Αναγνώριση Προτύπων

Εργασία 4

Ονοματεπώνυμο: Μωραΐτη Παναγιώτα

ΑΜ: 58054

Πίνακας περιεχομένων

Άσκηση 1

Το IRIS data set περιέχει μορφολογικές μετρήσεις για 150 φυτά iris (είδος κρίνου, αγριόκρινο). Τα δεδομένα είναι της μορφής: (μήκος σέπαλου, πλάτος σέπαλου, μήκος πετάλου, πλάτος πετάλου) σε cm. Από αυτά τα 150 φυτά, 50 είναι Iris Setosa (ω1), 50 είναι Iris Versicolour (ω2) και 50 είναι Iris Virginica (ω3).

Α. Να χωριστεί το dataset σε 80% training και 20% test δείγματα με τυχαίο τρόπο.

Αρχικά, τα δεδομένα πρέπει να είναι όλα σε αριθμητική μορφή (αριθμοί). Οπότε, με βάση τη στήλη με τα labels που περιέχει το όνομα κάθε κλάσης (στήλη class), δημιουργούμε τη στήλη class_encoded που περιέχει την κλάση σε αριθμητική μορφή (0, 1, 2).

	sepal_length	sepal_width	petal_length	petal_width	class	class_encoded
0	5.1	3.5	1.4	0.2	Iris-setosa	0
1	4.9	3.0	1.4	0.2	Iris-setosa	0
2	4.7	3.2	1.3	0.2	Iris-setosa	0
3	4.6	3.1	1.5	0.2	Iris-setosa	0
4	5.0	3.6	1.4	0.2	Iris-setosa	0
...	...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica	2
146	6.3	2.5	5.0	1.9	Iris-virginica	2
147	6.5	3.0	5.2	2.0	Iris-virginica	2
148	6.2	3.4	5.4	2.3	Iris-virginica	2
149	5.9	3.0	5.1	1.8	Iris-virginica	2

150 rows × 6 columns

Χρησιμοποιώντας τη συνάρτηση train_test_split της sklearn χωρίζουμε το dataset στα υποσύνολα που ζητούνται. Το training set περιέχει 120 δείγματα και το test set 30 δείγματα.

Β. Να εκπαιδευτεί νευρωνικό δίκτυο 2 επιπέδων για την ταξινόμηση των δειγμάτων, όπου το 1^{ο} layer θα έχει 30 νευρώνες και συνάρτηση ενεργοποίησης θα είναι sigmoid.

Για τη δημιουργία του μοντέλου, την εκπαίδευση και το testing χρησιμοποιήθηκε η βιβλιοθήκη Pytorch.

Τα δεδομένα από Numpy arrays πρέπει να μετατραπούν σε Tensors για να χρησιμοποιηθούν οι συναρτήσεις της Pytorch. Εδώ επιλέξαμε να μην χρησιμοποιήσουμε data generators.

Επίσης, ο κώδικας έχει σχεδιαστεί με τέτοιο τρόπο ώστε να τρέχει σε GPU αν βρει κάποια διαθέσιμη, αλλιώς χρησιμοποιεί τη CPU. Αυτό γίνεται με την παρακάτω γραμμή κώδικα.

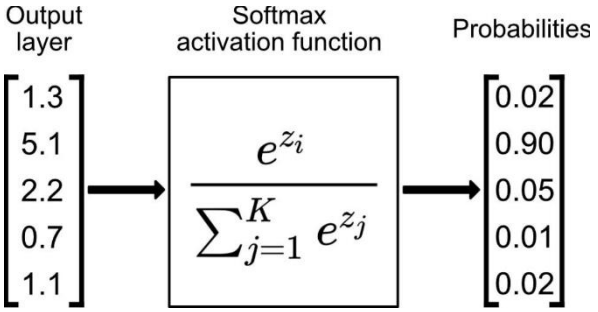
```
device = "cuda" if torch.cuda.is_available() else "cpu"
```

Πρέπει να προσέχουμε κάθε φορά το μοντέλο και τα δεδομένα να βρίσκονται στο ίδιο device. Επίσης, για να κάνουμε διαγράμματα με τη Matplotlib τα δεδομένα θα πρέπει να μετατραπούν από Tensors σε Numpy arrays. Όμως, η Numpy λειτουργεί μόνο στη CPU,

οπότε σε πολλά σημεία πριν γίνουν διαγράμματα γίνεται αυτή η μετατροπή και η μεταφορά στη CPU.

Στη συνέχεια, προχωράμε στη δημιουργία του μοντέλου και στην εκπαίδευση. Το 1ο layer έχει 30 νευρώνες και η συνάρτηση ενεργοποίησης είναι sigmoid και το 2ο layer έχει 3 νευρώνες (όσες και οι κλάσεις που υπάρχουν στο dataset) και η συνάρτηση ενεργοποίησης είναι softmax. Το input shape του μοντέλου (input 1ου layer) είναι όσα και τα features που υπάρχουν στο dataset, δηλαδή 4.

Σε προβλήματα multiclass classification το output layer έχει τόσους νευρώνες όσες και οι κλάσεις (one versus all προσέγγιση-κωδικοποίηση one hot). Στην πραγματικότητα κάθε νευρώνας του επιπέδου εξόδου παράγει έναν αριθμό. Αυτοί οι αριθμοί ονομάζονται logits και αντιπροσωπεύουν μη κανονικοποιημένες προβλέψεις. Στη συνέχεια τα logits εισάγονται στη softmax, η οποία κανονικοποιεί τις προβλέψεις, δηλαδή τις μετατρέπει σε πιθανότητες. Τέλος, επιλέγεται η θέση της μέγιστης τιμής πιθανότητας που αντιστοιχεί στην κλάση (label) στην οποία ανατίθεται το δείγμα. Ένα παράδειγμα για τη μετατροπή των logits σε πιθανότητες φαίνεται στην παρακάτω εικόνα.



Επίσης, στο multiclass classification χρησιμοποιούμε σαν loss function την Cross Entropy Loss (Categorical Cross Entropy).

Τέλος, επιλέξαμε να χρησιμοποιήσουμε σαν optimizer τον Adams που δίνει συνήθως καλύτερα αποτελέσματα από το stochastic gradient descent (SGD).

Επίσης, επιλέχθηκε να εκπαιδεύσουμε για 25 εποχές (το dataset είναι αρκετά μικρό οπότε αποφεύγουμε να εκπαιδεύσουμε πάρα πολύ για να μην υπάρξει overfitting) και με ρυθμό μάθησης (learning rate) 0.01. Οι δύο αυτές υπερπαράμετροι θα παραμείνουν σταθερές για όλη την άσκηση για να μπορέσουμε στο τέλος να συγκρίνουμε τα διαφορετικά μοντέλα μεταξύ τους.

Κατά την εκπαίδευση, σε κάθε εποχή πραγματοποιείται ένα βήμα training και ένα βήμα testing για να παρακολουθούμε την απόδοση του μοντέλου (στην πραγματικότητα εδώ το testing χρησιμοποιείται σαν validation, κανονικά testing κάνουμε μόνο στο τέλος της εκπαίδευσης). Το training step αποτελείται από το forward pass (με τα τρέχοντα βάρη υπολογισμός εξόδων των νευρώνων), τον υπολογισμό του loss και το backpropagation (ανανέωση παραμέτρων). Η ανανέωση των παραμέτρων γίνεται με βάση όλα τα training δεδομένα σε κάθε βήμα. Αυτό ονομάζεται batch learning και επιλέξαμε να το χρησιμοποιήσουμε, επειδή το dataset είναι πολύ μικρό και μπορεί με να φορτωθεί ολόκληρο σε ένα batch. Το testing step αποτελείται από το forward pass (με τα ανανεωμένα βάρη υπολογισμός εξόδων των νευρώνων) και τον υπολογισμό του loss. Μαζί με το loss υπολογίζουμε και την ακρίβεια του μοντέλου.

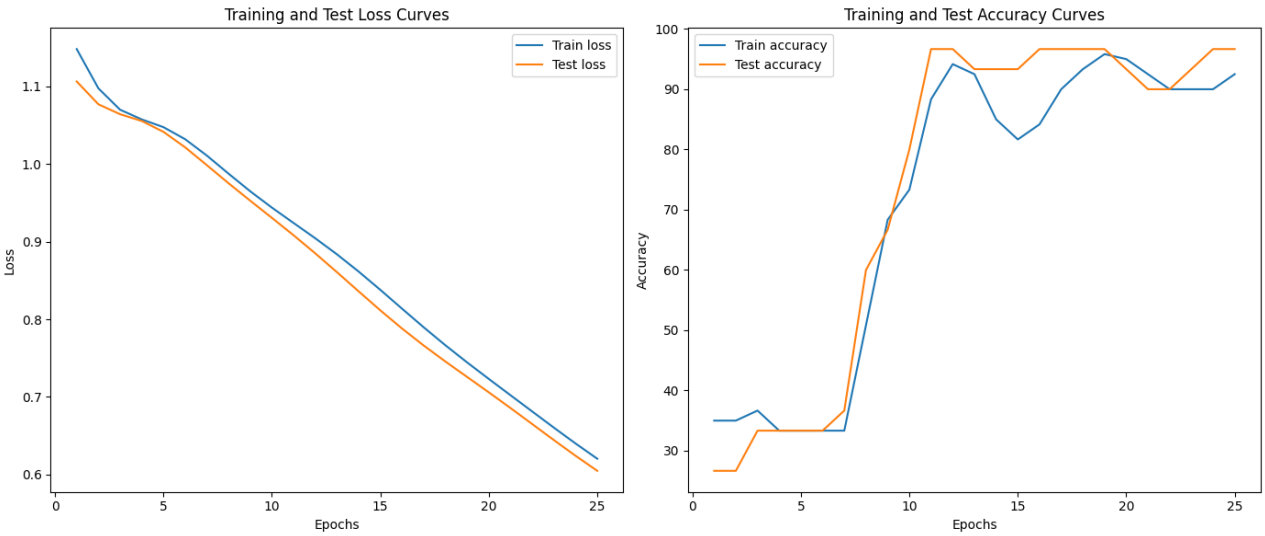
Τα αποτελέσματα της εκπαίδευσης φαίνονται παρακάτω.

```
100% |████████████████████████████████| 25/25 [00:00<00:00, 271.83it/s]
Epoch: 5 | Loss: 1.04733, Acc: 33.33% | Test Loss: 1.04130, Test Acc: 33.33%
Epoch: 10 | Loss: 0.94373, Acc: 73.33% | Test Loss: 0.93055, Test Acc: 80.00%
Epoch: 15 | Loss: 0.83783, Acc: 81.67% | Test Loss: 0.81146, Test Acc: 93.33%
Epoch: 20 | Loss: 0.72318, Acc: 95.00% | Test Loss: 0.70593, Test Acc: 93.33%
Epoch: 25 | Loss: 0.62047, Acc: 92.50% | Test Loss: 0.60489, Test Acc: 96.67%
```

Η ακρίβεια που επιτυγχάνεται με ένα τόσο απλό μοντέλο αντικατοπτρίζει την απλότητα του προβλήματος που προσπαθούμε να επιλύσουμε. Πιθανότατα αν συνεχίζαμε για περισσότερες εποχές την εκπαίδευση θα παίρναμε ακόμα υψηλότερη ακρίβεια, όμως θα σταματήσουμε στις 25 εποχές για να μπορέσουμε παρακάτω να συγκρίνουμε τα διαφορετικά μοντέλα.

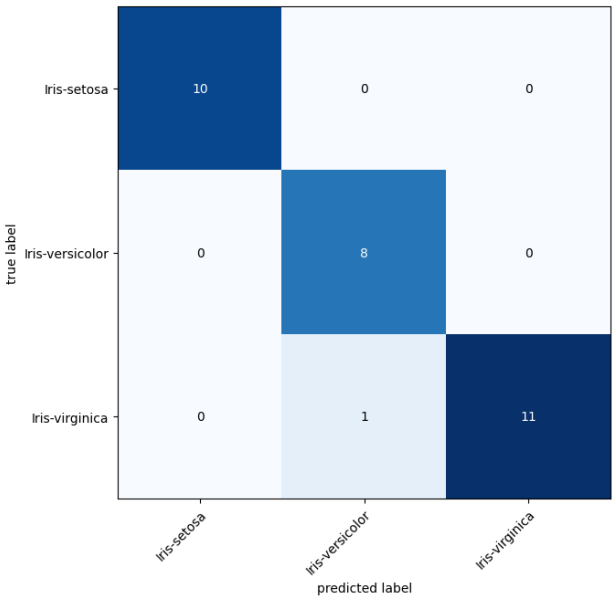
Γ. Να αποτυπωθεί η εξέλιξη του test accuracy μετά από κάθε epoch εκπαίδευσης σε διάγραμμα, και να υπολογιστεί ο πίνακας σύγχυσης του τελικού μοντέλου για το test set.

