuracy: 0.831875
Precision: 0.8044694251331781
Recall: 0.831875
```
Class-2

```
The final score is : 0.7158841864305614
Accuracy: 0.7353125
Precision: 0.7002556963296171
Recall: 0.7353125
```
Class-3

```
The final score is : 0.5799442770318382
Accuracy: 0.5775
Precision: 0.5845692684170839
Recall: 0.5775
```
Class-4

```
The final score is : 0.4658511158811068
Accuracy: 0.4809375
Precision: 0.45983223955704583
Recall: 0.4809375
```
Class-5

```
The final score is : 0.813550430677173
Accuracy: 0.831875
Precision: 0.7995002704563334
Recall: 0.831875
```
Class-6

```
The final score is : 0.47048913162895545
Accuracy: 0.4696875
Precision: 0.4729286273289492
Recall: 0.4696875
```
Class-7

```
The final score is : 0.23387181076020547
Accuracy: 0.2384375
Precision: 0.24982953721830847
Recall: 0.2384375
```
Class-8

```
The final score is : 0.8175338613406795
Accuracy: 0.838125
Precision: 0.799435800459594
Recall: 0.838125
```
Class-9

```
The final score is : 0.6055821649079509
Accuracy: 0.626875
Precision: 0.5905372394132209
Recall: 0.626875
```
Class-10

```
The final score is : 0.41109794511241593
Accuracy: 0.430625
Precision: 0.40152014668222635
Recall: 0.430625
```
Class-11

Figure: Final score of the SGD.

Convolutional Neural Networks

- Convolutional neural network is the best for the image classification
- Convoulution2D layer, MaxPooling2D, Flatten, Dense layers, Compile
- Activation functions: Sigmoid, TanH, ReLU, SoftMax
- Regularisation: Dropout
- Hype-parameters: learning rate, epoch, etc.

Convolutional Neural Networks-Code-Structure I

```
cnn = tf.keras.models.Sequential()
cnn.add(tf.keras.layers.Conv2D(4, kernel_size=(3,3), strides=1, padding="same", activation='relu', input_shape=(200,200,1)))
cnn.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2), strides=2))
cnn.add(tf.keras.layers.Activation('relu'))
cnn.add(tf.keras.layers.Conv2D(8, kernel_size=(3,3), strides=1, activation='relu', padding="valid"))
cnn.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2), strides=2))
cnn.add(tf.keras.layers.Activation('relu'))
cnn.add(tf.keras.layers.Conv2D(32, kernel_size=(3,3), strides=1, activation='relu', padding="valid"))
cnn.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2), strides=2))
cnn.add(tf.keras.layers.Activation('relu'))
cnn.add(tf.keras.layers.Conv2D(64, kernel_size=(3,3), strides=1, activation='relu', padding="valid"))
cnn.add(tf.keras.layers.AveragePooling2D(pool_size=(2,2), strides=2))
cnn.add(tf.keras.layers.Activation('relu'))
cnn.add(tf.keras.layers.Dropout(rate=0.2, seed=42))
cnn.add(tf.keras.layers.Flatten())
cnn.add(tf.keras.layers.Dense(128, activation='relu'))
cnn.add(tf.keras.layers.Dropout(rate=0.2, seed=42))
cnn.add(tf.keras.layers.Dense(7, activation='softmax'))
cnn.summary()
```

Figure: First part of code for convolutional neural networks.

Convolutional Neural Networks-Code-Structure II

```
adam = tf.keras.optimizers.Adam(lr=0.04)
cnn.compile(loss='categorical_crossentropy', optimizer=adam, metrics=['accuracy'])
```

