Student Name: Vladislav Dubrovenski, Student ID: 16281273

Final Project Report

Integrating Analysis of Dimension Reduction and GMM Model with Gaussian Support Vector Machine Classification

Vladislav Dubrovenski

Prepared for CS 5582 Computer Vision

GitHub Link:

https://github.com/vladi7/ComputerVision/tree/main/TermProject

Student Name: Vladislav Dubrovenski, Student ID: 16281273

Option A: Use pretrained VGG16 network final conv pooling features before fc_1, in this case 512 7x7 feature maps, and use Fisher Vector aggregation to do the classification :

1) [25pts] Compute the PCA and GMM models for the 49-dimension final conv features for kd=[16, 24] and number of components nc=[32, 64, 128] {Hint, refer to HW-2 for solutions} , show your code and results here.

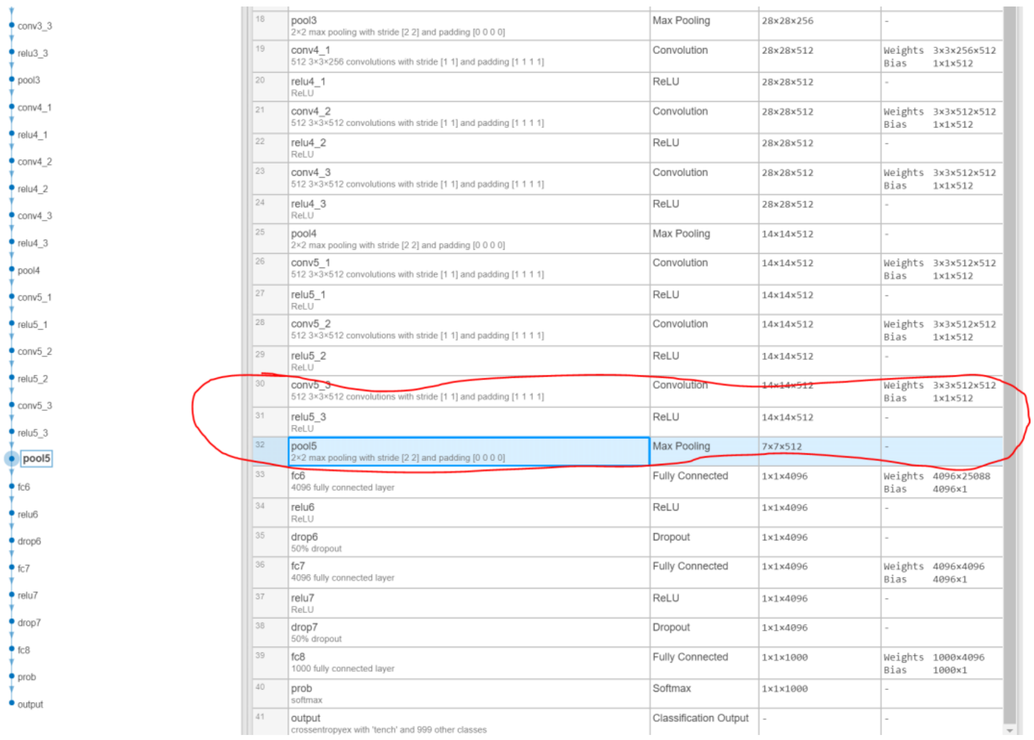
Analysis of the VGG16 Network

The following was used for the reference on how to get the gmm model in matlab: https://www.mathworks.com/help/stats/fitgmdist.html

Other link from documentation were also utilized, but the above one was the main inspiration on how to complete this term project.

First, let me describe the analysis of the vgg16 pretrained network. As the task described, I had to utilize 7x7x512 **pooling feature maps.** The vgg16 description giving in the task is overly simplified, lets take a look at the actual layers of the network that interest us via Matlab Deep Learning Network Analyzer:

Student Name: Vladislav Dubrovenski, Student ID: 16281273



18	pool3 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	28×28×256	-
19	conv4_1 512 3×3×256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×512	Weights 3×3×256×512 Bias 1×1×512
20	relu4_1 ReLU	ReLU	28×28×512	-
21	conv4_2 512 3×3×512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×512	Weights 3×3×512×512 Bias 1×1×512
22	relu4_2 ReLU	ReLU	28×28×512	-
23	conv4_3 512 3×3×512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	28×28×512	Weights 3×3×512×512 Bias 1×1×512
24	relu4_3 ReLU	ReLU	28×28×512	-
25	pool4 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	14×14×512	-
26	conv5_1 512 3×3×512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×512	Weights 3×3×512×512 Bias 1×1×512
27	relu5_1 ReLU	ReLU	14×14×512	-
28	conv5_2 512 3×3×512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×512	Weights 3×3×512×512 Bias 1×1×512
29	relu5_2 ReLU	ReLU	14×14×512	-
30	conv5_3 512 3×3×512 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	14×14×512	Weights 3×3×512×512 Bias 1×1×512
31	relu5_3 ReLU	ReLU	14×14×512	-
32	pool5 2×2 max pooling with stride [2 2] and padding [0 0 0 0]	Max Pooling	7×7×512	-
33	fc6 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×25088 Bias 4096×1
34	relu6 ReLU	ReLU	1×1×4096	-
35	drop6 50% dropout	Dropout	1×1×4096	-
36	fc7 4096 fully connected layer	Fully Connected	1×1×4096	Weights 4096×4096 Bias 4096×1
37	relu7 ReLU	ReLU	1×1×4096	-
38	drop7 50% dropout	Dropout	1×1×4096	-
39	fc8 1000 fully connected layer	Fully Connected	1×1×1000	Weights 1000×4096 Bias 1000×1
40	prob softmax	Softmax	1×1×1000	-
41	output crossentropyex with 'tench' and 999 other classes	Classification Output	-	-

I marked the three layers of our interests, conv5_3, relu5_3, and pool5. My first intention was to use conv5_3, but after careful analysis, I noticed that this layer has 14x14x512 features maps, which are not pooling with any stride or padding. I specifically do not account for the pool4 layer that happened before the 3 convolution layers, as this pooling is not applicable for the activations of conv5_3. The next layer, ReLU (relu5_3), cannot be used for the same reasons. The next layer (pool5), provides pooling with stride [2 2] and padding [0 0 0 0], as well as feature maps of 7x7x512, which is exactly what the task asked for. Therefore, I am using pool5 and extracting the features from it. The following Matlab commands were used to load the pretrained VGG16 model and analyze the network:

```
net = vgg16('Weights','imagenet');
analyzeNetwork(net);
```

Data Splitting and Feature Extraction

I first removed all the classes that were not required and left only 15 first classes described in the task.
The following command was used to load the images into the datastore:

Student Name: Vladislav Dubrovenski, Student ID: 16281273

```
imds =
imageDatastore('NWPU-RESISC45','IncludeSubfolders',tr
ue,'LabelSource','foldernames');
```

The above command is very useful since I could also use the folder names to extract labels from the image folders.

In order to split the as requested in the task, the following command can be utilized:

```
[training, validation, testing] = splitEachLabel(imds,0.714285,0.142857);
```

However, as I am going to utilize the Gaussian Classifier from Classification Learned App, I could set the validation size directly from that app. Therefore, I include the validation set in the training set that I further load in aforementioned app. The following is used for the actual data splitting:

```
[training, testing] = splitEachLabel(imds,0.714285);
```

I am able to use Deep Learning toolbox from the toolbox, and, therefore, I first augmented the image data store, and then extract the features with activations command:

```
augimdsTrain =
augmentedImageDatastore(inputSize(1:2),training);
augimdsTest =
augmentedImageDatastore(inputSize(1:2),testing);

layer = 'pool5';
featuresTestConv5_3 =
activations(net,augimdsTest,layer,'OutputAs','rows','
MiniBatchSize', 32);
featuresTrainConv5_3 =
activations(net,augimdsTrain,layer,'OutputAs','rows',
'MiniBatchSize', 32);
```

In the above command, I have to set MiniBatchSize in order to avoid out of memory error as I only have 6GB in my Discrete Video Card.

I also extract labels from training and testing sets using the following simple commands:

```
TrainLabels = training.Labels;
TestLabels = testing.Labels;
```

PCA with GMM Models

First, I compute the scores with PCA for 16 dimensions as well as 24 dimensions:

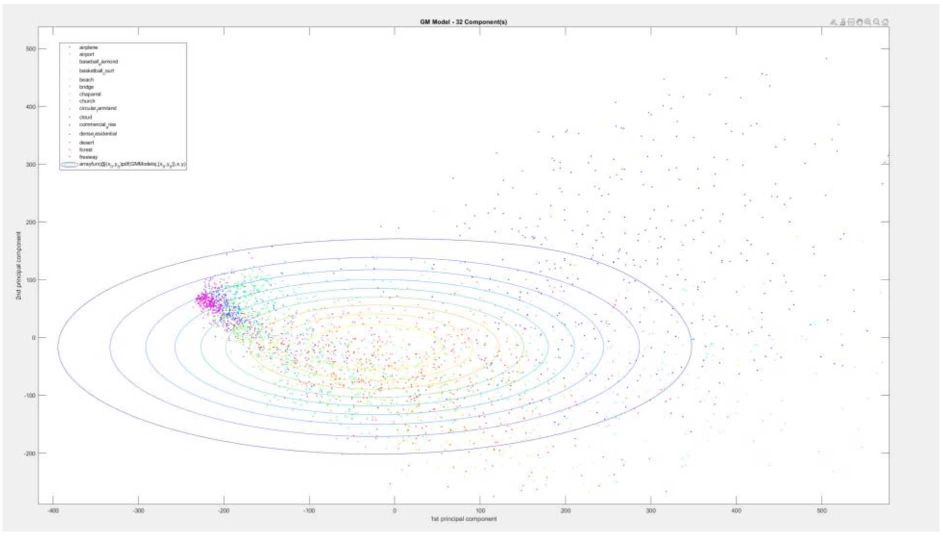
```
[~,score] = pca(featuresTrainConv5_3,'NumComponents',16);
[~,score] = pca(featuresTrainConv5_3,'NumComponents',24);
```