Η εξέλιξη του accuracy και του loss μετά από κάθε εποχή εκπαίδευσης για το training set και το test set φαίνονται στα παρακάτω διαγράμματα. Το train και test loss συνεχίζουν να πέφτουν στο σημείο που σταματήσαμε την εκπαίδευση, κάτι που υποδηλώνει ότι αν συνεχιστεί η εκπαίδευση θα βελτιωθεί το μοντέλο. Εφόσον τα train και test loss είναι κοντά μεταξύ τους (μάλιστα το test loss είναι μικρότερο από το train loss) καταλαβαίνουμε ότι υπάρχει γενίκευση στο test set και δεν κάνουμε overfitting στο training set.



Στη συνέχεια, χρησιμοποιούμε το μοντέλο για να κάνουμε προβλέψεις στο test set. Εδώ, αυτό το βήμα είναι προαιρετικό γιατί δεν έχουμε validation set, αλλά επειδή σε άλλες περιπτώσεις (σχεδόν πάντα σε μεγαλύτερα dataset χρησιμοποιούμε validation set) είναι απαραίτητο αποφασίσαμε να το κάνουμε. Το αποτέλεσμα πάντως θα είναι το ίδιο με αυτό που είχαμε στην τελευταία εποχή εκπαίδευσης (στο τέλος της εκπαίδευσης).

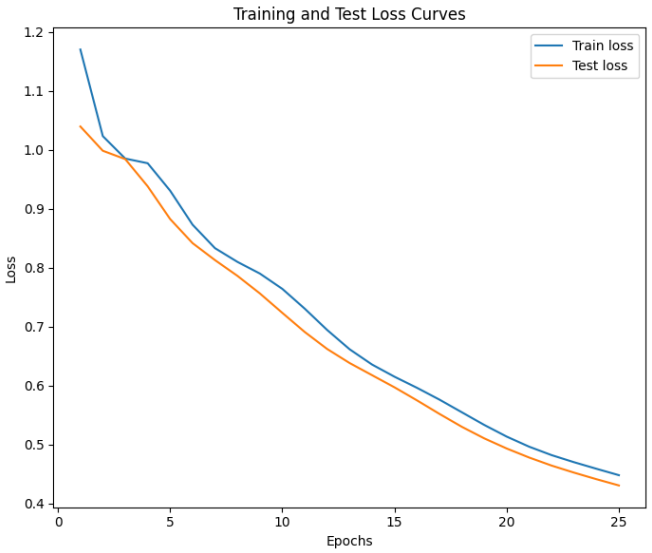
Με βάση τις προβλέψεις στο test set και τα πραγματικά labels υπολογίζεται ο πίνακας σύγχυσης. Παρατηρούμε ότι από τα 30 δείγματα, υπάρχει μόνο ένα λάθος (το μοντέλο πρόβλεψε ότι ένα δείγμα ανήκει στην κλάση Iris-versicolor ενώ η πραγματική κλάση που ανήκει είναι η Iris-virginica). Επίσης, δεν έχουν όλες οι κλάσεις ίδιο αριθμό δειγμάτων. Γενικά αν το συνολικό dataset είναι balanced, είναι καλό να διατηρούμε ίδιο αριθμό δειγμάτων για κάθε κλάση και στο test set. Εδώ αυτό δεν έγινε, παρ' όλα αυτά όμως όπως φαίνεται και παρακάτω η απόδοση όλων των μοντέλων είναι πάρα πολύ καλή.

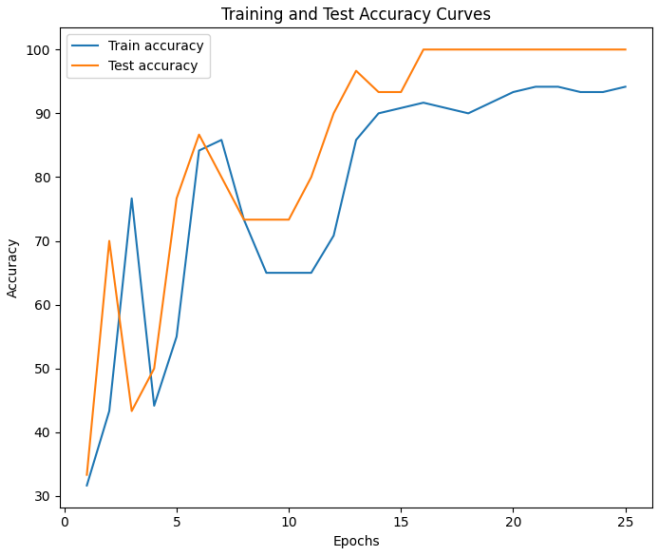


Δ. Να επαναληφθεί η εκπαίδευση με συνάρτηση ενεργοποίησης την ReLu. Τί παρατηρείτε;

Επαναλάβαμε την εκπαίδευση αλλάζοντας μόνο τη συνάρτηση ενεργοποίησης του 1ου layer, Αντί για sigmoid δοκιμάσαμε ReLU. Τα αποτελέσματα της εκπαίδευσης είναι τα ακόλουθα.

```
100% |████████████████████| 25/25 [00:00<00:00, 262.57it/s]
Epoch: 5 | Loss: 0.93075, Acc: 55.00% | Test Loss: 0.88260, Test Acc: 76.67%
Epoch: 10 | Loss: 0.76405, Acc: 65.00% | Test Loss: 0.72332, Test Acc: 73.33%
Epoch: 15 | Loss: 0.61493, Acc: 90.83% | Test Loss: 0.59692, Test Acc: 93.33%
Epoch: 20 | Loss: 0.51328, Acc: 93.33% | Test Loss: 0.49308, Test Acc: 100.00%
Epoch: 25 | Loss: 0.44780, Acc: 94.17% | Test Loss: 0.43037, Test Acc: 100.00%
```





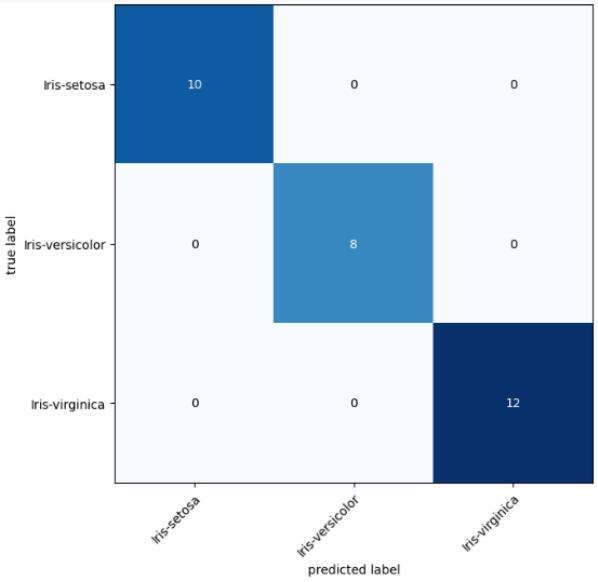
Η βελτίωση είναι εμφανής. Από την εποχή 20 και μετά το μοντέλο επιτυγχάνει 100% ακρίβεια στο test set. Αυτό σημαίνει ότι όλες οι προβλέψεις στο test set είναι σωστές, κάτι που φαίνεται και στον πίνακα σύγχυσης.

Αν και το loss πάντα μειώνεται, το accuracy κάποιες φορές αντί να ανεβαίνει μόνο, κατεβαίνει κιόλας, εφόσον όμως η συνολική ακρίβεια αυξάνεται στη συνέχεια δεν έχουμε πρόβλημα.

Βέβαια, το train accuracy είναι 94.17%. Αυτό σημαίνει ότι υπάρχει ακόμα περιθώριο βελτίωσης και αυτό φαίνεται και από τα loss που συνεχίζουν να πέφτουν (δεν έχουν σταθεροποιηθεί). Η σύγκριση των μοντέλων γίνεται συνήθως με το test accuracy ή το test loss.

Συμπεραίνουμε ότι ο τυχαίος διαχωρισμός (random split) του dataset έχει οδηγήσει σε ένα test set το οποίο είναι αρκετά εύκολο να διαχωριστεί, ενώ το train set είναι πιο απαιτητικό.

Παρ' όλα αυτά το αποτέλεσμα με τη χρήση ReLU είναι πολύ καλύτερο από αυτό με τη χρήση sigmoid συνάρτησης ενεργοποίησης και στο training set, αλλά και στο test set. Επιτυγχάνεται καλύτερο αποτέλεσμα και σε λιγότερες εποχές (πιθανώς αν εκπαιδεύαμε για περισσότερες εποχές στο ερώτημα Β να πετυχαίναμε και εκεί 100% test accuracy). Η βελτίωση δε φαίνεται μόνο από το test accuracy αλλά και από το test loss, το οποίο είναι μικρότερο από πριν.



Γενικά καλό είναι εκτός από το accuracy να παρατηρούμε και το loss, διότι μπορεί το μοντέλο να κάνει σωστές προβλέψεις αλλά να μην είναι πολύ confident για αυτές, το confidence (πόσο σίγουρο είναι το μοντέλο ότι έκανε σωστή πρόβλεψη) φαίνεται από το loss.

Ε. Πειραματιστείτε με τις παραμέτρους σχεδίασης, και προτείνετε ένα δίκτυο που επιτυγχάνει καλύτερη ακρίβεια από τα προηγούμενα. Τεκμηριώστε το αποτέλεσμα με τα απαραίτητα γραφήματα και πίνακες σύγχυσης.

Αποφασίσαμε να αυξήσουμε τα επίπεδα του δικτύου κατά ένα αλλά και τον αριθμό των νευρώνων σε κάθε επίπεδο. Πλέον το 1ο και το 2ο layer έχουν 60 νευρώνες και συνάρτηση ενεργοποίησης ReLU το καθένα. Το 3ο layer έχει 3 νευρώνες.

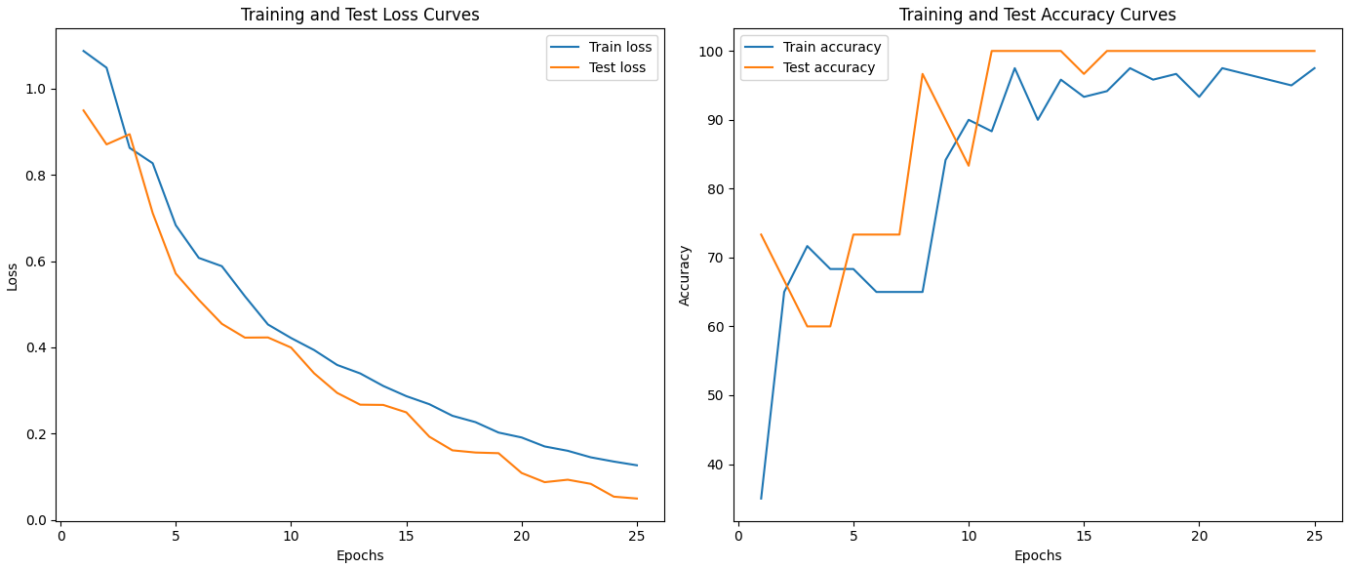
Το δίκτυο είναι πολύ μεγαλύτερο από πριν, οπότε πρέπει να δοθεί προσοχή στο αν υπάρχει overfitting. Η αύξηση έγινε σταδιακά (πρώτα αυξήσαμε τους νευρώνες και μετά δοκιμάσαμε να προσθέσουμε ένα επίπεδο), μέχρι να πετύχουμε την ελαχιστοποίηση του test loss σε κάποια πολύ χαμηλή τιμή. Η αρχιτεκτονική στην οποία καταλήξαμε είναι αυτή που περιγράφεται παραπάνω.

Τα αποτελέσματα της εκπαίδευσης.