```
#Fit the model
history = cnn.fit(np.array(img_train), np.array(train_set), epochs=5, batch_size=200, verbose=1,
                  validation_data=(np.array(img_test),np.array(test_set)))
```

Figure: Second part of code for convolutional neural networks.

Convolutional Neural Networks-Learning Rate

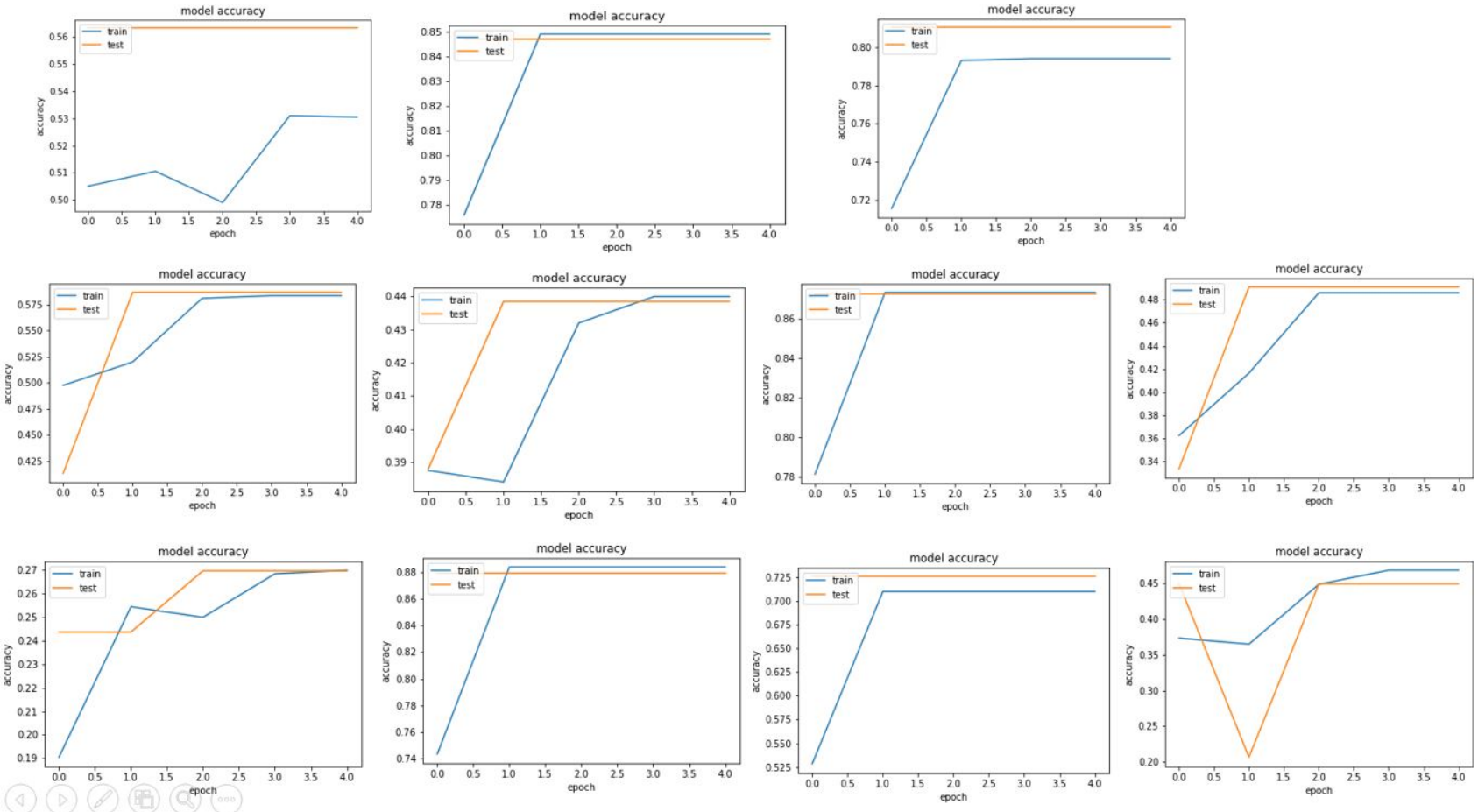


Figure: Learning rate for cnn.

Convolutional Neural Networks-Loss Rate

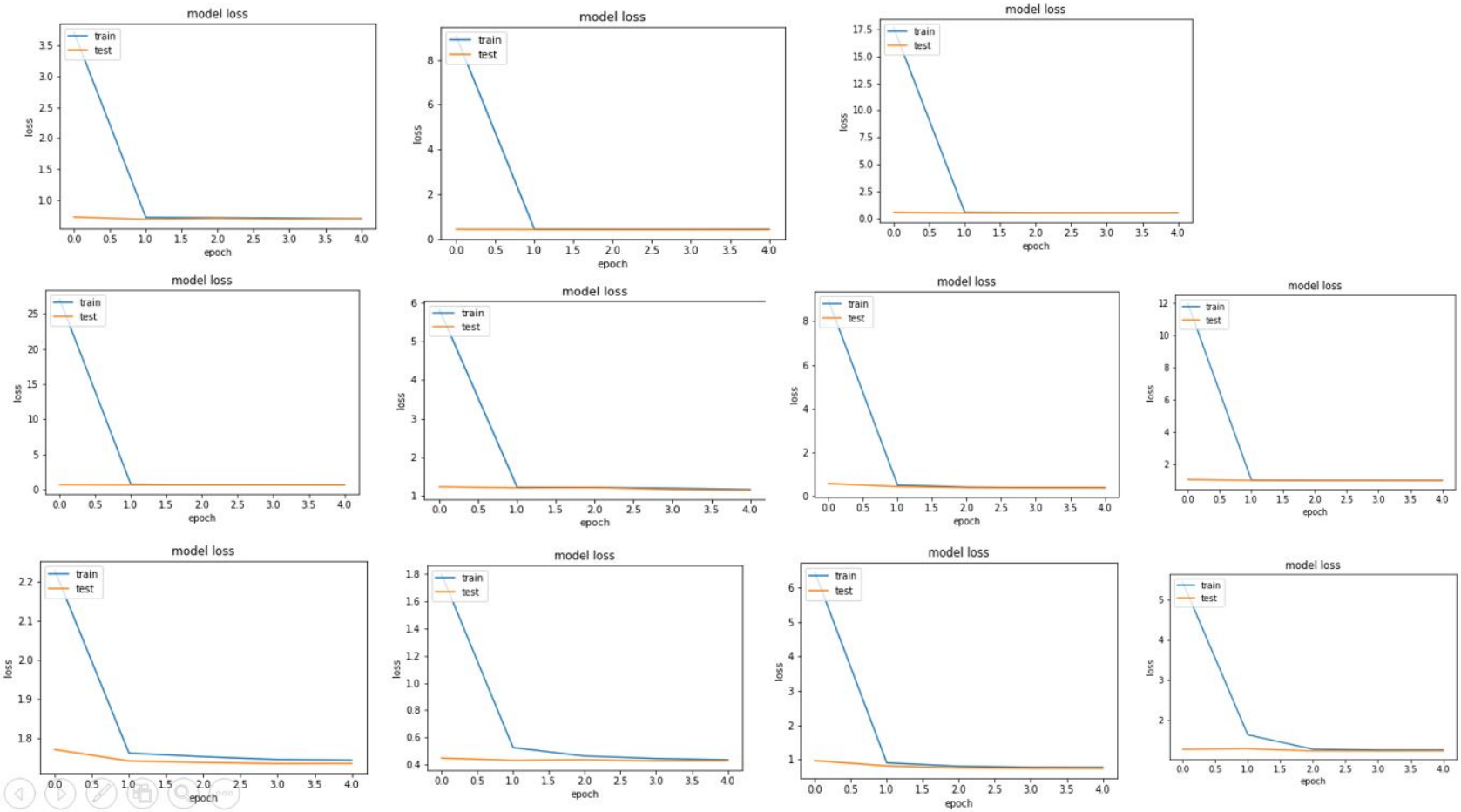