Student Name: Vladislav Dubrovenski, Student ID: 16281273

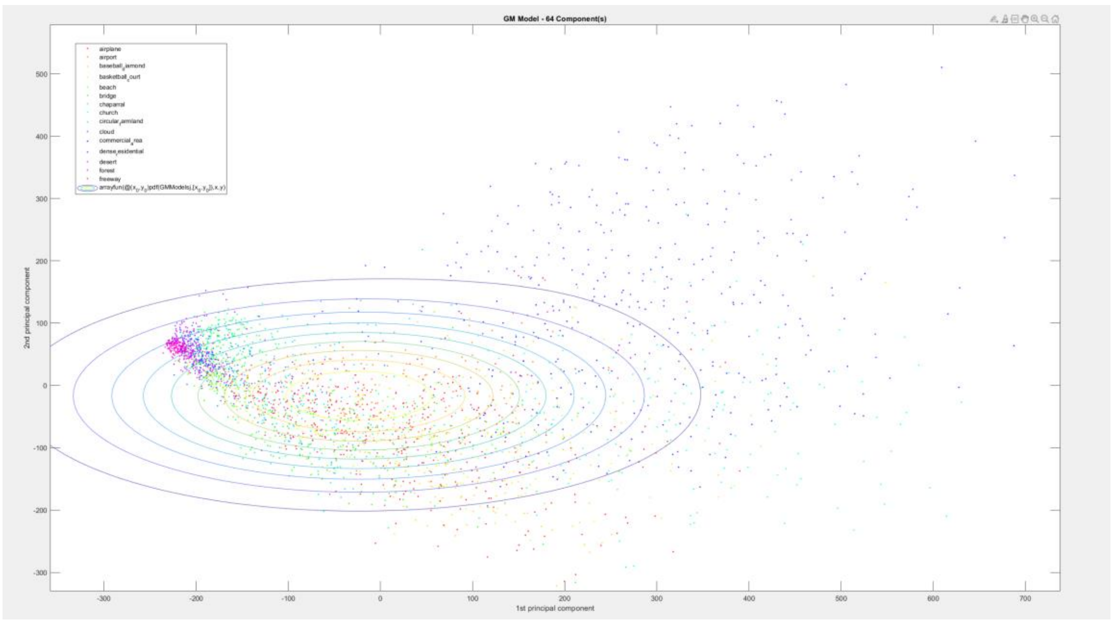
Please note that I do everything in a loop with the number of clusters, and therefore, store the scores and GMMModels in the array.
The following is used for computing the GMModels:

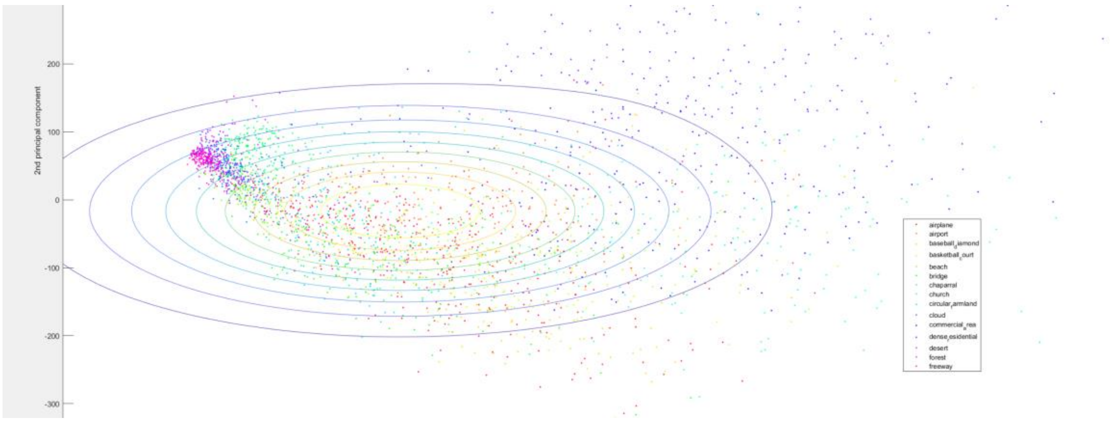
```
GMModels{j} =
fitgmdist(score,nc(j),'Options',options,'SharedCov',true,'CovType','diagonal', 'ProbabilityTolerance',0.0000003)
```

The complete code for the loop is going to be included in the next question, as it also contains the code for finding the optimal combination of number of dimensions and number of clusters. However, for the demonstration purposes, I include the plots of probability density functions(pdf) for number of clusters described in the task and number of dimensions reduced to 2 (only for plotting purposes):



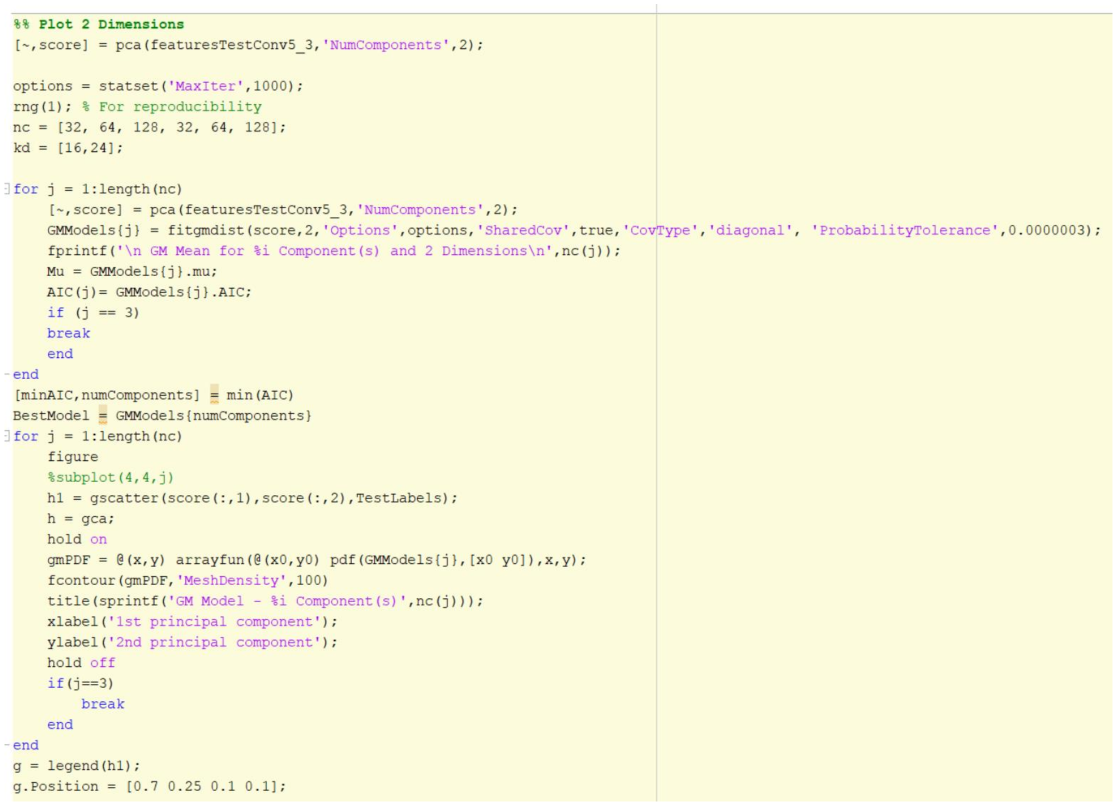
Student Name: Vladislav Dubrovenski, Student ID: 16281273





As we can see, the clustering looks similar, since I preserve only two dimensions in each case. I argue, however, that with the number of dimensions larger than 2, the GMM will be proved an efficient solution for the model fitting of this data. The following commands were used to compute the pdfs for each graph:

Student Name: Vladislav Dubrovenski, Student ID: 16281273



```matlab
%% Plot 2 Dimensions
[~,score] = pca(featuresTestConv5_3,'NumComponents',2);

options = statset('MaxIter',1000);
rng(1); % For reproducibility
nc = [32, 64, 128, 32, 64, 128];
kd = [16,24];

for j = 1:length(nc)
    [~,score] = pca(featuresTestConv5_3,'NumComponents',2);
    GMModels{j} = fitgmdist(score,2,'Options',options,'SharedCov',true,'CovType','diagonal', 'ProbabilityTolerance',0.0000003);
    fprintf('\n GM Mean for %i Component(s) and 2 Dimensions\n',nc(j));
    Mu = GMModels{j}.mu;
    AIC(j)= GMModels{j}.AIC;
    if (j == 3)
    break
    end
end
[minAIC,numComponents] = min(AIC)
BestModel = GMModels{numComponents}
for j = 1:length(nc)
    figure
    %subplot(4,4,j)
    h1 = gscatter(score(:,1),score(:,2),TestLabels);
    h = gca;
    hold on
    gmPDF = @(x,y) arrayfun(@(x0,y0) pdf(GMModels{j},[x0 y0]),x,y);
    fcontour(gmPDF,'MeshDensity',100)
    title(sprintf('GM Model - %i Component(s)',nc(j)));
    xlabel('1st principal component');
    ylabel('2nd principal component');
    hold off
    if(j==3)
        break
    end
end
g = legend(h1);
g.Position = [0.7 0.25 0.1 0.1];
```

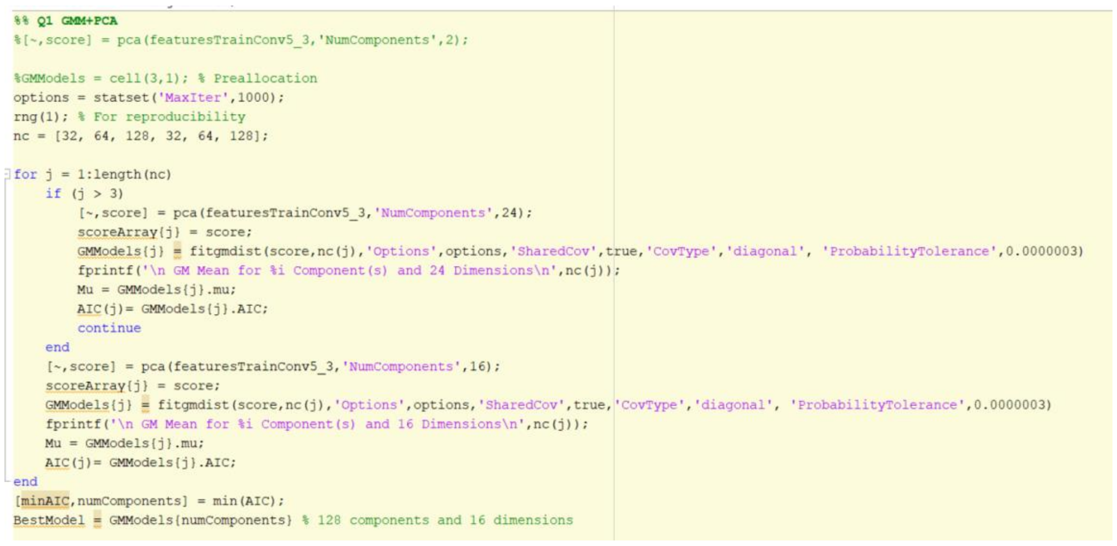
2) [75pts] For each of the 15 categories, determine the optimal subset of dimensions, and components to use in 1-vs-rest classification, e.g, for an kd=16, nc=64 FV, we can choose to turn off certain components in GMM to maximize the classification accuracy, similar to the SCFV case covered in class (Lec 9). The basic idea is to examine the class specific component prior, and turn off ones that are too small. Plot your results.