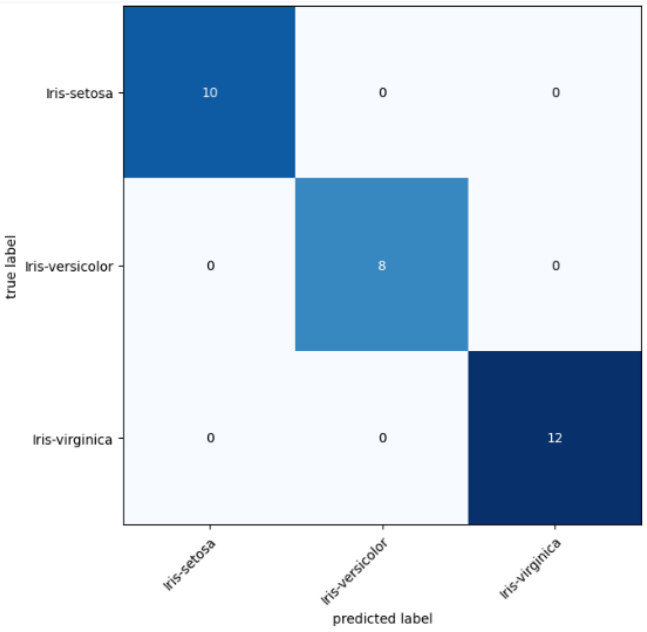
```
100% |████████████████████| 25/25 [00:00<00:00, 168.97it/s]
Epoch: 5 | Loss: 0.68319, Acc: 68.33% | Test Loss: 0.57111, Test Acc: 73.33%
Epoch: 10 | Loss: 0.42176, Acc: 90.00% | Test Loss: 0.39965, Test Acc: 83.33%
Epoch: 15 | Loss: 0.28698, Acc: 93.33% | Test Loss: 0.24945, Test Acc: 96.67%
Epoch: 20 | Loss: 0.19122, Acc: 93.33% | Test Loss: 0.10892, Test Acc: 100.00%
Epoch: 25 | Loss: 0.12677, Acc: 97.50% | Test Loss: 0.04972, Test Acc: 100.00%
```

Πάλι το test accuracy είναι 100%. Το train accuracy αυξήθηκε, αυτό σημαίνει ότι και στο training set υπάρχει βελτίωση. Γενικά, αν το test accuracy είναι μεγαλύτερο από το train accuracy, υπάρχει underfitting (υπάρχουν περιθώρια βελτίωσης του μοντέλου). Οπότε, πλέον έχει μειωθεί το underfitting, αξιοποιείται μεγαλύτερο μέρος πληροφορίας που υπάρχει στο training set, σε σχέση με το προηγούμενο μοντέλο. Αυτό είναι καλό γιατί σε περίπτωση που βρεθούν νέα πιο απαιτητικά δείγματα από αυτά που έχουμε στο test set, το μοντέλο είναι πιθανότερο να καταφέρει να τα προβλέψει σωστά (επίτευξη γενίκευσης).

Προφανώς δεν υπάρχει overfitting όταν test accuracy είναι μεγαλύτερο από το train accuracy. Οπότε, θα μπορούσαμε να αυξήσουμε το μέγεθος του μοντέλου μέχρι το σημείο όπου θα εξαλειφθεί το underfitting, αλλά πρέπει να σταματήσουμε πριν γίνει overfitting.



Θέλουμε οι καμπύλες του train και test loss να πλησιάσουν όσο το δυνατόν περισσότερο. Αυτό βέβαια ίσως γίνει και αν εκπαιδεύσουμε για περισσότερες εποχές, η εκπαίδευση κανονικά πρέπει να σταματήσει μόλις δούμε τις καμπύλες να σταθεροποιούνται σε κάποια τιμή (πχ. το loss και στα δύο set να μην μειώνεται άλλο).



Προφανώς ο πίνακας σύγχυσης είναι ίδιος με του ερωτήματος Δ, γιατί εξαρτάται από το test accuracy.

Συγκεντρωτικά τα αποτελέσματα από τα 3 μοντέλα φαίνονται στον παρακάτω πίνακα.

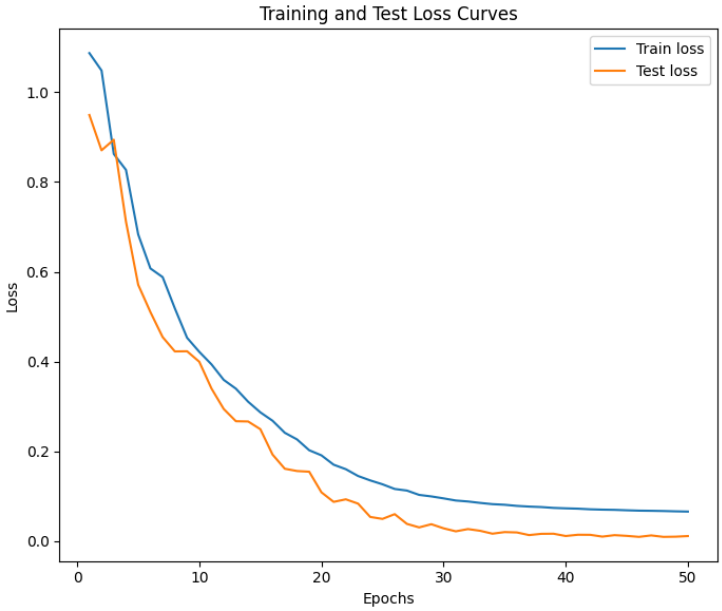
	model_name	train_loss	train_acc	test_loss	test_acc
0	IrisModelV0	0.62	92.50	0.60	96.67
1	IrisModelV1	0.45	94.17	0.43	100.00
2	IrisModelV2	0.13	97.50	0.05	100.00

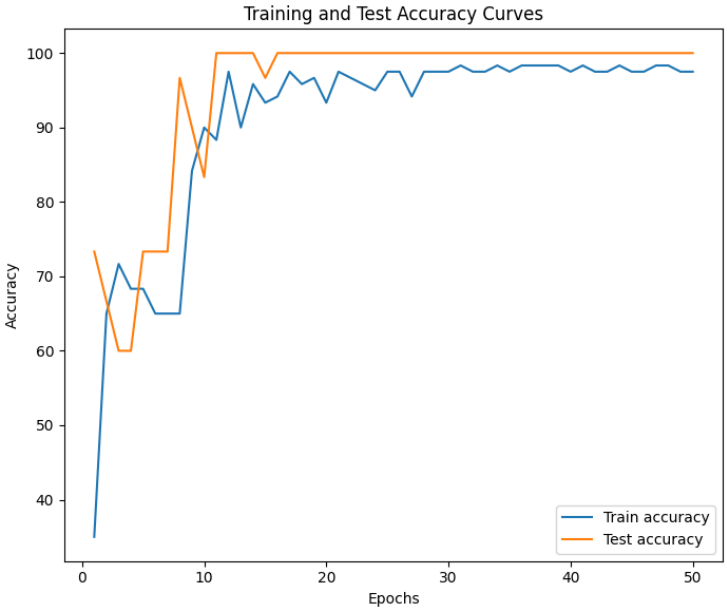
Στο καλύτερό μας μοντέλο, αν και εκπαιδεύουμε μόνο για 25 εποχές το test loss έχει πέσει στο 0.05 (0 loss σημαίνει ότι όλα έχουν προβλεφθεί σωστά με confidence 100%).

Αν εκπαιδεύσουμε για περισσότερες εποχές (50 epochs) τότε τα αποτελέσματα φαίνονται στον παρακάτω πίνακα. Το test loss μειώθηκε ακόμα περισσότερο για όλα τα μοντέλα και πλέον για όλα το test accuracy είναι 100%.

	model_name	train_loss	train_acc	test_loss	test_acc
0	IrisModelV0	0.36	96.67	0.34	100.0
1	IrisModelV1	0.22	95.83	0.19	100.0
2	IrisModelV2	0.07	97.50	0.01	100.0

Για το καλύτερο μοντέλο από τις παρακάτω καμπύλες έχουμε ότι περίπου μετά την εποχή 40 το loss σταθεροποιείται. Εφόσον, ακόμα το test loss είναι μικρότερο από το train loss, υπάρχει σε κάποιο βαθμό underfitting το οποίο όμως δεν μπορεί να μειωθεί με περεταίρω εκπαίδευση. Αν θέλουμε να το μειώσουμε ίσως να δοκιμάζαμε να μεγαλώσουμε κι άλλο το μοντέλο.





Με διαφορετικό random split για 25 εποχές έχουμε:

	model_name	train_loss	train_acc	test_loss	test_acc
0	IrisModelV0	0.57	82.50	0.59	76.67
1	IrisModelV1	0.43	80.83	0.46	73.33
2	IrisModelV2	0.19	96.67	0.15	100.00

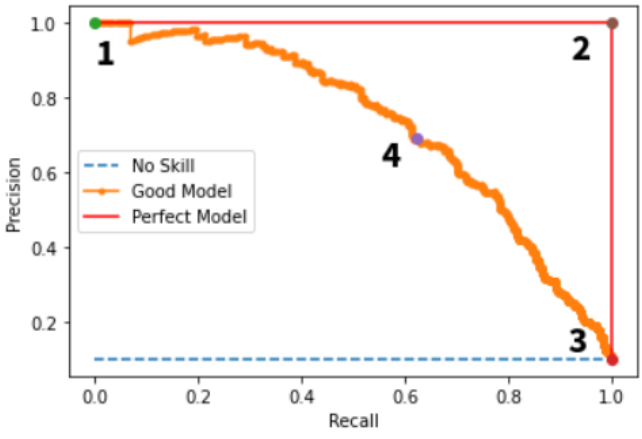
Και για 50 εποχές έχουμε:

	model_name	train_loss	train_acc	test_loss	test_acc
0	IrisModelV0	0.31	95.83	0.30	100.0
1	IrisModelV1	0.23	96.67	0.20	100.0
2	IrisModelV2	0.08	97.50	0.01	100.0

Αν και στις 25 εποχές τα δύο πρώτα μοντέλα δεν καταφέρνουν να εκπαιδευτούν τόσο καλά όσο πριν, στις 50 εποχές τα αποτελέσματα είναι σχεδόν ίδια με πριν. Οπότε, τα μοντέλα μας αν εκπαιδευτούν για αρκετές εποχές, μπορούν να δώσουν παρόμοια αποτελέσματα για διαφορετικά splits των δεδομένων, κάτι που είναι και το επιθυμητό. Άρα, κρατάμε τις 50 εποχές για εκπαίδευση. Στο notebook υπάρχουν τα διαγράμματα για 50 εποχές, παραπάνω τα διαγράμματα (με εξαίρεση ένα διάγραμμα) αναφέρονται σε 25 εποχές (για λόγους σύγκρισης).

ΣΤ. Στο καλύτερο δίκτυο από αυτά που εκπαιδεύσατε, υπολογίστε και αποτυπώστε τις καμπύλες precision-recall για τις τρεις κλάσεις, μεταβάλλοντας το κατώφλι της πιθανότητας κάθε κλάσης. Ποια κλάση είναι πιο εύκολα διαχωρίσιμη με βάση το AUC (Area Under Curve) που προκύπτει από τις καμπύλες;

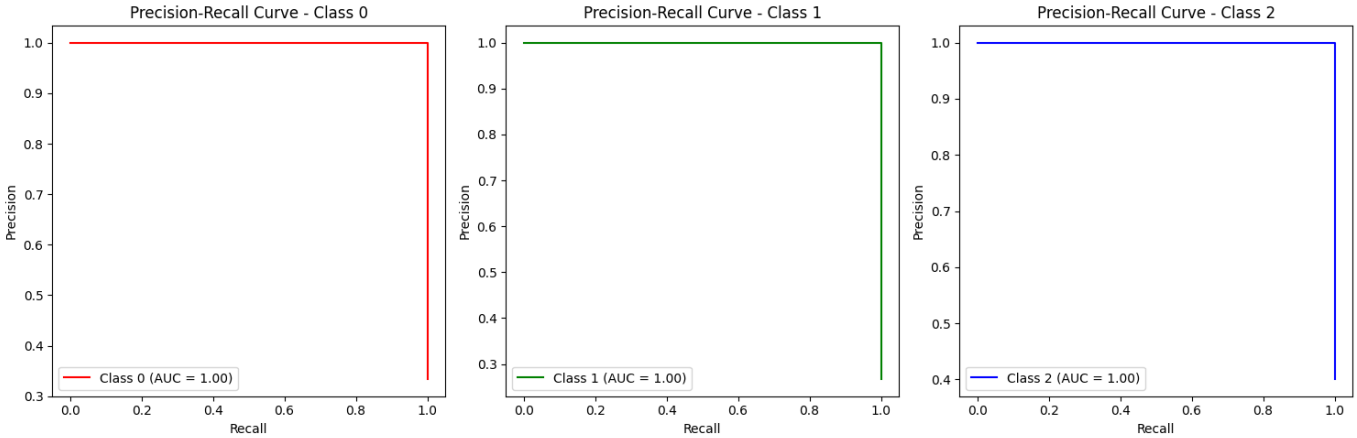
Όσον αφορά τις καμπύλες precision-recall, όσο απλώνονται προς τα πάνω και δεξιά, τόσο καλύτερο είναι το μοντέλο (περισσότερες σωστές προβλέψεις, καθώς το confidence threshold μεταβάλλεται).