Figure: Loss rate for cnn.

Convolutional Neural Networks-Hyperparameters tuning

- Dropout rate
- Epoch
- Batch size
- Learning rate

Convolutional Neural Networks-ROC

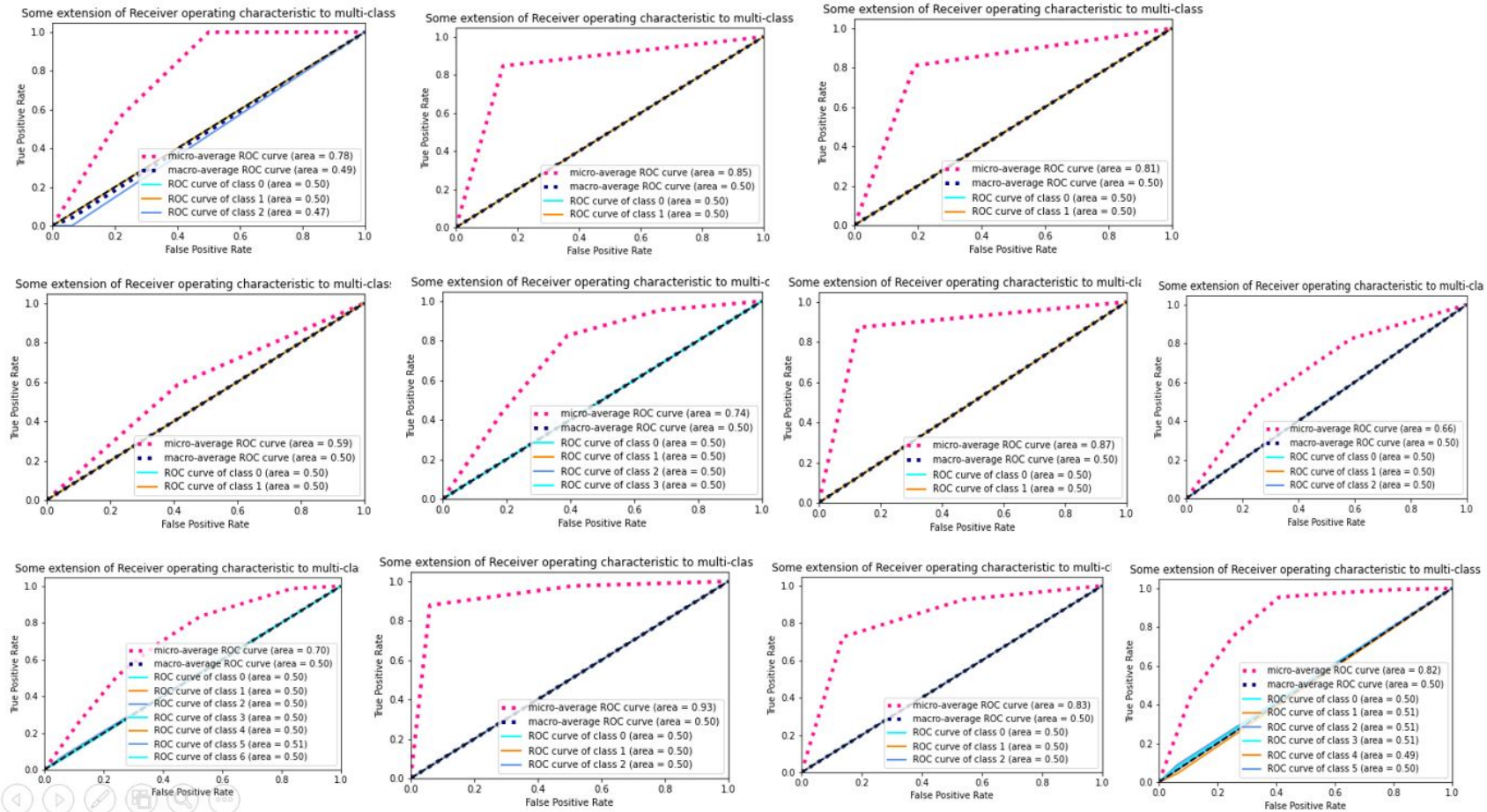


Figure: Roc curve for cnn.

Convolutional Neural Networks-Final Score

Column1	accuracy	epoch
class-1	0.5403	4
class-2	0.8618	2
class-3	0.8045	4
class-4	0.6017	5
class-5	0.4384	5
class-6	0.8797	3
class-7	0.5	5
class-8	0.28	4
class-9	0.8909	3
class-10	0.7265	3
class-11	0.4745	4

Figure: Final score for cnn.

Best classifier

- class1 - RF
- class2 - RF
- class3 - RF
- class4 - SVM
- class5 - SVM
- class6 - CNN
- class7 - RF
- class8 - RF
- class9 - CNN
- class10 - CNN
- class11 - SVM

References

[1] https://towardsdatascience.com/10-minutes-to-building-a-cnn-binary-image-classifier-in-tensorflow-4e216b2034aa
[2]https://scikit-learn.org/stable/