Akaike's Information Criterion(AIC)

The Matlab implementation of GMM model usefully computes AIC by utilization of prior probabilities. AIC further analyzes the amount of information that can be lost by the model, and that what this criterion actually represents, so, the lower the value the better the model.

The following is the complete code for creating six models (combination of number of dimensions and number of clusters described in the task), as well as finding the best model, which occurred to be 128 components with 16 dimensions (which is actually correct since we have only 15 classes, and there is a direct correlation, and 24 is not suitable for our data).

Student Name: Vladislav Dubrovenski, Student ID: 16281273



```
%% Q1 GMM+PCA
%[~,score] = pca(featuresTrainConv5_3,'NumComponents',2);

%GMModels = cell(3,1); % Preallocation
options = statset('MaxIter',1000);
rng(1); % For reproducibility
nc = [32, 64, 128, 32, 64, 128];

for j = 1:length(nc)
    if (j > 3)
        [~,score] = pca(featuresTrainConv5_3,'NumComponents',24);
        scoreArray{j} = score;
        GMModels{j} = fitgmdist(score,nc(j),'Options',options,'SharedCov',true,'CovType','diagonal', 'ProbabilityTolerance',0.0000003)
        fprintf('\n GM Mean for %i Component(s) and 24 Dimensions\n',nc(j));
        Mu = GMModels{j}.mu;
        AIC(j)= GMModels{j}.AIC;
        continue
    end
    [~,score] = pca(featuresTrainConv5_3,'NumComponents',16);
    scoreArray{j} = score;
    GMModels{j} = fitgmdist(score,nc(j),'Options',options,'SharedCov',true,'CovType','diagonal', 'ProbabilityTolerance',0.0000003)
    fprintf('\n GM Mean for %i Component(s) and 16 Dimensions\n',nc(j));
    Mu = GMModels{j}.mu;
    AIC(j)= GMModels{j}.AIC;
end
[minAIC,numComponents] = min(AIC);
BestModel = GMModels{numComponents} % 128 components and 16 dimensions
```

Let me describe the most important code lines below:

The following is used to compute the scores from pca and store them in the score vector. As it can be seen from the above screenshot of the whole code, I did similar thing for 16 dimensions as well.

```
[~,score] =
pca(featuresTrainConv5_3,'NumComponents',24);
scoreArray{j} = score;
```

After that, I compute GMM model with the following command:

```
GMModels{j} =
fitgmdist(score,nc(j),'Options',options,'SharedCov',true,
'CovType','diagonal', 'ProbabilityTolerance',0.0000003)
```

Please note the ProbabilityTolerance option that I pass to the fitter of the GMM model. I found out that 0.0000003 Prior Probability tolerance is the optimal for every class and that is how I turn of the prior probabilities that are too small to consider.
Another important piece of the code is AIC extraction and storage in AIC vector:

```
AIC(j)= GMModels{j}.AIC;
```

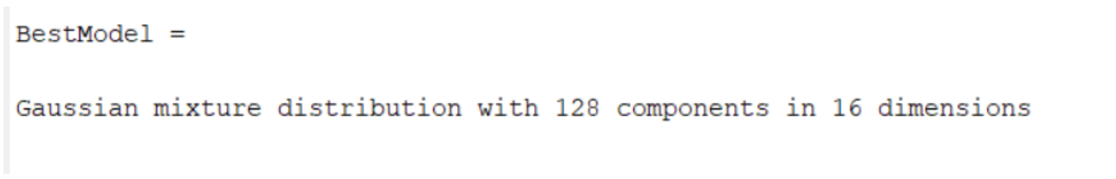
After the execution of the loop is done, I find the model where AIC value is minimal:

```
[minAIC,numComponents] = min(AIC);
```

After that, I extract the best model with the following command and print it to the output:

```
BestModel = GMModels{numComponents}
```

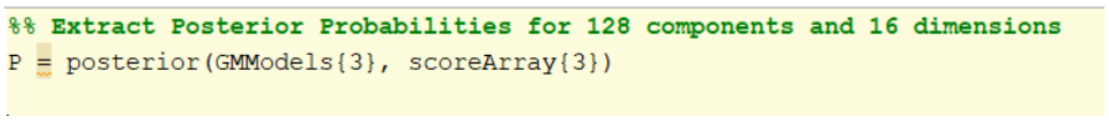
Student Name: Vladislav Dubrovenski, Student ID: 16281273



```
BestModel =

Gaussian mixture distribution with 128 components in 16 dimensions
```

Furthermore, I extract the posterior probabilities for this model and pca scores that are used in the Gaussian SVM classifier:



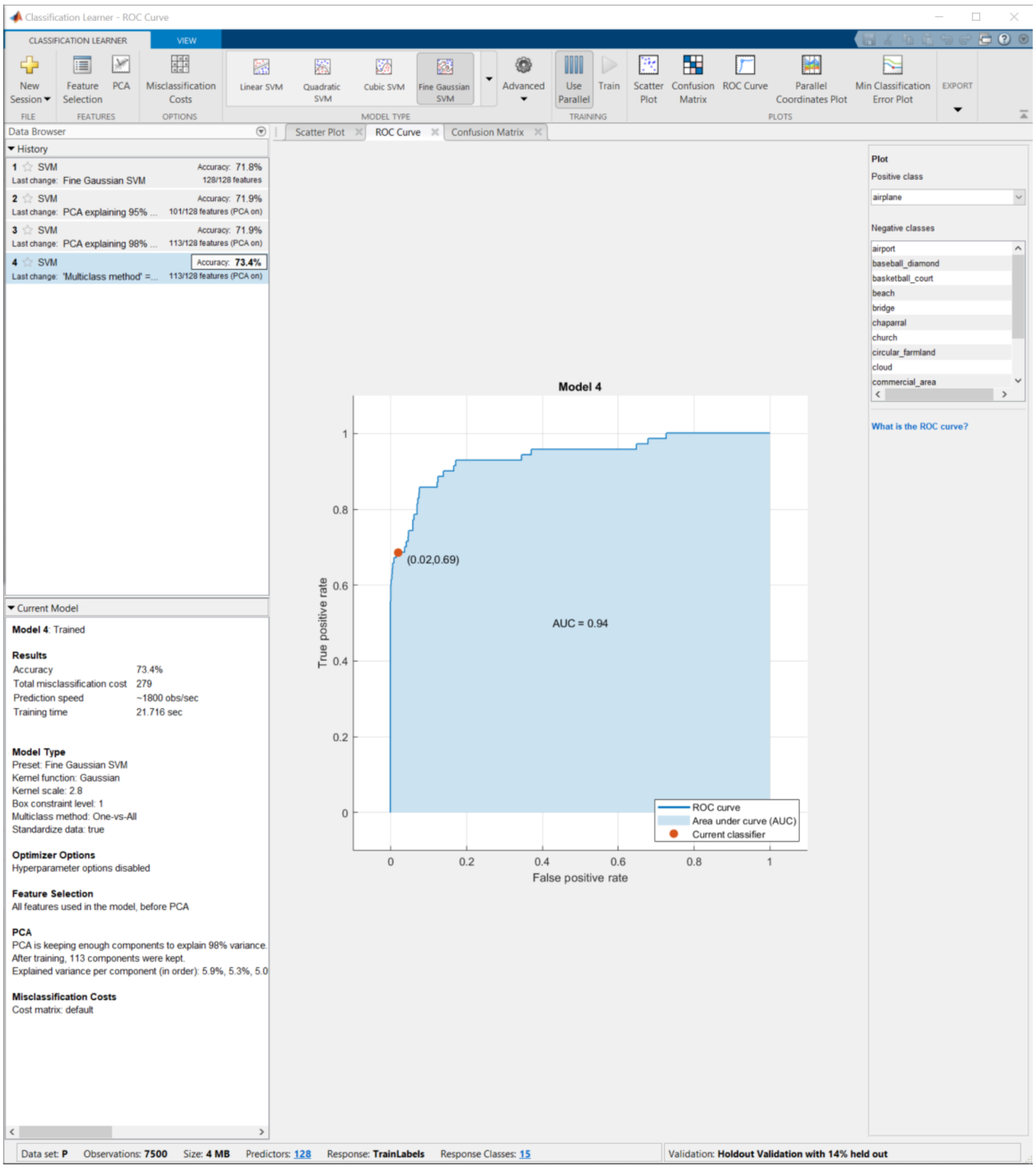
```
%% Extract Posterior Probabilities for 128 components and 16 dimensions
P = posterior(GMModels{3}, scoreArray{3})
```

Let me note that I was not able to compute the baseline model performance (of the pooling layer) without the dimension reduction due to the limitations of my software. It was not required in the task, however, I did note that when I attempted to compute such baseline model, it took 2 hours to attempt to train it and it was not successful. The training time for the model with Dimension/Feature reductions is just 21.7 seconds, while the prediction speed is around 1800 observations/second (7500 observations total of course).

To further improve the classification, I use the PCA again for Gaussian SVM 1 vs rest to further filter the components. I only used components that explain 98% of the variance. I achieve **73.4%** of accuracy with the above optimizations. I argue that is an excellent result as the training time and inference time is extremely low. This technique can be useful for the computers with the limited computational power (e.g. cell phones).