Το confidence threshold, είναι στην ουσία το όριο πάνω από το οποίο πρέπει να βρίσκεται η προβλεπόμενη πιθανότητα για μία κλάση για να θεωρηθεί ότι προβλέφθηκε σωστά (True positive).

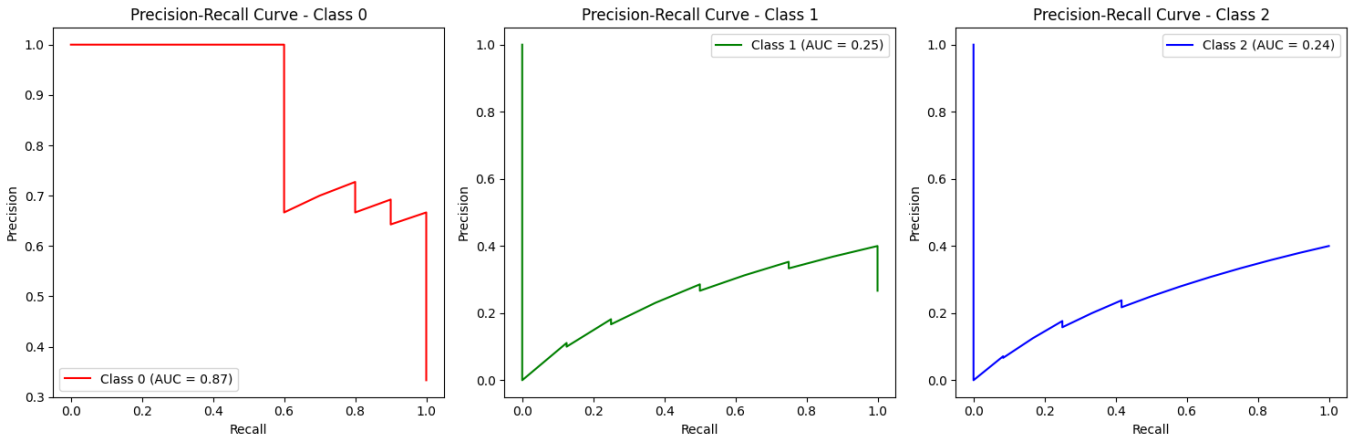
Στο διάγραμμα, όσο πιο αριστερά βρισκόμαστε, τόσο το confidence threshold και το precision αυξάνονται. Όσο πάμε προς τα δεξιά, τόσο μικρότερο το confidence threshold και το recall αυξάνεται.

Το καλύτερο δίκτυο είναι αυτό με το μικρότερο test loss, δηλαδή αυτό που πέτυχε 0.01 test loss (IrisModelV2 για 50 εποχές). Οι καμπύλες precision-recall για τις τρεις κλάσεις, μεταβάλλοντας το κατώφλι της πιθανότητας κάθε κλάσης φαίνονται στην παρακάτω εικόνα.



Το AUC (Area under curve) είναι η περιοχή που περικλείεται από την κάθε καμπύλη. Όσο μεγαλύτερο, τόσο καλύτερο το μοντέλο. Στα παραπάνω διαγράμματα έχουμε AUC ίσο με μονάδα για όλες τις κλάσεις. Αυτό σημαίνει ότι το μοντέλο επιτυγχάνει μέγιστη ακρίβεια (100%) στην ταξινόμηση, ακόμα και αν το confidence threshold αυξηθεί πάρα πολύ. Με άλλα λόγια το μοντέλο κάνει σωστές προβλέψεις και είναι σίγουρο για αυτές. Με βάση το AUC που υπολογίστηκε για κάθε κλάση, συμπεραίνουμε ότι όλες οι κλάσεις είναι εύκολα διαχωρίσιμες, με τη χρήση του συγκεκριμένου μοντέλου.

Αν υπολογίσουμε τις καμπύλες πριν εκπαιδευτεί το μοντέλο (περίπου 26.67% test accuracy), έχουμε τις παρακάτω καμπύλες. Εδώ φαίνεται ξεκάθαρα ότι για το μη εκπαιδευμένο μοντέλο, η πιο εύκολα διαχωρίσιμη κλάση με βάση το AUC είναι η κλάση 0.



Bonus (Cifar 10)

Το Cifar 10 είναι ένα dataset ταξινόμησης εικόνων σε 10 κατηγορίες. Προκειμένου να κατανοήσουμε περισσότερο τις ιδιότητες των νευρωνικών δικτύων θα χρησιμοποιήσουμε τα μοντέλα της άσκησης 1 για να δούμε την απόδοσή τους στο Cifar 10.

Ενδεικτικά κάποιες εικόνες του dataset φαίνονται παρακάτω.

Πλέον στο Cifar 10, επειδή είναι αρκετά μεγάλο dataset δεν μπορούμε να χρησιμοποιήσουμε batch learning. Οπότε, θα φορτώνουμε τα δεδομένα σε mini batches και σε κάθε mini batch θα γίνεται η ανανέωση των παραμέτρων του δικτύου. Επίσης για να κάνουμε plot το loss και το accuracy σε κάθε εποχή, συσσωρεύουμε τις δύο αυτές μετρικές από κάθε batch και στο τέλος κάθε εποχής υπολογίζουμε τον μέσο όρο για κάθε mini batch.

Οι εποχές που επιλέξαμε είναι 25 και το learning rate το ρυθμίσαμε στο 0.001 (για 0.01 είχαμε πολύ απότομες αυξομειώσεις στις καμπύλες loss και accuracy). Επίσης, το batch size είναι 128. Όπως και πριν σαν optimizer χρησιμοποιούμε τον Adam.

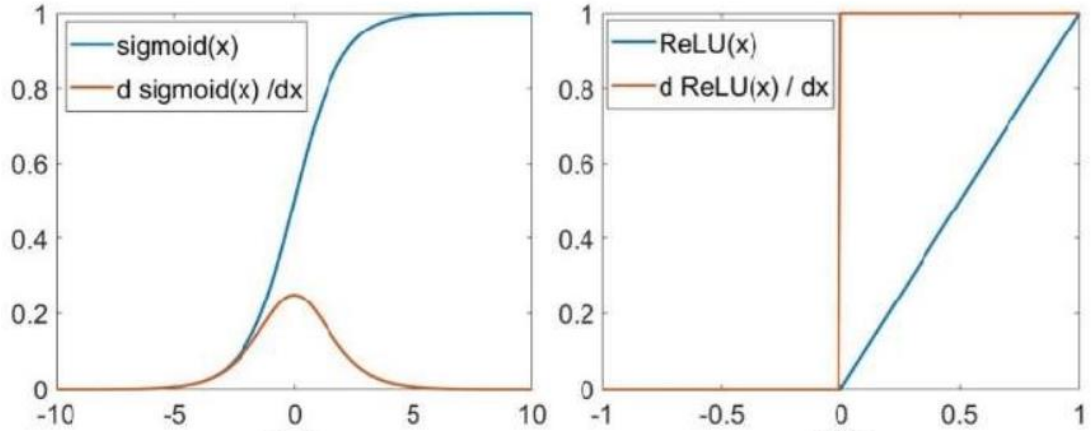
Τα αποτελέσματα από τη σύγκριση των μοντέλων φαίνονται παρακάτω.

	model_name	train_loss	train_acc	test_loss	test_acc
0	Cifar10ModelV0	1.4431	0.4879	1.5450	0.4448
1	Cifar10ModelV1	1.6131	0.4276	1.6483	0.4118
2	Cifar10ModelV2	1.4205	0.4882	1.5260	0.4519
3	Cifar10ModelV3	1.3317	0.5220	1.4281	0.4932
4	Cifar10ModelV4	1.0726	0.6148	1.3756	0.5320
5	Cifar10ModelV4	1.4018	0.4926	1.5711	0.4438

Το μοντέλο 0 και 1 αποτελούνται από ένα layer με 30 νευρώνες και το output layer. Το μοντέλο 0 έχει στο 1^{ο} layer συνάρτηση ενεργοποίησης sigmoid, ενώ το μοντέλο 1 έχει συνάρτηση ενεργοποίησης ReLU. Παρατηρούμε ότι το μοντέλο με τη sigmoid σημείωσε υψηλότερο accuracy και χαμηλότερο loss από αυτό με τη ReLU. Σε αυτήν την περίπτωση η

sigmoid δούλεψε καλύτερα, επειδή έχουμε μόνο ένα hidden layer, ενώ τα πλεονεκτήματα της ReLU φαίνονται όταν έχουμε περισσότερα επίπεδα (βαθύτερο δίκτυο).

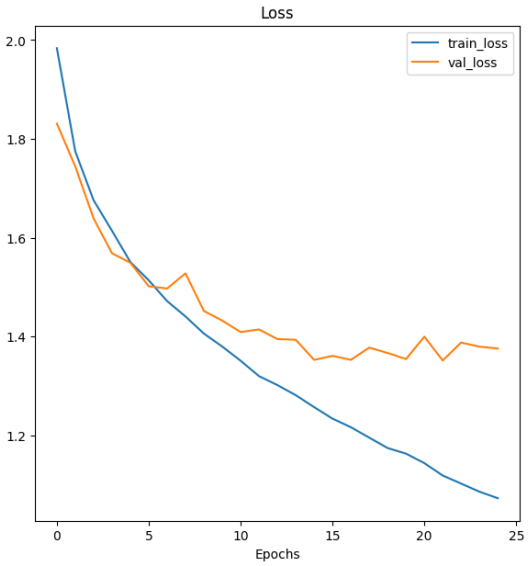
Τα μοντέλα 2 και 3 αποτελούνται από δύο layers με 60 νευρώνες το καθένα και το output layer. Το μοντέλο 2 έχει σε κάθε hidden layer συνάρτηση ενεργοποίησης sigmoid, ενώ το μοντέλο 3 έχει συνάρτηση ενεργοποίησης ReLU. Παρατηρούμε ότι το μοντέλο 3 με τη ReLU τα πήγε καλύτερα. Όπως αναφέραμε και προηγουμένως, η ReLU σηνήθως δουλεύει καλύτερα σε βαθύτερα δίκτυα, επειδή για μεγάλες τιμές η παράγωγός της είναι αρκετά μεγάλη, ενώ στη sigmoid για μεγάλες τιμές η παράγωγός της γίνεται πάρα πολύ μικρή και τείνει να μηδενιστεί, οπότε δεν μπορεί να προχωρήσει σε ανανέωση των βαρών, κατά το backpropagation (vanishing gradient problem).

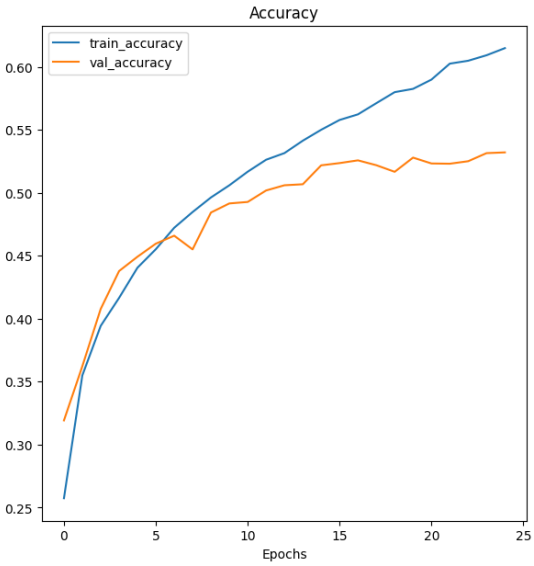


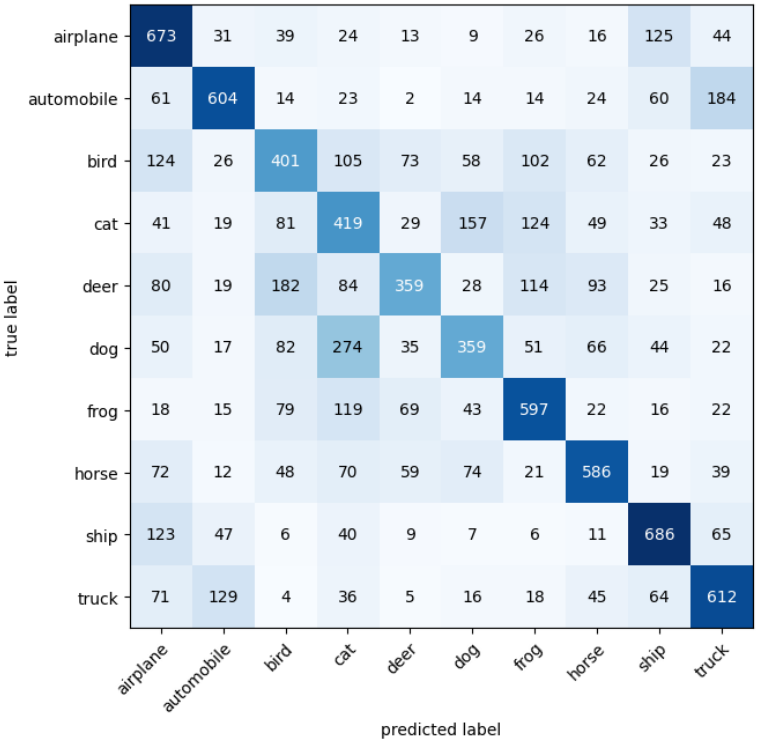
Επίσης, επιλέξαμε να δοκιμάσουμε ένα βαθύτερο νευρωνικό δίκτυο για να παρατηρήσουμε την απόδοσή του. Στο δίκτυο αυτό το 1ο layer έχει 512 νευρώνες, το 2ο layer έχει 256 νευρώνες, το 3ο layer έχει 128 νευρώνες, το 4ο layer έχει 64 νευρώνες, το 5ο layer έχει 32 νευρώνες και όλα έχουν συνάρτηση ενεργοποίησης ReLU. Το 6ο layer έχει 10 νευρώνες (όσες και οι κλάσεις). Αυτό το δίκτυο είναι το μοντέλο 4, το οποίο έχει και την καλύτερη απόδοση (**53.20%** testing accuracy).