The below is ROC curve for the best model as well as other important info about the model:

Student Name: Vladislav Dubrovenski, Student ID: 16281273



History

1 SVM Accuracy: 71.8%
Last change: Fine Gaussian SVM 128/128 features

2 SVM Accuracy: 71.9%
Last change: PCA explaining 95% ... 101/128 features (PCA on)

3 SVM Accuracy: 71.9%
Last change: PCA explaining 98% ... 113/128 features (PCA on)

4 SVM Accuracy: **73.4%**
Last change: 'Multiclass method' =... 113/128 features (PCA on)

Current Model

Model 4: Trained

Results

Accuracy	73.4%
Total misclassification cost	279
Prediction speed	~1800 obs/sec
Training time	21.716 sec

Model Type
Preset: Fine Gaussian SVM
Kernel function: Gaussian
Kernel scale: 2.8
Box constraint level: 1
Multiclass method: One-vs-All
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

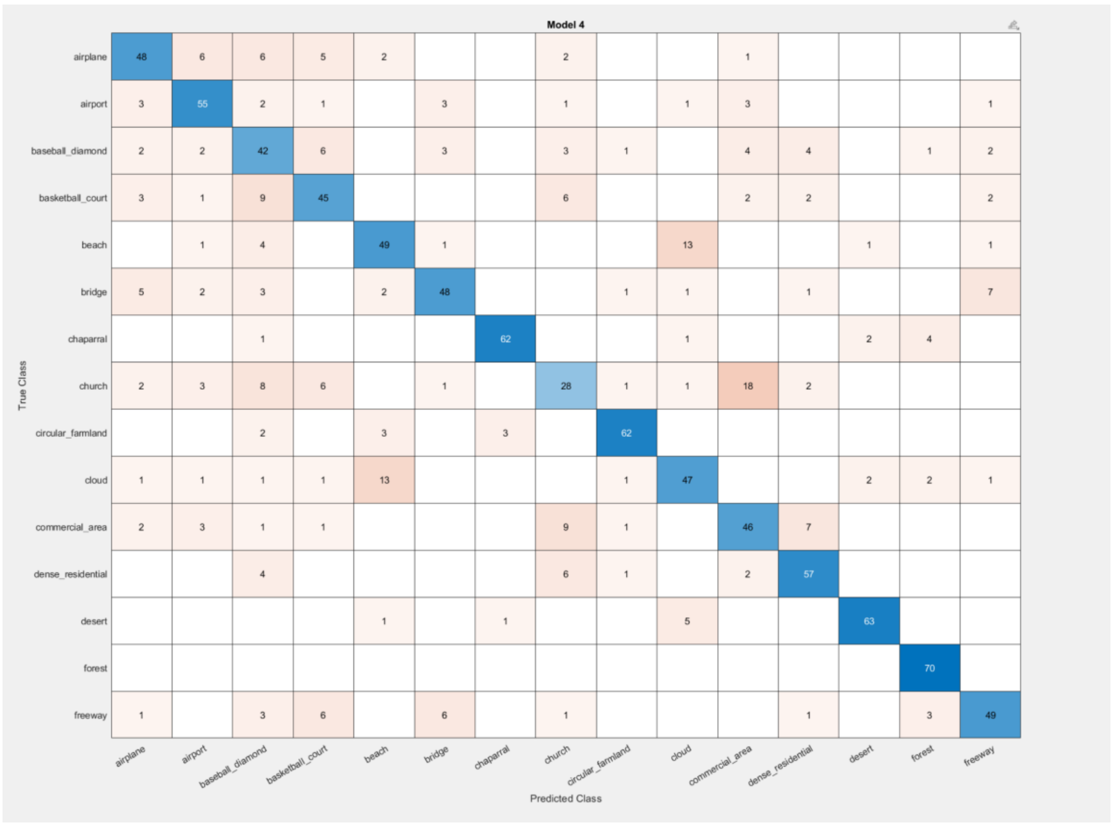
PCA
PCA is keeping enough components to explain 98% variance.
After training, 113 components were kept.
Explained variance per component (in order): 5.9%, 5.3%, 5.0

Misclassification Costs
Cost matrix: default

Data set: P Observations: **7500** Size: **4 MB** Predictors: 128 Response: **TrainLabels** Response Classes: 15 Validation: **Holdout Validation with 14% held out**

The confusion matrix is as follows:

Student Name: Vladislav Dubrovenski, Student ID: 16281273



	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	48	6	6	5	2			2			1				
airport	3	55	2	1		3		1		1	3				1
baseball_diamond	2	2	42	6		3		3	1		4	4		1	2
basketball_court	3	1	9	45				6			2	2			2
beach		1	4		49	1				13			1		1
bridge	5	2	3		2	48			1	1		1			7
chaparral			1				62			1			2	4	
church	2	3	8	6		1		28	1	1	18	2			
circular_farmland			2		3		3		62						
cloud	1	1	1	1	13				1	47			2	2	1
commercial_area	2	3	1	1				9	1		46	7			
dense_residential			4					6	1		2	57			
desert					1		1			5			63		
forest														70	
freeway	1		3	6		6		1				1		3	49

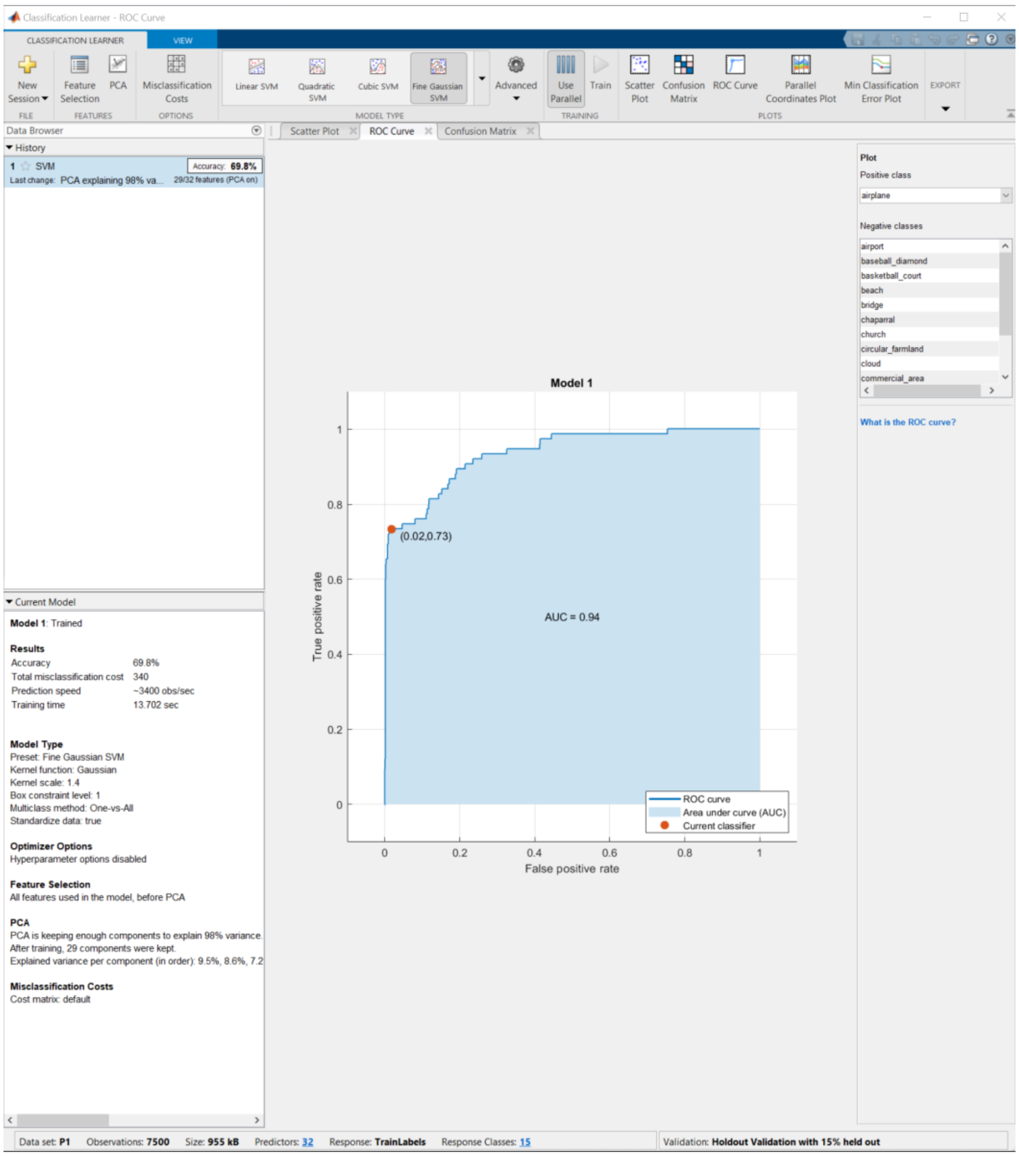
As it can be seen from the above confusion matrix, the worst predicted classes were church, cloud, and beach. It is understandable, as the features of beach and clouds are very similar (misclassification of both therefore), and church with commercial area have similar features as well. The turning of the specific priors did not yield good results as the other classes were misclassified after. Therefore, I further argue that my model is the best model possible for the model described in the tasks.

The following screenshots describe the statistics information for all other cases with the exact same parameters:

KD = 16 NC= 32(Accuracy = 69.8%):

ROC Curve and all other information:

Student Name: Vladislav Dubrovenski, Student ID: 16281273



Current Model

Model 1: Trained

Results

Accuracy	69.8%
Total misclassification cost	340
Prediction speed	~3400 obs/sec
Training time	13.702 sec

Model Type
Preset: Fine Gaussian SVM
Kernel function: Gaussian
Kernel scale: 1.4
Box constraint level: 1
Multiclass method: One-vs-All
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

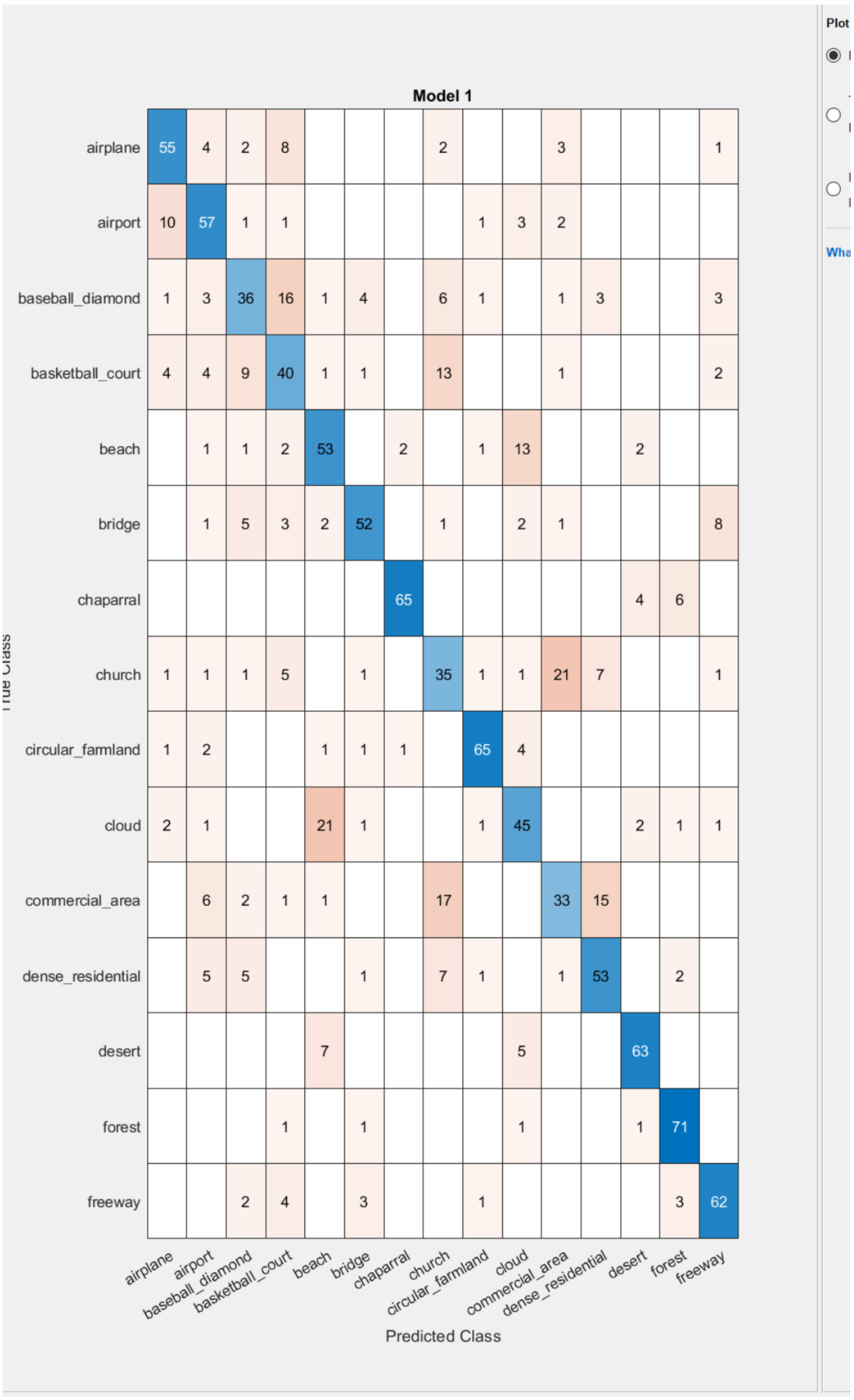
PCA
PCA is keeping enough components to explain 98% variance.
After training, 29 components were kept.
Explained variance per component (in order): 9.5%, 8.6%, 7.2

Misclassification Costs
Cost matrix: default

Data set: **P1** Observations: **7500** Size: **955 kB** Predictors: **32** Response: **TrainLabels** Response Classes: **15** Validation: **Holdout Validation with 15% held out**

Confusion Matrix:

Student Name: Vladislav Dubrovenski, Student ID: 16281273

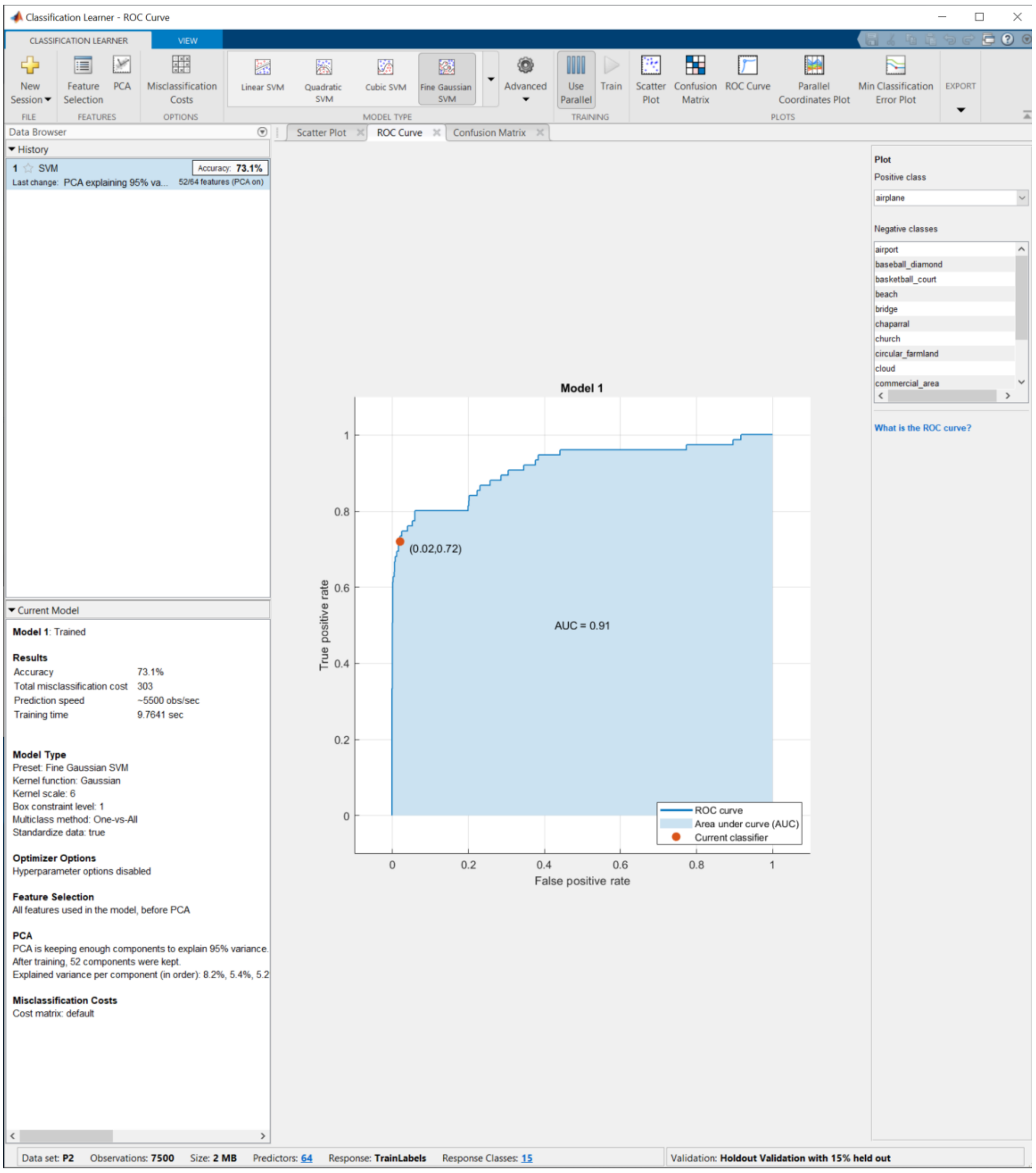


	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	55	4	2	8				2			3				1
airport	10	57	1	1					1	3	2				
baseball_diamond	1	3	36	16	1	4		6	1		1	3			3
basketball_court	4	4	9	40	1	1		13			1				2
beach		1	1	2	53		2		1	13			2		
bridge		1	5	3	2	52		1		2	1				8
chaparral							65						4	6	
church	1	1	1	5		1		35	1	1	21	7			1
circular_farmland	1	2			1	1	1		65	4					
cloud	2	1			21	1			1	45			2	1	1
commercial_area		6	2	1	1			17			33	15			
dense_residential		5	5			1		7	1		1	53		2	
desert					7					5			63		
forest				1		1				1			1	71	
freeway			2	4		3			1					3	62

Student Name: Vladislav Dubrovenski, Student ID: 16281273

KD = 16 NC= 64(Accuracy = 73.1%):

ROC Curve and all other information:



1 SVM Accuracy: 73.1%
Last change: PCA explaining 95% va... 52/64 features (PCA on)

Current Model

Model 1: Trained

Results

Accuracy	73.1%
Total misclassification cost	303
Prediction speed	~5500 obs/sec
Training time	9.7641 sec

Model Type
Preset: Fine Gaussian SVM
Kernel function: Gaussian
Kernel scale: 6
Box constraint level: 1
Multiclass method: One-vs-All
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

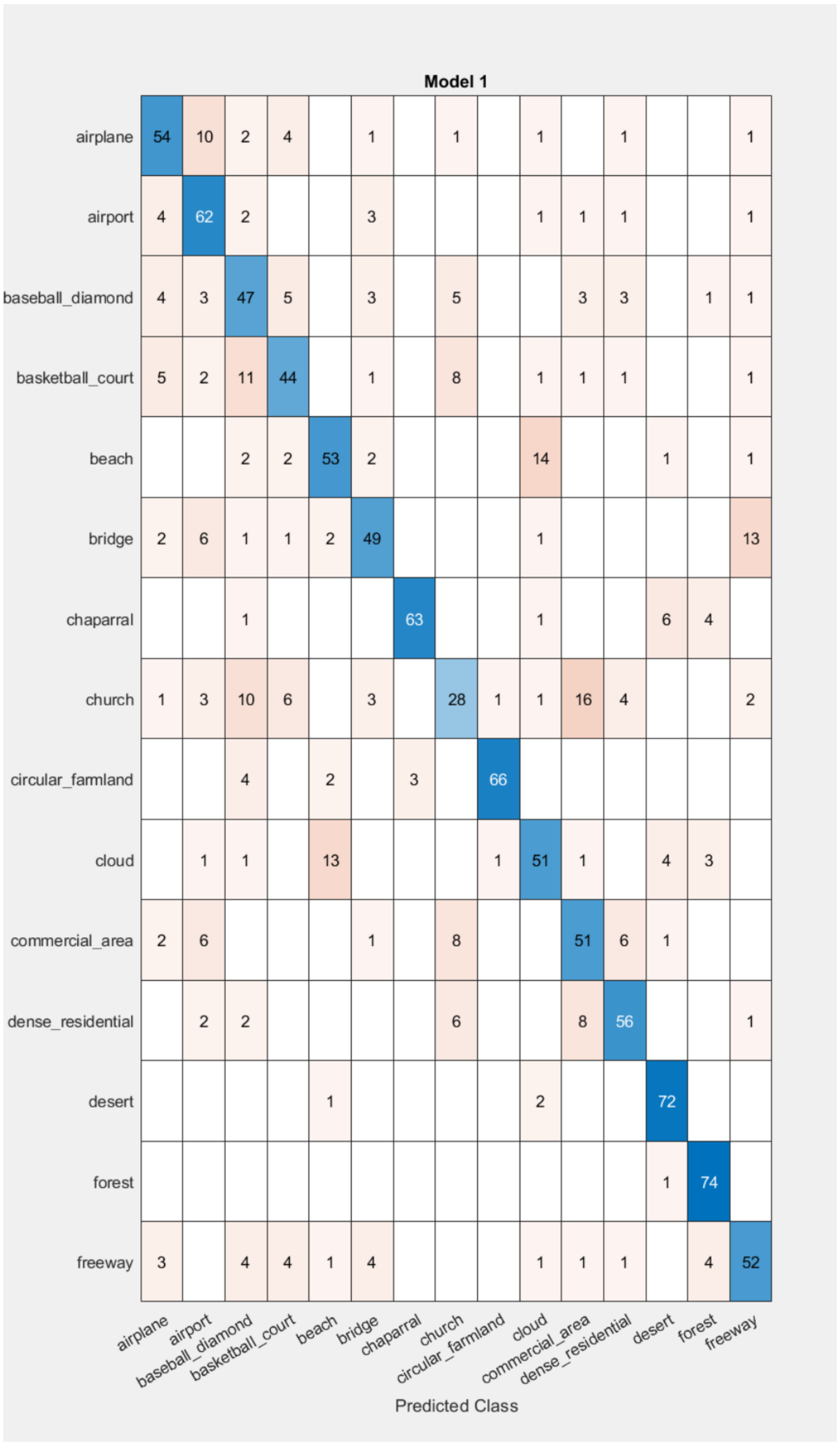
PCA
PCA is keeping enough components to explain 95% variance.
After training, 52 components were kept.
Explained variance per component (in order): 8.2%, 5.4%, 5.2