Το μοντέλο 5 είναι ακριβώς το ίδιο με το 4 με τη διαφορά ότι οι νευρώνες από layer σε layer αυξάνονται αντί να μειώνονται. Αυτή η δοκιμή δε φάνηκε να δουλεύει καλά, πέτυχε αποτελέσματα παρόμοια με το μοντέλο 0, που έχει μόνο ένα hidden layer. Πιθανότατα επειδή το input είναι πολύ μεγάλο (32x32x3) ίσως για αυτό είναι καλύτερο να ξεκινάμε από πολλούς νευρώνες για να μπορεί το μοντέλο να εξάγει περισσότερες πληροφορίες και στη συνέχεια να τους μειώνουμε (σταδιακή μείωση διαστάσεων σε κάθε νέα αναπαράσταση από επίπεδο σε επίπεδο).

Τα αποτελέσματα για το καλύτερο μοντέλο, το μοντέλο 4 φαίνονται παρακάτω. Είναι φανερό ότι υπάρχει overfitting σε μεγαλύτερο βαθμό από ότι στα προηγούμενα μοντέλα, παρ' όλο που έχουμε πετύχει μεγαλύτερο testing accuracy. Το training accuracy είναι **61.48%**, ενώ το testing accuracy είναι **53.20%**. Η απόσταση αυτή οφείλεται σε overfitting στο training set, που μπορεί να οφείλεται στο μέγεθος του δικτύου.

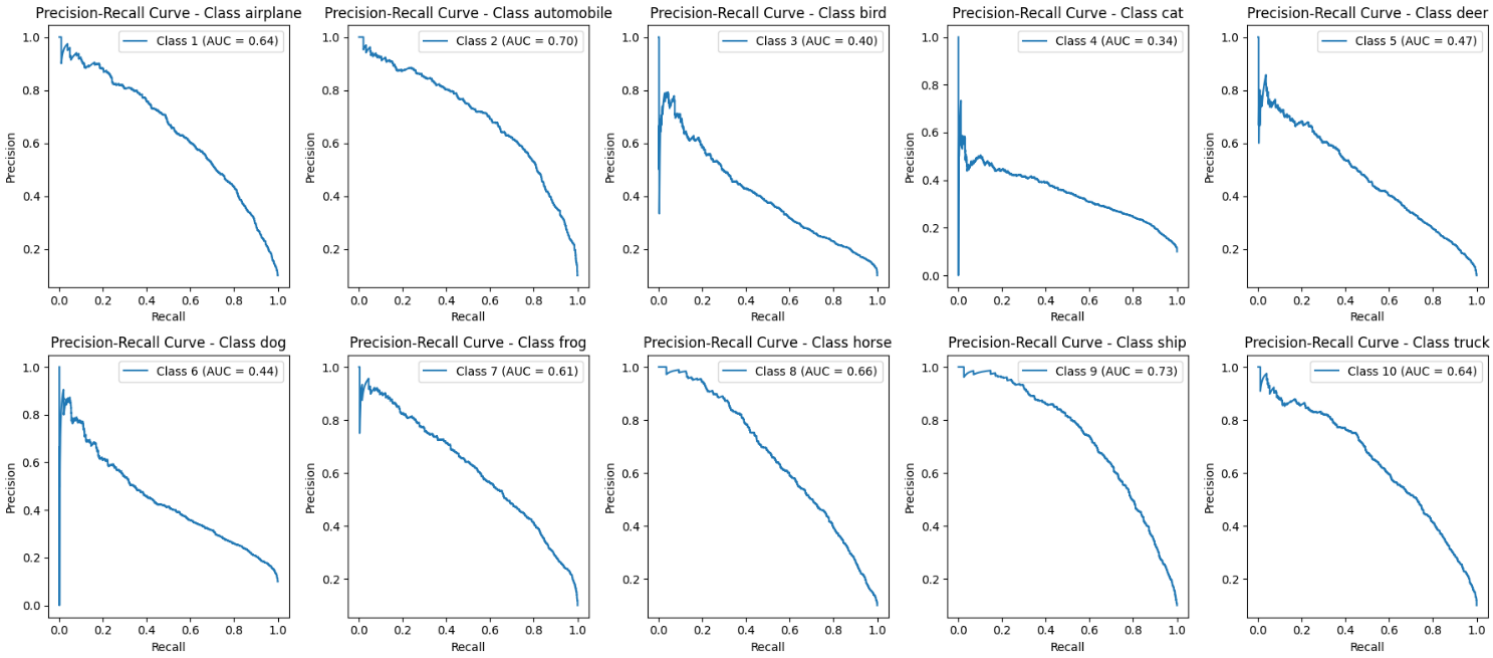






Από τον πίνακα σύγχυσης παρατηρούμε ότι πολλά λάθη γίνονται μεταξύ κλάσεων που μοιάζουν αρκετά μεταξύ τους. Για παράδειγμα, μεταξύ των κλάσεων cat-dog έχουμε πάρα πολλές λανθασμένες προβλέψεις.

Τα διαγράμματα precision-recall φαίνονται παρακάτω. Με βάση το AUC η κλάση που διαχωρίζεται ευκολότερα από τις υπόλοιπες είναι η κλάση ship (AUC 0.73). Ενώ η κλάση που διαχωρίζεται δυσκολότερα είναι η κλάση cat (AUC 0.34).



Άσκηση 2

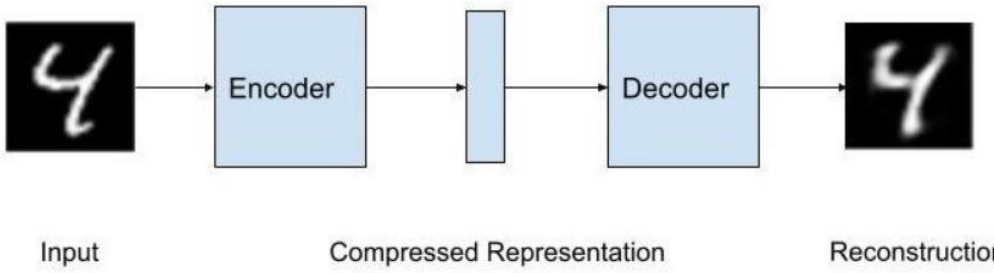
Να εκπαιδεύσετε ένα νευρωνικό δίκτυο με αρχιτεκτονική autoencoder, με 3 επίπεδα κωδικοποίησης 128, 32 και 3 νευρώνων και 3 αντίστοιχα επίπεδα αποκωδικοποίησης, χρησιμοποιώντας το training set της βάσης δεδομένων MNIST Digits.

Τα autoencoders είναι ένα είδος τεχνητών νευρωνικών δικτύων που χρησιμοποιούνται κυρίως για αυτόματη εξαγωγή χαρακτηριστικών και συμπίεση δεδομένων. Αποτελούνται από τον encoder (κωδικοποίηση) και τον decoder (αποκωδικοποίηση).

Ο encoder μειώνει τις διαστάσεις της εισόδου και αναπαριστά τα δεδομένα εισόδου σε ένα κωδικοποιημένο χώρο (latent space). Η σημαντική πληροφορία συμπιέζεται σε πολύ λίγες διαστάσεις. Ο decoder αποκωδικοποιεί την εσωτερική αναπαράσταση και την ανακατασκευάζει στην αρχική της μορφή. Στόχος είναι η εξαγωγή μιας αναπαράστασης που να πλησιάζει όσο το δυνατόν περισσότερο την αρχική.

Για την εκπαίδευση δεν υπάρχουν labels. Το δίκτυο εκπαιδεύεται, ώστε να μειώσει το σφάλμα (loss) μεταξύ της αρχικής εισόδου και της εξόδου (ανακατασκευή με το μικρότερο δυνατό σφάλμα). Επειδή πρόκειται για regression πρόβλημα (πρόβλεψη της τιμής για κάθε pixel της εικόνας εξόδου) το loss function που χρησιμοποιείται είναι το μέσο τετραγωνικό σφάλμα (Mean squared Error/MSE).

Στην παρακάτω εικόνα φαίνεται μια ενδεικτική αρχιτεκτονική autoencoder.



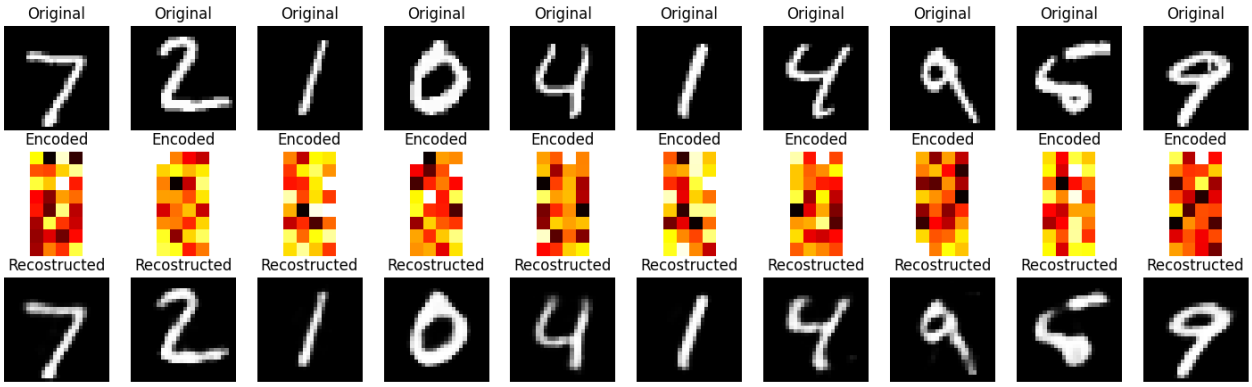
Α. Ποια αρχιτεκτονική autoencoder πέτυχε το μικρότερο μέσο τετραγωνικό σφάλμα στα δεδομένα εκπαίδευσης; Αυτή που χρησιμοποιήθηκε στην εργαστηριακή άσκηση ή αυτή που εκπαιδεύσατε; Γιατί πιστεύετε ότι συνέβη αυτό;

Η αρχιτεκτονική που χρησιμοποιήθηκε στην εργαστηριακή άσκηση έχει δύο επίπεδα κωδικοποίησης 128 και 32 νευρώνων και δύο αντίστοιχα επίπεδα αποκωδικοποίησης. Τα αποτελέσματα της εκπαίδευσης φαίνονται στην εικόνα.

```
Epoch [1/20], Loss: 0.0191
Epoch [2/20], Loss: 0.0130
Epoch [3/20], Loss: 0.0108
Epoch [4/20], Loss: 0.0098
Epoch [5/20], Loss: 0.0079
Epoch [6/20], Loss: 0.0079
Epoch [7/20], Loss: 0.0080
Epoch [8/20], Loss: 0.0080
Epoch [9/20], Loss: 0.0088
Epoch [10/20], Loss: 0.0076
Epoch [11/20], Loss: 0.0063
Epoch [12/20], Loss: 0.0075
Epoch [13/20], Loss: 0.0061
Epoch [14/20], Loss: 0.0067
Epoch [15/20], Loss: 0.0067
Epoch [16/20], Loss: 0.0063
Epoch [17/20], Loss: 0.0061
Epoch [18/20], Loss: 0.0065
Epoch [19/20], Loss: 0.0072
Epoch [20/20], Loss: 0.0073
```

Το μέσο τετραγωνικό σφάλμα στο τέλος της εκπαίδευσης γίνεται **0.0073**, το οποίο είναι μια πολύ καλή τιμή (ιδανικά θα θέλαμε να γίνει μηδέν). Αυτό σημαίνει ότι η ανακατασκευασμένη αναπαράσταση εξόδου ταιριάζει σε πολύ μεγάλο βαθμό με την αρχική αναπαράσταση εισόδου. Στην ουσία, το σφάλμα είναι τόσο μικρό που η ανακατασκευασμένη εικόνα θα μοιάζει σε πολύ μεγάλο βαθμό με την αρχική.

Στη συνέχεια, φαίνονται ενδεικτικά κάποιες εικόνες από το test set, οι αντίστοιχες εικόνες μετά την κωδικοποίηση (32 αριθμοί-διάνυσμα) και οι ανακατασκευασμένες εικόνες μετά την αποκωδικοποίηση. Παρατηρούμε ότι οι ανακατασκευασμένες εικόνες εξόδου είναι σχεδόν ίδιες με τις αρχικές εικόνες (στις περισσότερες δεν μπορούμε να παρατηρήσουμε διαφορές). Ακόμα και στην εικόνα 9 που δεν μπορούμε να διακρίνουμε ποιο ψηφίο αναπαριστά (μοιάζει με το ψηφίο 5), η έξοδος έχει μιμηθεί σε πολύ μεγάλο βαθμό την είσοδο.



Η αρχιτεκτονική που χρησιμοποιήθηκε σε αυτή την άσκηση έχει τρία επίπεδα κωδικοποίησης 128, 32 και 3 νευρώνων και τρία αντίστοιχα επίπεδα αποκωδικοποίησης. Τα αποτελέσματα της εκπαίδευσης φαίνονται παρακάτω.

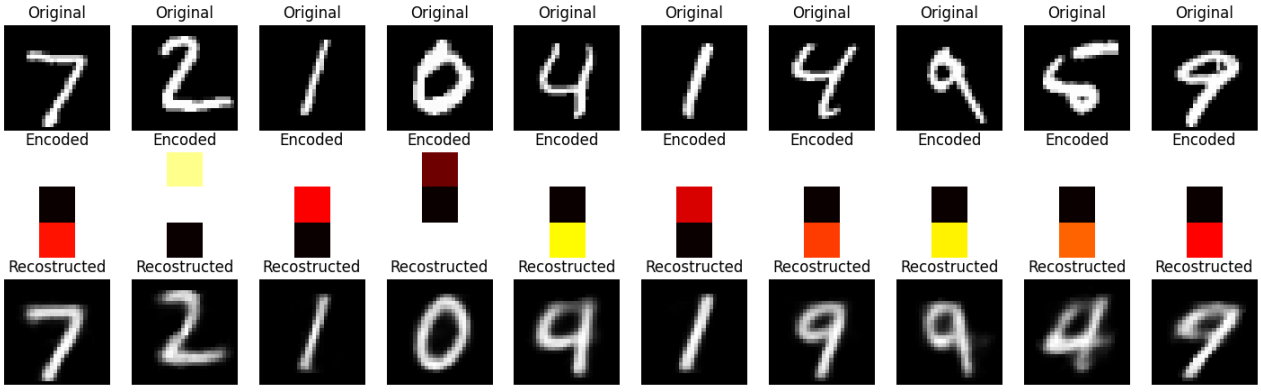
```
Epoch [1/20], Loss: 0.0488
Epoch [2/20], Loss: 0.0367
Epoch [3/20], Loss: 0.0460
Epoch [4/20], Loss: 0.0337
Epoch [5/20], Loss: 0.0369
Epoch [6/20], Loss: 0.0396
Epoch [7/20], Loss: 0.0314
Epoch [8/20], Loss: 0.0318
Epoch [9/20], Loss: 0.0349
Epoch [10/20], Loss: 0.0297
Epoch [11/20], Loss: 0.0308
Epoch [12/20], Loss: 0.0362
Epoch [13/20], Loss: 0.0337
Epoch [14/20], Loss: 0.0406
Epoch [15/20], Loss: 0.0346
Epoch [16/20], Loss: 0.0315
Epoch [17/20], Loss: 0.0299
Epoch [18/20], Loss: 0.0323
Epoch [19/20], Loss: 0.0363
Epoch [20/20], Loss: 0.0333
```

Το μέσο τετραγωνικό σφάλμα στο τέλος της εκπαίδευσης γίνεται **0.0333**, το οποίο είναι μια σχετικά καλή τιμή.

Το σφάλμα είναι πολύ μεγαλύτερο από το σφάλμα της προηγούμενης αρχιτεκτονικής (πριν ήταν **0.0073**).

Από τα σφάλματα συμπεραίνουμε ότι στην αρχιτεκτονική που συμπιέζει τις εικόνες σε latent space με 3 διαστάσεις, χάνεται μεγάλο μέρος σημαντικής πληροφορίας, που είναι απαραίτητη για την ποιότητα της ανακατασκευής. Οπότε, η τελική εικόνα εξόδου δε θα μοιάζει σε τόσο μεγάλο βαθμό με την αρχική εικόνα εξόδου. Στην αρχιτεκτονική που συμπιέζει τις εικόνες σε latent space με 32 διαστάσεις, απ' ότι φαίνεται οι 32 διαστάσεις είναι αρκετές για να διατηρήσουν τη σημαντική πληροφορία που είναι απαραίτητη για την ανακατασκευή. Γι' αυτό το λόγο, η τελική εικόνα εξόδου μοιάζει πάρα πολύ με την αρχική.

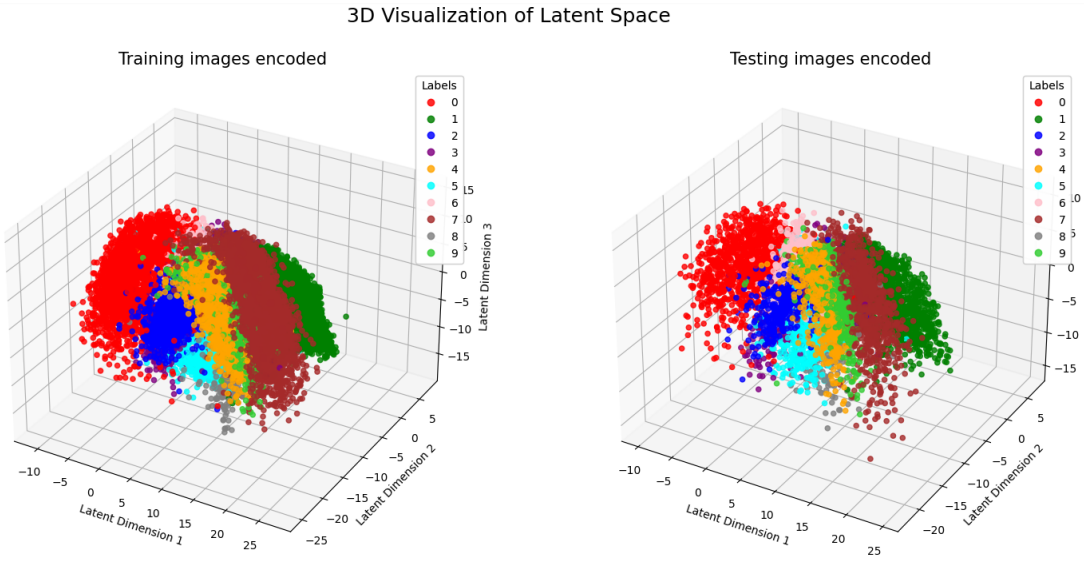
Στη συνέχεια, φαίνονται ενδεικτικά κάποιες εικόνες από το test set, οι αντίστοιχες εικόνες μετά την κωδικοποίηση (3 αριθμοί-διάνυσμα) και οι ανακατασκευασμένες εικόνες μετά την αποκωδικοποίηση. Οι διαφορές που υπάρχουν είναι ορατές σε όλες τις εικόνες. Το λευκό χρώμα στα ψηφία δεν είναι τόσο έντονο, όσο στην αρχική εικόνα. Επίσης, οι εικόνες 5 και 7 (από αριστερά και αν ξεκινήσουμε την αρίθμηση από το 1) έχουν αλλάξει κλάση, πλέον (το ψηφίο 4 έγινε 9). Αυτό λογικά έγινε επειδή οι δύο αυτοί αριθμοί μοιάζουν πολύ μεταξύ τους, άρα θα βρίσκονται πολύ κοντά στο latent space. Οπότε αν το δίκτυο δεν οδηγεί σε τόσο καλή διαχωρισιμότητα στο latent space είναι λογικό δύο κοντινά δείγματα που ανήκουν σε διαφορετικές κλάσεις, κατά την ανακατασκευή οι διαφορές που υπάρχουν να οδηγούν σε διαφορετική κλάση από την αρχική. Επίσης, η εικόνα 9 δεν μοιάζει σχεδόν καθόλου με την ανακατασκευή της.

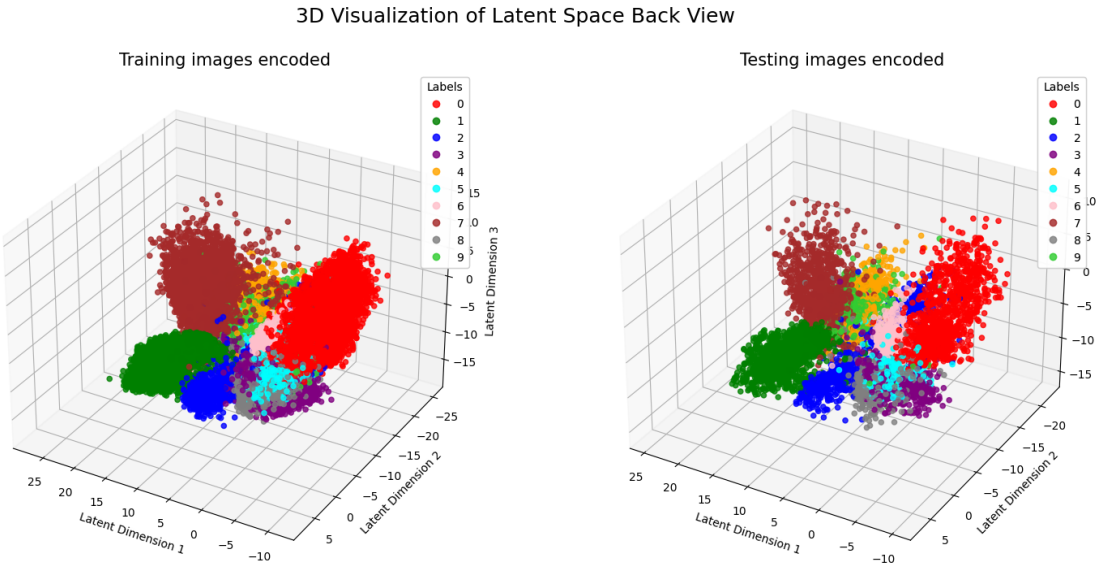


Β. Να απεικονίσετε κατάλληλα το σύνολο των εικόνων του training set όπως κωδικοποιούνται στο latent space, με κατάλληλο χρώμα που να αντιστοιχεί στο label καθεμιάς. Κάνετε το ίδιο και για τις εικόνες του test set. Διατηρούνται τα χαρακτηριστικά της κατανομής των απεικονίσεων και στα δύο σύνολα; Σχολιάστε.

Εφόσον η κωδικοποιημένη αναπαράσταση αποτελείται από 3 αριθμούς, μπορούμε να απεικονίσουμε την κάθε εικόνα, σαν ένα σημείο στο latent space, που στην περίπτωσή μας είναι ένας χώρος 3 διαστάσεων. Από τα διαγράμματα που απεικονίζουν τις εικόνες σαν σημεία στο latent space, παρατηρούμε ότι οι κλάσεις είναι συγκεντρωμένες σε μια περιοχή και βρίσκονται όλες πολύ κοντά μεταξύ τους. Τα κύρια χαρακτηριστικά της κατανομής των απεικονίσεων διατηρούνται και στο training set και στο test set, οι κλάσεις δηλαδή και για τα δύο σύνολα βρίσκονται περίπου στις ίδιες περιοχές. Στην απεικόνιση για το test set τα σημεία είναι πιο αραιά, επειδή έχουμε λιγότερες εικόνες.

Σε αρκετά σημεία η διαχωρισιμότητα των κλάσεων δεν είναι πολύ καλή. Αυτό μπορεί να συμβαίνει επειδή κάποιοι αριθμοί μοιάζουν μεταξύ τους, οπότε στο latent space θα βρίσκονται σε πολύ κοντινές θέσεις (η πληροφορία που διαχώριζε τις δύο κλάσεις έχει πλέον χαθεί). Για παράδειγμα, παρατηρούμε ότι η κλάση 4 με την 9 επικαλύπτονται σε μεγάλο βαθμό. Για αυτό το λόγο, όπως αναφέραμε και στο ερώτημα Α, πολλές φορές το ψηφίο 4, κατά την ανακατασκευή μοιάζει με 9. Αν επιλέξουμε σημεία σε αυτήν την περιοχή και ανακατασκευάσουμε εικόνες από αυτά, δεν θα ήμαστε σίγουροι πιο ψηφίο θα αναπαριστά η εικόνα. Επίσης, η κλάση 3 με 8 επικαλύπτονται σε μεγάλο βαθμό.