Misclassification Costs
Cost matrix: default

Data set: **P2** Observations: **7500** Size: **2 MB** Predictors: **64** Response: **TrainLabels** Response Classes: **15** Validation: **Holdout Validation with 15% held out**

Confusion Matrix:

Student Name: Vladislav Dubrovenski, Student ID: 16281273

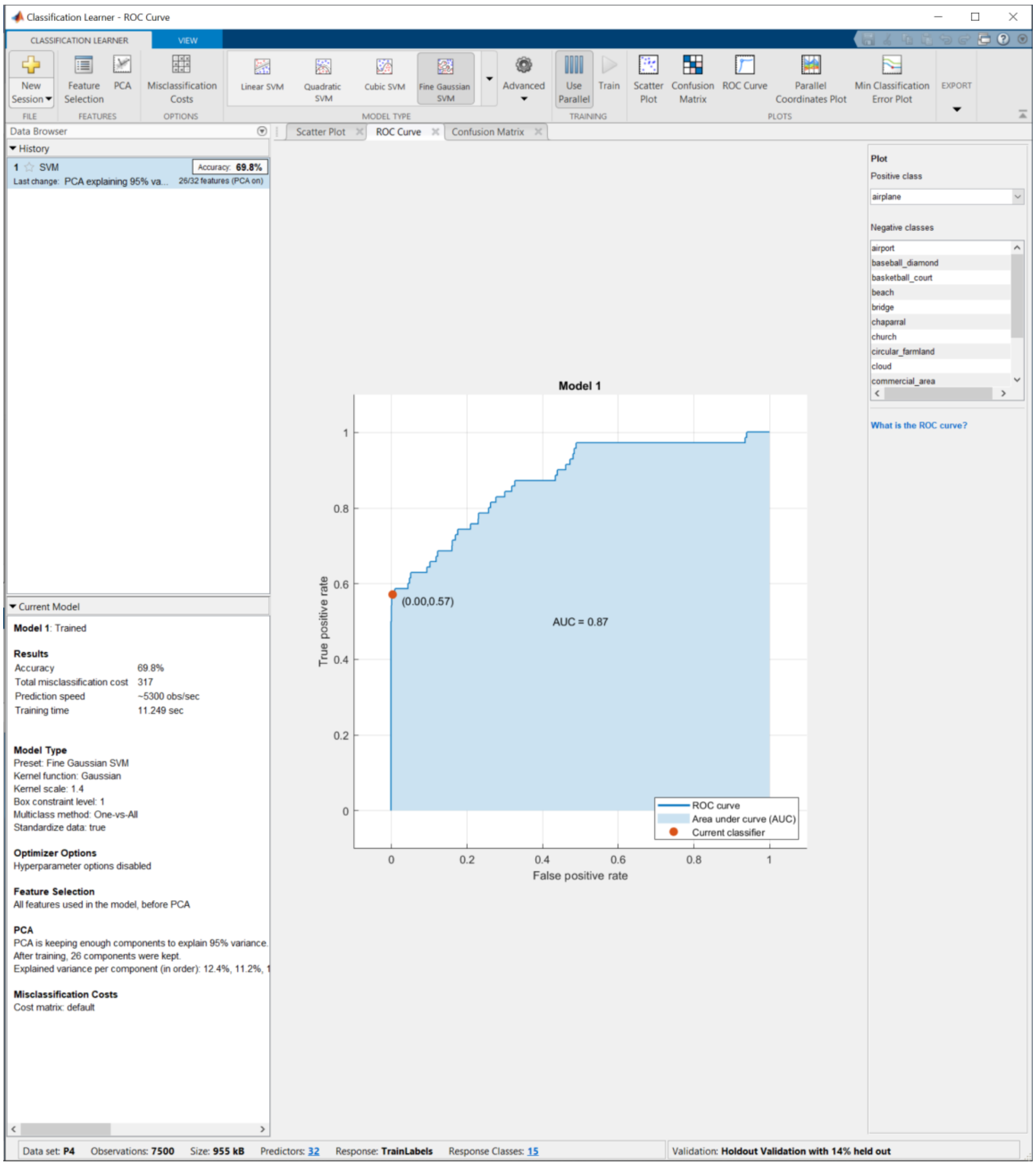


	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	54	10	2	4		1		1		1		1			1
airport	4	62	2			3				1	1	1			1
baseball_diamond	4	3	47	5		3		5			3	3		1	1
basketball_court	5	2	11	44		1		8		1	1	1			1
beach			2	2	53	2				14			1		1
bridge	2	6	1	1	2	49				1					13
chaparral			1				63			1			6	4	
church	1	3	10	6		3		28	1	1	16	4			2
circular_farmland			4		2		3		66						
cloud		1	1		13				1	51	1		4	3	
commercial_area	2	6				1		8			51	6	1		
dense_residential		2	2					6			8	56			1
desert					1					2			72		
forest													1	74	
freeway	3		4	4	1	4				1	1	1		4	52

KD = 24 NC= 32(Accuracy = 69.8%):

ROC Curve and all other information:

Student Name: Vladislav Dubrovenski, Student ID: 16281273



1 SVM Accuracy: 69.8%
Last change: PCA explaining 95% va... 26/32 features (PCA on)

Current Model

Model 1: Trained

Results

Accuracy	69.8%
Total misclassification cost	317
Prediction speed	~5300 obs/sec
Training time	11.249 sec

Model Type
Preset: Fine Gaussian SVM
Kernel function: Gaussian
Kernel scale: 1.4
Box constraint level: 1
Multiclass method: One-vs-All
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

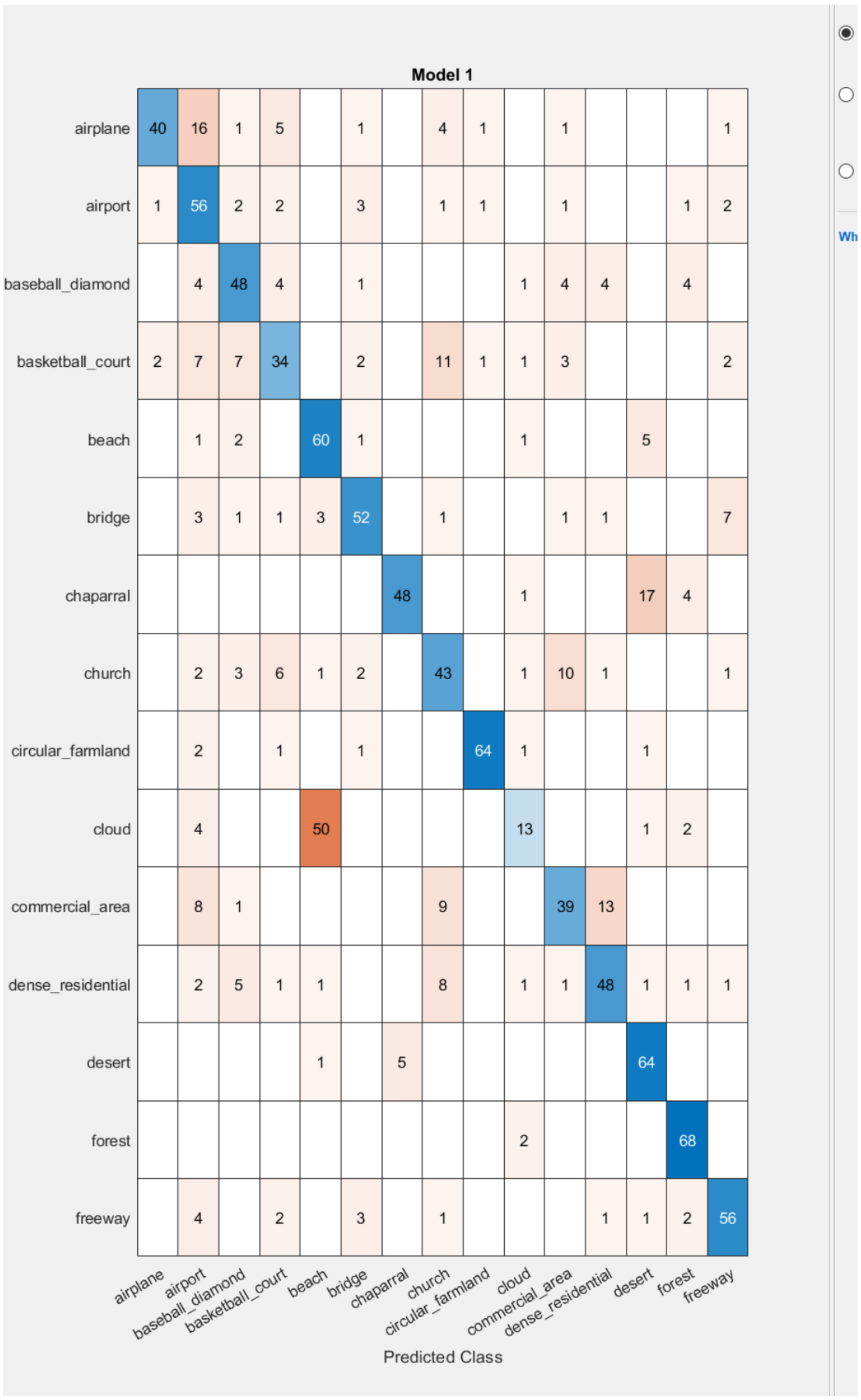
PCA
PCA is keeping enough components to explain 95% variance.
After training, 26 components were kept.
Explained variance per component (in order): 12.4%, 11.2%,