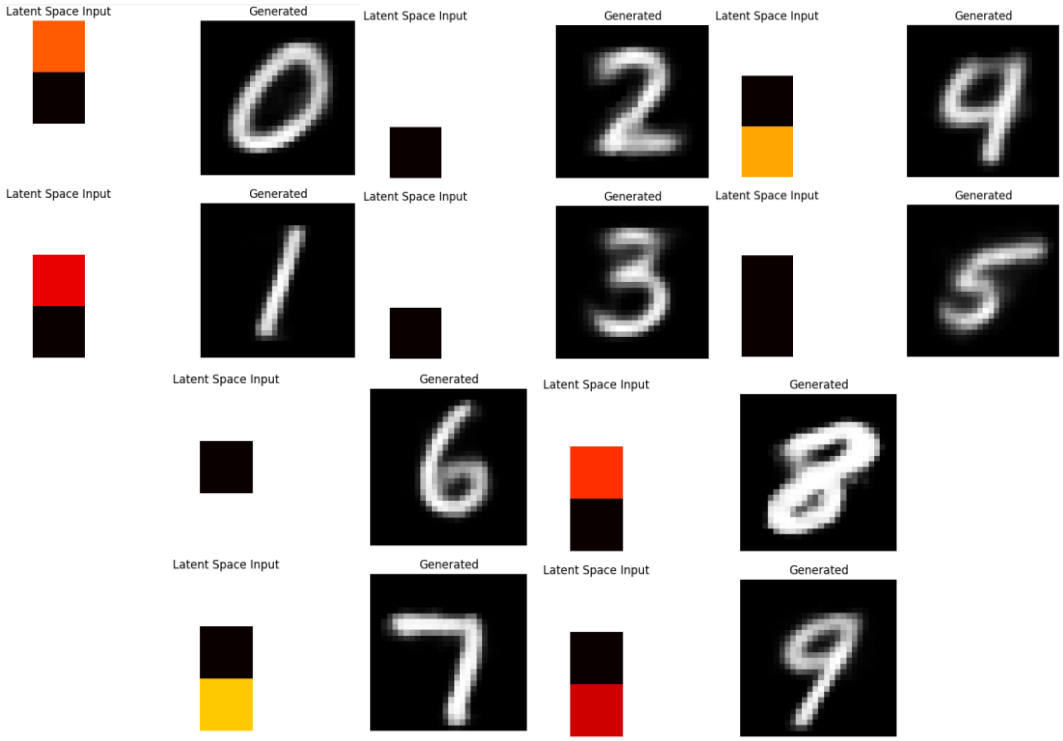




Γ. Να χρησιμοποιήσετε τον κλάδο αποκωδικοποίησης του εκπαιδευμένου autoencoder ώστε να δημιουργήσετε εικόνες ξεκινώντας από τυχαία σημεία του latent space. Εξερευνήστε το latent space δοκιμάζοντας διάφορα σημεία του χώρου και παρατηρώντας τις εικόνες που παράγονται. Αντιστοιχούν όλες οι περιοχές του latent space σε πραγματικά ψηφία; Σχολιάστε τα ευρήματά σας παραθέτοντας παραδείγματα εικόνων.

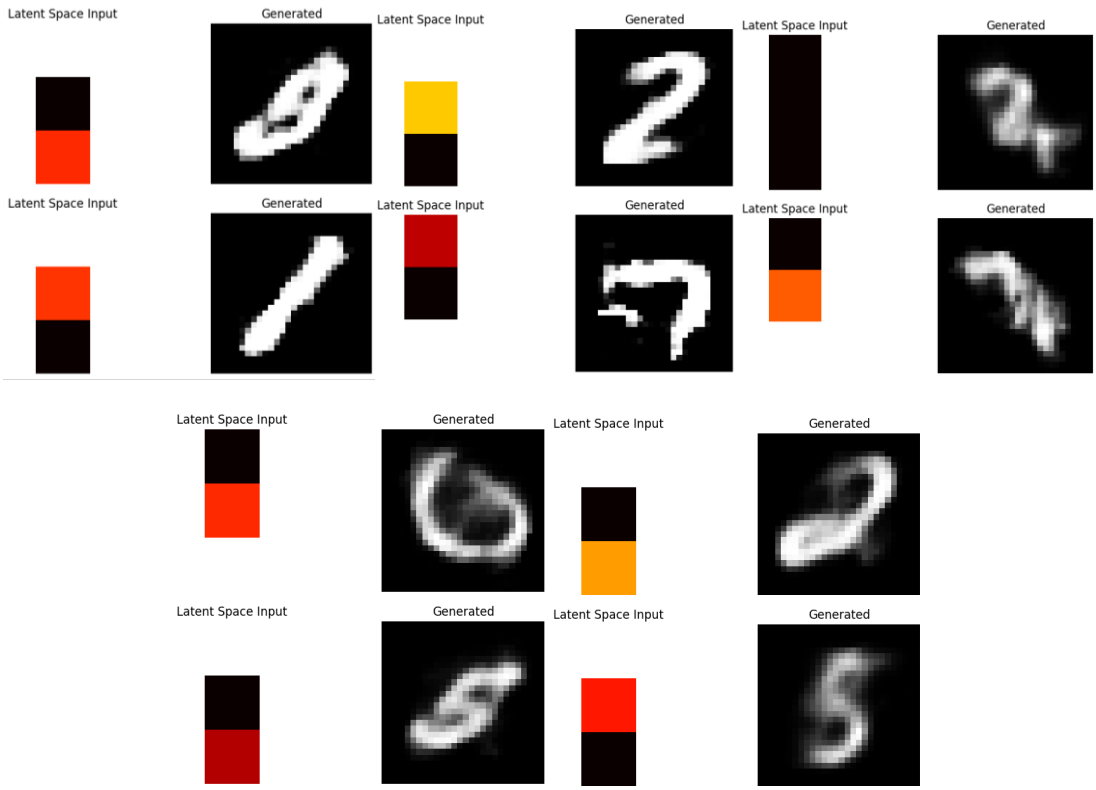
Χρησιμοποιώντας τον κλάδο αποκωδικοποίησης του εκπαιδευμένου autoencoder, μπορούμε δίνοντας τρεις τυχαίους αριθμούς (ένα τυχαίο σημείο στο latent space), να δημιουργήσουμε νέες εικόνες.

Αρχικά, με βάση το διάγραμμα που απεικονίζει τις κλάσεις στο latent space, εισάγουμε σημεία στον decoder, τα οποία βρίσκονται κοντά στις κλάσεις που υπάρχουν για να δούμε αν η εικόνα που θα παραχθεί θα μοιάζει με το ψηφίο της αντίστοιχης κλάσης. Στις περισσότερες εικόνες έχει επιτευχθεί το επιθυμητό αποτέλεσμα, όμως αν το σημείο βρίσκεται στα όρια δύο γειτονικών κλάσεων, η παραγόμενη εικόνα δεν φαίνεται ξεκάθαρα σε ποια κλάση ανήκει (σε κάποιες περιπτώσεις υπάρχει σύγχιση δύο κλάσεων). Εδώ επιλέχθηκαν οι καλύτερες εικόνες που παράχθηκαν (το σημείο που επιλέξαμε κάθε φορά βρίσκεται περίπου στη μέση και όχι στα όρια της κάθε κλάσης), μία για κάθε κλάση. Στην κλάση 4 όσο και αν προσπαθήσαμε, σχεδόν πάντα η παραγόμενη εικόνα μοιάζει να απεικονίζει το ψηφίο 9.



Στη συνέχεια, επιλέξαμε σημεία τα οποία είναι σε κάποιο κενό χώρο στο latent space (χώρος στον οποίο δεν υπάρχουν σημεία του dataset) ή σημεία τα οποία βρίσκονται ανάμεσα σε πολλές κλάσεις (περιοχή στην οποία υπάρχουν δύο ή περισσότερες επικαλυπτόμενες κλάσεις). Σε κάποιες περιπτώσεις το ψηφίο στην παραγόμενη εικόνα φαίνεται ξεκάθαρα σε ποια κλάση ανήκει. Για παράδειγμα, τα ψηφία 1 και 2 στις παρακάτω εικόνες παράχθηκαν πιθανότατα επειδή τα δύο σημεία που επιλέξαμε βρίσκονται πιο κοντά σε αυτές τις κλάσεις μόνο. Οι υπόλοιπες εικόνες δεν μπορούμε ξεκάθαρα να διακρίνουμε ποιο ψηφίο απεικονίζουν. Κάποιες θα μπορούσαμε να φανταστούμε ποιον αριθμό δείχνουν αν και δεν είναι ξεκάθαρο, κάποιες άλλες όμως δεν δείχνουν κάποιο πραγματικό ψηφίο. Οπότε, δεν αντιστοιχούν όλες οι περιοχές του latent space σε πραγματικά ψηφία. Αυτό συμβαίνει επειδή ο autoencoder που έχουμε δημιουργήσει είναι unconstrained. Για να επιλυθεί αυτό το πρόβλημα θα πρέπει να χρησιμοποιηθεί Variational autoencoder.

Ειδικά όταν επιλέγαμε σημεία στα όρια πολλών κλάσεων, η παραγόμενη εικόνα απεικονίζει συνήθως κάτι ενδιάμεσο από όλες τις κλάσεις (αν δεν υπερισχύσει κάποια κλάση πιο πολύ από τις άλλες), οπότε στο τέλος αυτό που παράγεται δεν είναι πλέον κάποιος αριθμός.



Άσκηση 3

Το Breast Cancer Wisconsin είναι ένα σύνολο δεδομένων που περιέχει 30 μορφολογικά χαρακτηριστικά των καρκινικών κυττάρων από 569 βιοψίες μαστού, καθώς και το είδος του όγκου για κάθε δείγμα (κακοήθης: κλάση 1 ή M | καλοήθης: κλάση 0 ή B).

Α. Να χωριστεί το dataset σε 70% training και 30% test δείγματα με τυχαίο τρόπο.

Αρχικά, τα δεδομένα πρέπει να είναι όλα σε αριθμητική μορφή (αριθμοί). Οπότε, με βάση τη στήλη με τα Diagnosis που περιέχει το label κάθε κλάσης, δημιουργούμε τη στήλη class_encoded που περιέχει την κλάση σε αριθμητική μορφή (0 για B και 1 για M). Η στήλη ID δεν περιγράφει κάποιο πραγματικό χαρακτηριστικό, οπότε μπορεί να παραλειφθεί.

Χρησιμοποιώντας τη συνάρτηση train_test_split της sklearn χωρίζουμε το dataset στα υποσύνολα που ζητούνται. Το training set περιέχει 398 δείγματα και το test set 171 δείγματα.

Β. Να μετατρέψετε το 10% των training δεδομένων σε missing values με τυχαία ομοιόμορφη κατανομή, και να εκπαιδεύσετε έναν δενδρικό ταξινομητή (tree classifier) με μέγιστο βάθος 5 επίπεδα. Υπολογίστε την ακρίβεια πρόβλεψης του ταξινομητή στο test set.

Αρχικά, ένας tree classifier εκπαιδεύεται στο training set, στο οποίο δεν υπάρχουν missing values. Στη συνέχεια, αξιολογούμε τον ταξινομητή στο test set και βλέπουμε ότι πετυχαίνει πολύ μεγάλη ακρίβεια.

```
Train with 0% missing values. Accuracy of the tree classifier is 93.57%
```

Αφού το 10% των training δεδομένων μετατρέπονται σε missing values με τυχαία ομοιόμορφη κατανομή, εκπαιδεύουμε ξανά τον tree classifier και τα αποτελέσματα της αξιολόγησης στο test set είναι τα ακόλουθα.

```
Train with 10.0% missing values. Accuracy of the tree classifier is 94.74%
```

Παρατηρούμε ότι ο ταξινομητής επιτυγχάνει μεγαλύτερη ακρίβεια, όταν εκπαιδεύεται στο training set που έχει missing values. Αυτό πιθανότατα συμβαίνει επειδή τα missing values εισάγουν κάποια τυχαιότητα που βοηθάει τον ταξινομητή να επιτύγχει γενίκευση και να δουλεύει καλά και στο test set. Ενώ όταν δεν υπάρχουν missing values, πιθανότατα υπάρχει overfitting σε κάποιο βαθμό, για αυτό και η ακρίβεια στο test set είναι μειωμένη.

Γ. Με τα ίδια training και test sets να εκπαιδεύσετε ένα random forest (RF) με 100 ταξινομητές μέγιστου βάθους 3, χρησιμοποιώντας 5 features ανά δένδρο χωρίς bootstrapping των training δεδομένων. Να υπολογίσετε την ακρίβεια πρόβλεψης του RF στο test set.