Misclassification Costs
Cost matrix: default

Data set: **P4** Observations: **7500** Size: **955 kB** Predictors: **32** Response: **TrainLabels** Response Classes: **15** Validation: **Holdout Validation with 14% held out**

Confusion Matrix:

Student Name: Vladislav Dubrovenski, Student ID: 16281273

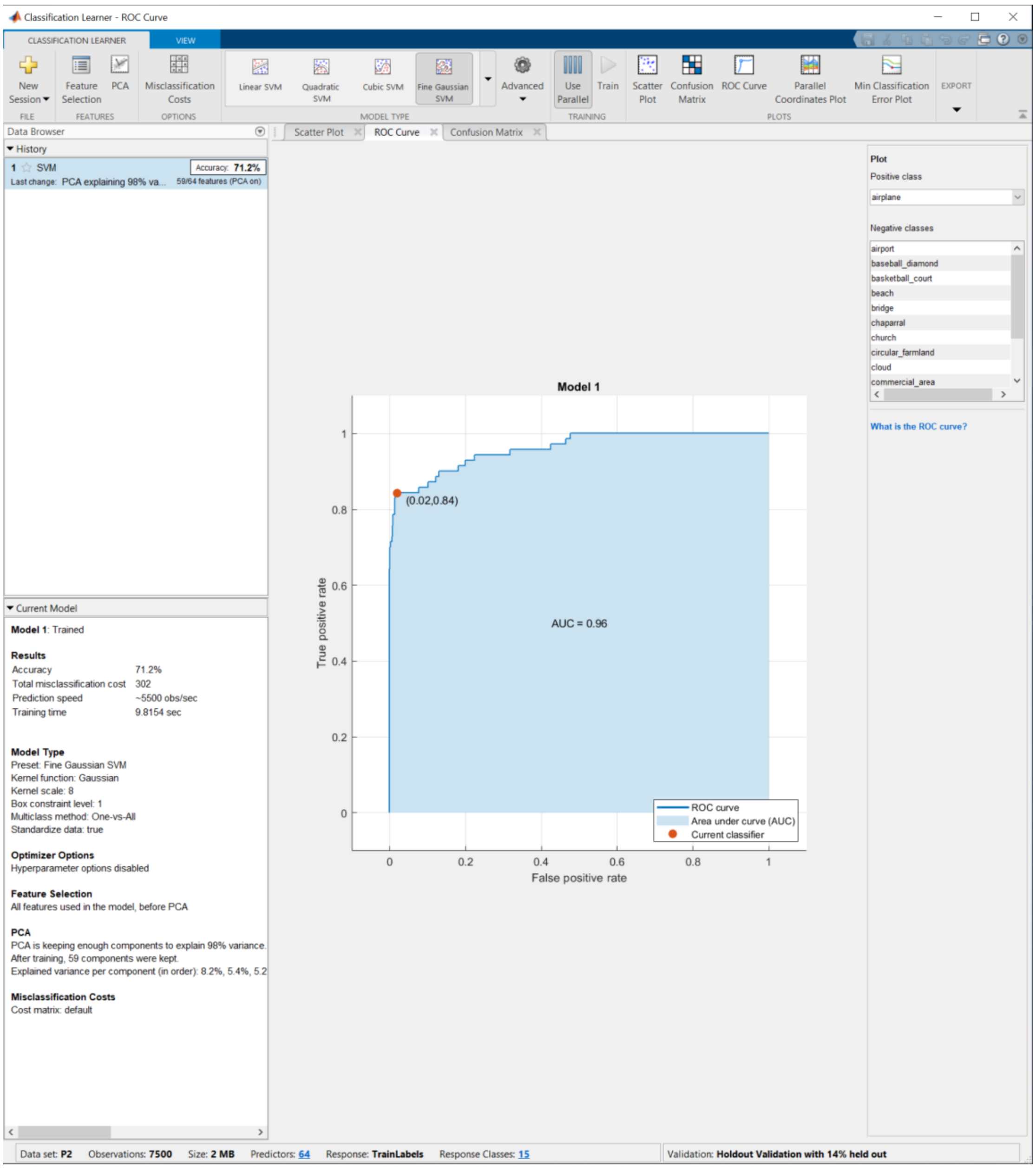


	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	40	16	1	5		1		4	1		1				1
airport	1	56	2	2		3		1	1		1			1	2
baseball_diamond		4	48	4		1				1	4	4		4	
basketball_court	2	7	7	34		2		11	1	1	3				2
beach		1	2		60	1				1			5		
bridge		3	1	1	3	52		1			1	1			7
chaparral							48			1			17	4	
church		2	3	6	1	2		43		1	10	1			1
circular_farmland		2		1		1			64	1			1		
cloud		4			50					13			1	2	
commercial_area		8	1					9			39	13			
dense_residential		2	5	1	1			8		1	1	48	1	1	1
desert					1		5						64		
forest										2				68	
freeway		4		2		3		1				1	1	2	56

Student Name: Vladislav Dubrovenski, Student ID: 16281273

KD = 24 NC= 64(Accuracy = 71.2%):

ROC Curve and all other information:



History
1 SVM Accuracy: 71.2%
Last change: PCA explaining 98% va... 59/64 features (PCA on)

Current Model

Model 1: Trained

Results

Accuracy	71.2%
Total misclassification cost	302
Prediction speed	~5500 obs/sec
Training time	9.8154 sec

Model Type
Preset: Fine Gaussian SVM
Kernel function: Gaussian
Kernel scale: 8
Box constraint level: 1
Multiclass method: One-vs-All
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

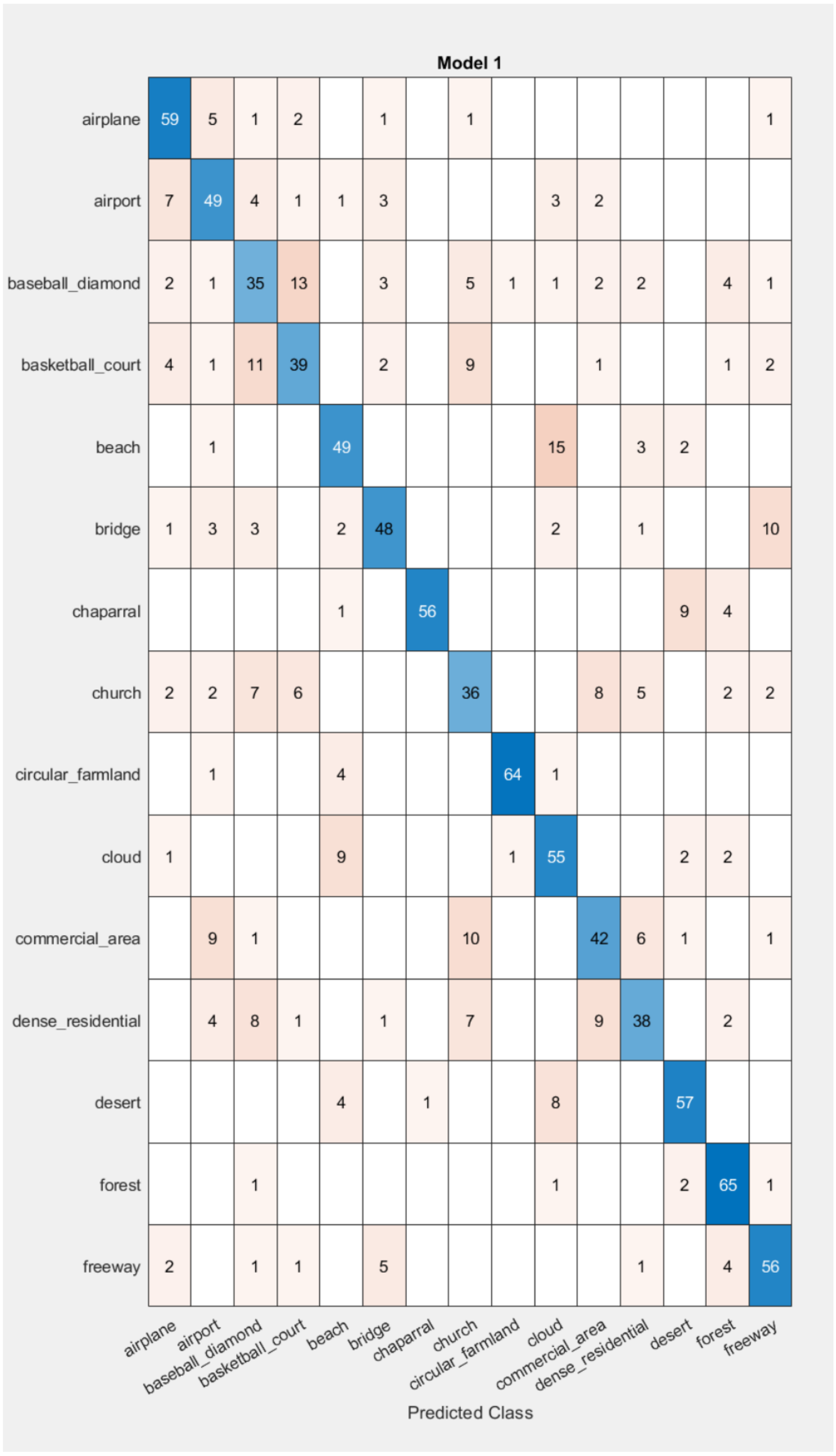
PCA
PCA is keeping enough components to explain 98% variance.
After training, 59 components were kept.
Explained variance per component (in order): 8.2%, 5.4%, 5.2

Misclassification Costs
Cost matrix: default

Data set: **P2** Observations: **7500** Size: **2 MB** Predictors: **64** Response: **TrainLabels** Response Classes: **15** Validation: **Holdout Validation with 14% held out**

Confusion Matrix:

Student Name: Vladislav Dubrovenski, Student ID: 16281273

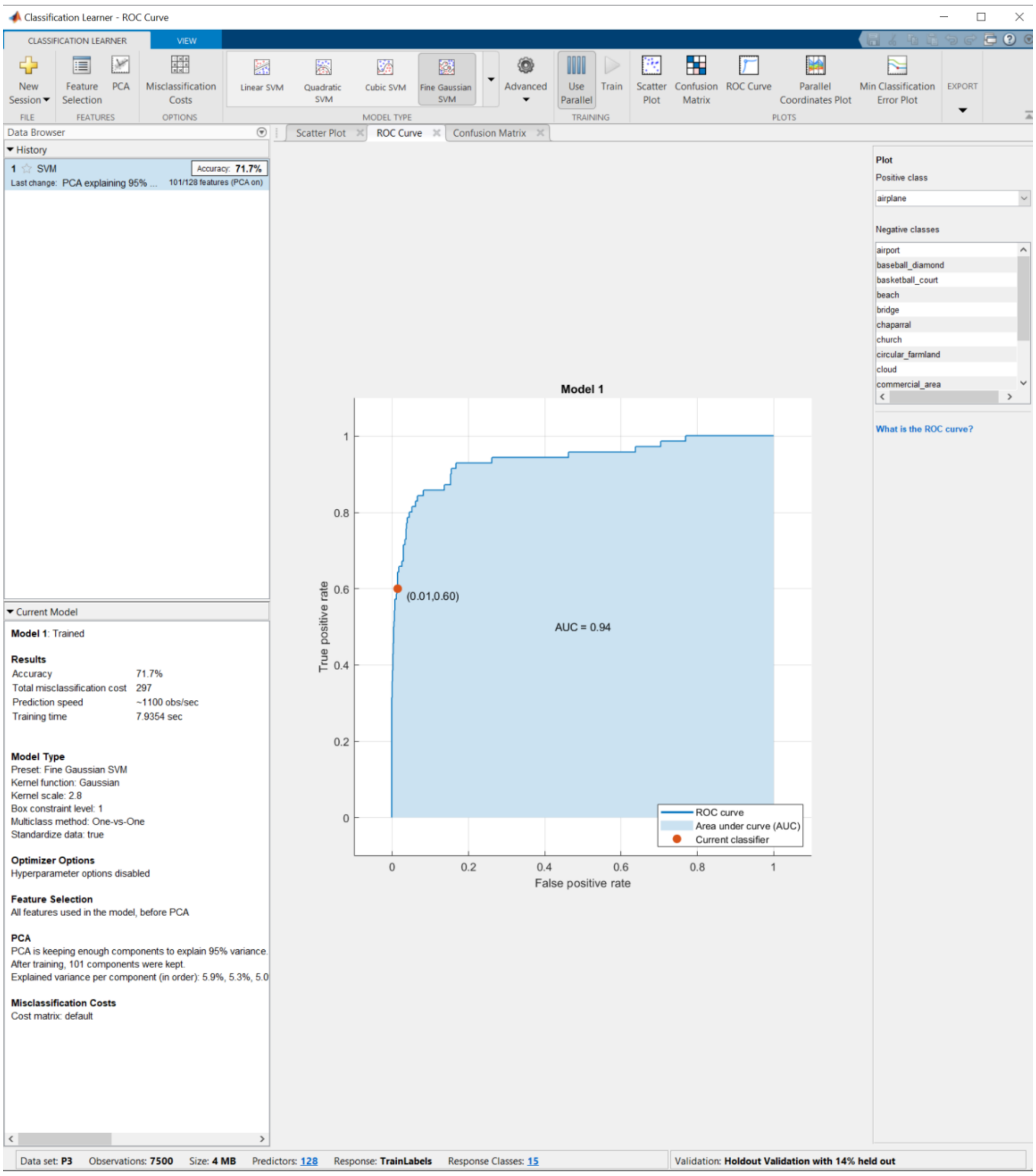


	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	59	5	1	2		1		1							1
airport	7	49	4	1	1	3				3	2				
baseball_diamond	2	1	35	13		3		5	1	1	2	2		4	1
basketball_court	4	1	11	39		2		9			1			1	2
beach		1			49					15		3	2		
bridge	1	3	3		2	48				2		1			10
chaparral					1		56						9	4	
church	2	2	7	6				36			8	5		2	2
circular_farmland		1			4				64	1					
cloud	1				9				1	55			2	2	
commercial_area		9	1					10			42	6	1		1
dense_residential		4	8	1		1		7			9	38		2	
desert					4		1			8			57		
forest			1							1			2	65	1
freeway	2		1	1		5						1		4	56

Student Name: Vladislav Dubrovenski, Student ID: 16281273

KD = 24 NC= 128(Accuracy = 71.7%):

ROC Curve and all other information:



1 SVM Accuracy: 71.7%
Last change: PCA explaining 95% ... 101/128 features (PCA on)

Current Model

Model 1: Trained

Results
Accuracy 71.7%
Total misclassification cost 297
Prediction speed ~1100 obs/sec
Training time 7.9354 sec

Model Type
Preset: Fine Gaussian SVM
Kernel function: Gaussian
Kernel scale: 2.8
Box constraint level: 1
Multiclass method: One-vs-One
Standardize data: true

Optimizer Options
Hyperparameter options disabled

Feature Selection
All features used in the model, before PCA

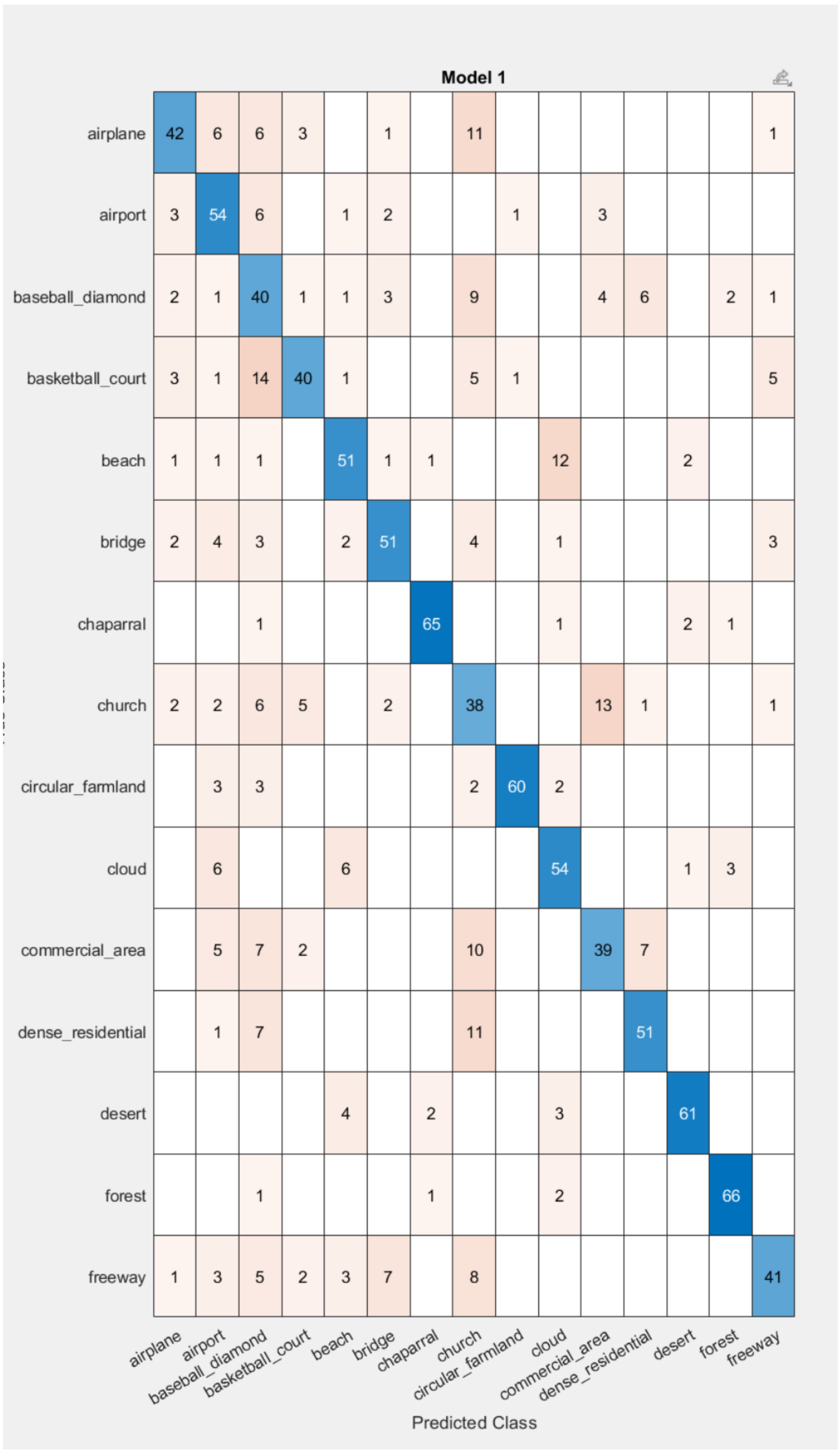
PCA
PCA is keeping enough components to explain 95% variance.
After training, 101 components were kept.
Explained variance per component (in order): 5.9%, 5.3%, 5.0

Misclassification Costs
Cost matrix: default

Data set: **P3** Observations: **7500** Size: **4 MB** Predictors: **128** Response: **TrainLabels** Response Classes: **15** Validation: **Holdout Validation with 14% held out**

Confusion Matrix:

Student Name: Vladislav Dubrovenski, Student ID: 16281273



Model 1

	airplane	airport	baseball_diamond	basketball_court	beach	bridge	chaparral	church	circular_farmland	cloud	commercial_area	dense_residential	desert	forest	freeway
airplane	42	6	6	3		1		11							1
airport	3	54	6		1	2			1		3				
baseball_diamond	2	1	40	1	1	3		9			4	6		2	1
basketball_court	3	1	14	40	1			5	1						5
beach	1	1	1		51	1	1			12			2		
bridge	2	4	3		2	51		4		1					3
chaparral			1				65			1			2	1	
church	2	2	6	5		2		38			13	1			1
circular_farmland		3	3					2	60	2					
cloud		6			6					54			1	3	
commercial_area		5	7	2				10			39	7			
dense_residential		1	7					11				51			
desert					4		2			3			61		
forest			1				1			2				66	
freeway	1	3	5	2	3	7		8							41

Predicted Class

Student Name: Vladislav Dubrovenski, Student ID: 16281273

The following running instructions were slightly modified from HW4 as the set up did not change and every feature of Matlab that was used in the HW4 were used in this term project:

Requirements: Matlab 2020b, Deep Learning license, Statistics license, and discrete video card is preferable.

1. Run the splitter.
2. Import vgg16 pretrained. Most likely, you don't have that model downloaded, so just use the link from the error message. After that, just click on it and follow downloading instruction. You might need an account, it doesn't have to be attached to a license (gmail worked for me). You may see the architecture to make sure it is the correct model.
3. Extract the features (need some toolbox license, not sure which one, most likely DL again).
4. Use Classification learner with the same parameters as on the screenshots above with P1, P2, P3, P4, P5, P6