Αρχικά, εκπαιδεύουμε ένα Random Forest (RF) ταξινομητή στο training set χωρίς missing values. Η ακρίβεια είναι αρκετά μεγαλύτερη από αυτή που υπολογίσαμε με τον tree ταξινομητή.

```
Train with 0% missing values. Accuracy of the random forest classifier is 97.08%
```

Στη συνέχεια, το 10% των training δεδομένων μετατρέπονται σε missing values με τυχαία ομοιόμορφη κατανομή και εκπαιδεύουμε ξανά τον tree classifier. Επειδή η βιβλιοθήκη sklearn δεν υποστηρίζει εκπαίδευση RF με missing values, εκπαιδεύσαμε 100 δεντρικούς ταξινομητές ξεχωριστά. Στη συνέχεια, για την αξιολόγηση στο test set, πήραμε τον μέσο όρο των προβλέψεων του κάθε δέντρου και χρησιμοποιήσαμε majority voting (αν μέση τιμή προβλέψεων για ένα δείγμα > 0.5, τότε ανήκει στην κλάση 1) για τον υπολογισμό της τελικής κλάσης. Με κατάλληλη ρύθμιση του seed για το κάθε δέντρο, ώστε να μπορούμε να επιτύχουμε reproducability, μπορούμε να λάβουμε ακριβώς την ίδια τιμή για το accuracy με τον RF ταξινομητή που εκπαιδεύτηκε σε δεδομένα χωρίς missing values.

```
Train with 10.0% missing values. Accuracy of the random forest classifier is 97.08%
```

Οπότε, ο ταξινομητής RF εισάγει από μόνος του τυχαιότητα στην επιλογή των χαρακτηριστικών που θα χρησιμοποιηθούν από κάθε δέντρο του, άρα η επιπλέον

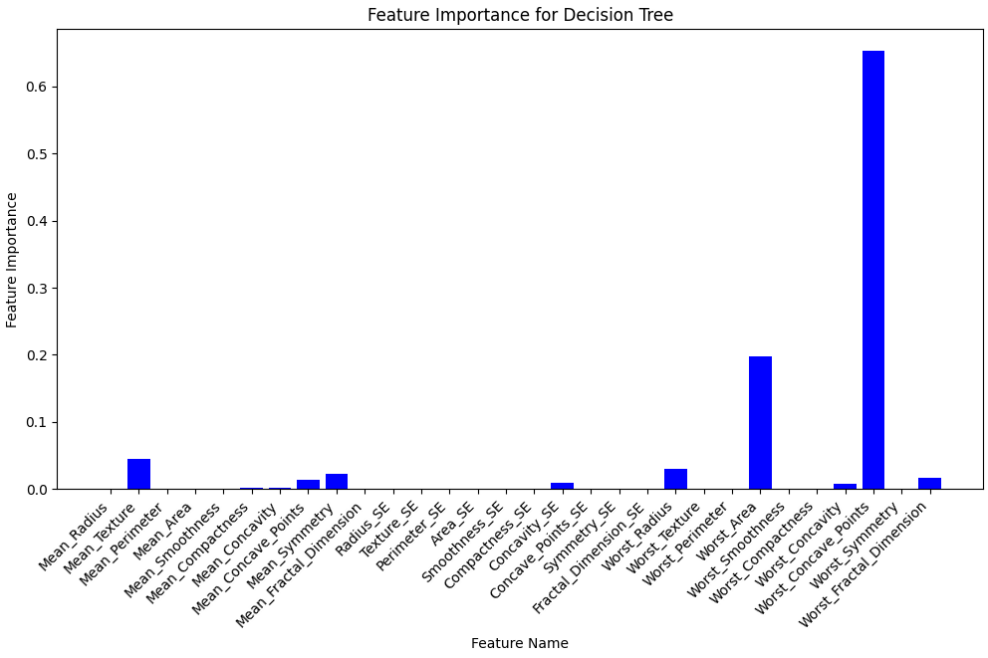
τυχαιότητα που εισάγουν τα missing values είναι είτε αμελητέα (όπως στην περίπτωσή μας), είτε σε ορισμένες περιπτώσεις μπορεί να οδηγήσει σε χαμηλότερο accuracy.

Τέλος, ο ταξινομητής RF είναι καλύτερος από ένα μεμονωμένο tree classifier. Αυτό συμβαίνει επειδή το tree classifier προχωράει σε splits με βάση τα καλύτερα features, με greedy τρόπο, οπότε μπορεί να κολλήσει σε κάποιο τοπικό ελάχιστο (τα σημαντικά features είναι πολύ λίγα). Ενώ, ο RF classifier διερευνά όλες τις περιοχές λύσεων, υπάρχουν πολλά σημαντικά features, άρα σχεδόν πάντα καταλήγει σε καλύτερη λύση.

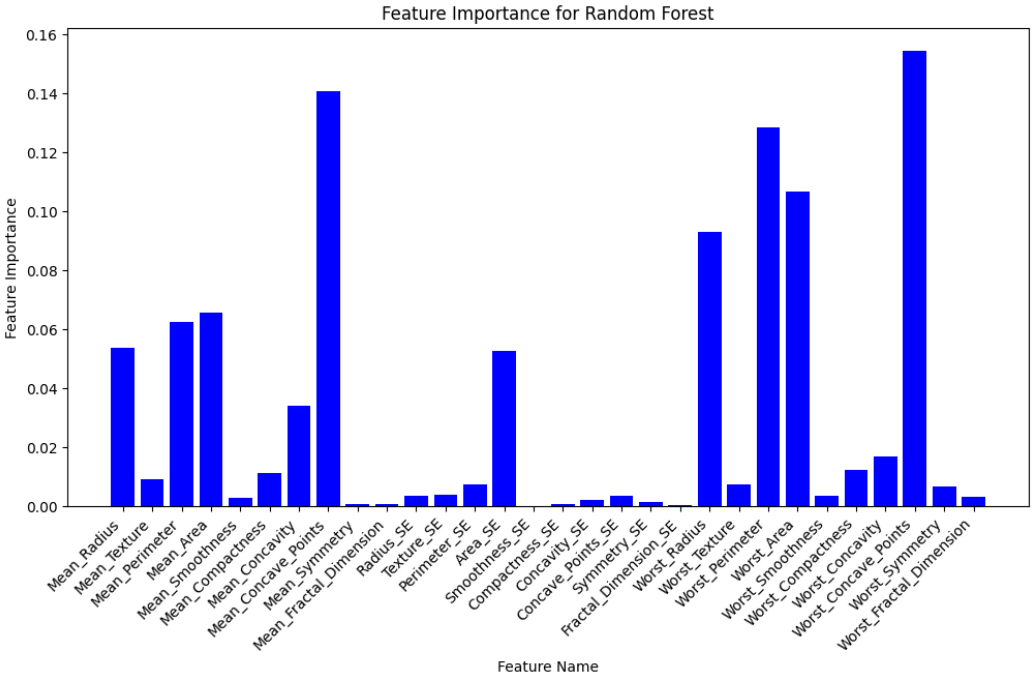
Δ. Υπολογίστε το feature importance για τον δενδρικό ταξινομητή και το RF. Σχολιάστε τα μέχρι τώρα αποτελέσματα και τις διαφορές που παρατηρείτε.

Το feature importance είναι ένας πίνακας που περιέχει τιμές για τη σημαντικότητα κάθε χαρακτηριστικού, στο συγκεκριμένο ταξινομητή. Σε έναν tree classifier, ανάλογα με ποια χαρακτηριστικά χρησιμοποιούνται για να γίνουν τα splits, αυτά τα χαρακτηριστικά είναι πιο σημαντικά για τον ταξινομητή. Σε έναν RF ταξινομητή, τα σημαντικά χαρακτηριστικά είναι το σύνολο όλων των σημαντικών χαρακτηριστικών των δέντρων από τα οποία αποτελείται.

Οι τιμές και το διάγραμμα για το feature importance για τον tree classifier και το RF φαίνονται στις παρακάτω εικόνες.



Features	Importance
1 - Mean_Radius	0.0000
2 - Mean_Texture	0.0452
3 - Mean_Perimeter	0.0000
4 - Mean_Area	0.0000
5 - Mean_Smoothness	0.0000
6 - Mean_Compactness	0.0018
7 - Mean_Concavity	0.0027
8 - Mean_Concave_Points	0.0139
9 - Mean_Symmetry	0.0225
10 - Mean_Fractal_Dimension	0.0000
11 - Radius_SE	0.0000
12 - Texture_SE	0.0000
13 - Perimeter_SE	0.0000
14 - Area_SE	0.0000
15 - Smoothness_SE	0.0000
16 - Compactness_SE	0.0000
17 - Concavity_SE	0.0092
18 - Concave_Points_SE	0.0000
19 - Symmetry_SE	0.0000
20 - Fractal_Dimension_SE	0.0000
21 - Worst_Radius	0.0303
22 - Worst_Texture	0.0000
23 - Worst_Perimeter	0.0000
24 - Worst_Area	0.1975
25 - Worst_Smoothness	0.0000
26 - Worst_Compactness	0.0000
27 - Worst_Concavity	0.0083
28 - Worst_Concave_Points	0.6525
29 - Worst_Symmetry	0.0000
30 - Worst_Fractal_Dimension	0.0162

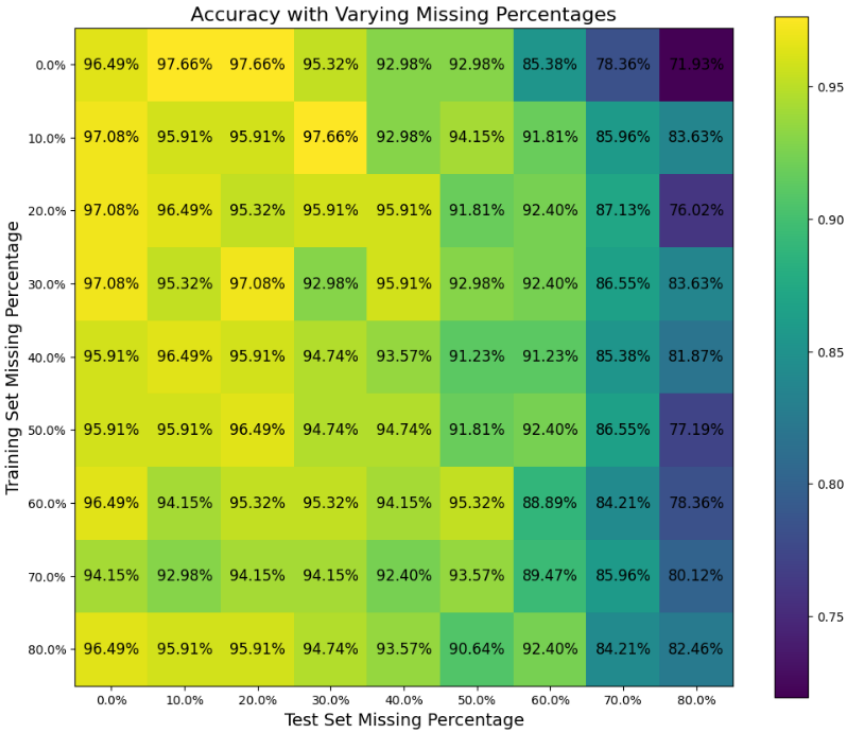


Features	Importance
1 - Mean_Radius	0.0539
2 - Mean_Texture	0.0093
3 - Mean_Perimeter	0.0625
4 - Mean_Area	0.0656
5 - Mean_Smoothness	0.0029
6 - Mean_Compactness	0.0113
7 - Mean_Concavity	0.0340
8 - Mean_Concave_Points	0.1407
9 - Mean_Symmetry	0.0007
10 - Mean_Fractal_Dimension	0.0009
11 - Radius_SE	0.0037
12 - Texture_SE	0.0040
13 - Perimeter_SE	0.0073
14 - Area_SE	0.0526
15 - Smoothness_SE	0.0000
16 - Compactness_SE	0.0008
17 - Concavity_SE	0.0022
18 - Concave_Points_SE	0.0035
19 - Symmetry_SE	0.0014
20 - Fractal_Dimension_SE	0.0006
21 - Worst_Radius	0.0931
22 - Worst_Texture	0.0075
23 - Worst_Perimeter	0.1287
24 - Worst_Area	0.1069
25 - Worst_Smoothness	0.0035
26 - Worst_Compactness	0.0125
27 - Worst_Concavity	0.0171
28 - Worst_Concave_Points	0.1544
29 - Worst_Symmetry	0.0069
30 - Worst_Fractal_Dimension	0.0031

Παρατηρούμε ότι στον tree classifier τα σημαντικά χαρακτηριστικά είναι λίγα (τα πιο σημαντικά χαρακτηριστικά είναι μόνο 2). Στο RF τα σημαντικά χαρακτηριστικά είναι πολύ περισσότερα (περίπου 10), αλλά οι τιμές σημαντικότητας του κάθε χαρακτηριστικού είναι μικρότερες από τις τιμές του tree classifier. Αυτό είναι λογικό, γιατί θέλουμε όλες οι τιμές σημαντικότητας να αθροίζουν στο 1. Οπότε, στο RF δίνεται βαρύτητα σε πολλά χαρακτηριστικά, δηλαδή διερευνώνται πολλές περιοχές λύσεων, σε αντίθεση με τον tree classifier που χρησιμοποιεί μόνο ένα πολύ μικρό υποσύνολο των χαρακτηριστικών. Για αυτό το λόγο, όπως αναφέραμε και στο ερώτημα Γ, ο RF σχεδόν πάντα καταλήγει σε καλύτερη λύση.

Ε. Να εκπαιδεύσετε το ίδιο RF για μεταβλητό ποσοστό missing values (0% έως 80% ανά 10%) και για κάθε forest να υπολογίσετε το classification accuracy εισάγοντας στο test set μεταβλητό ποσοστό missing values (0% έως 80% ανά 10%). Απεικονίστε κατάλληλα τη συνολική συμπεριφορά της ακρίβειας του RF ταξινομητή σε σχέση με το ποσοστό missing values. Σχολιάστε την αντοχή του random forest στα missing data, τόσο στο training όσο και στο testing. Σε ποιο από τα δύο είδη missing data είναι περισσότερο ευαίσθητος;

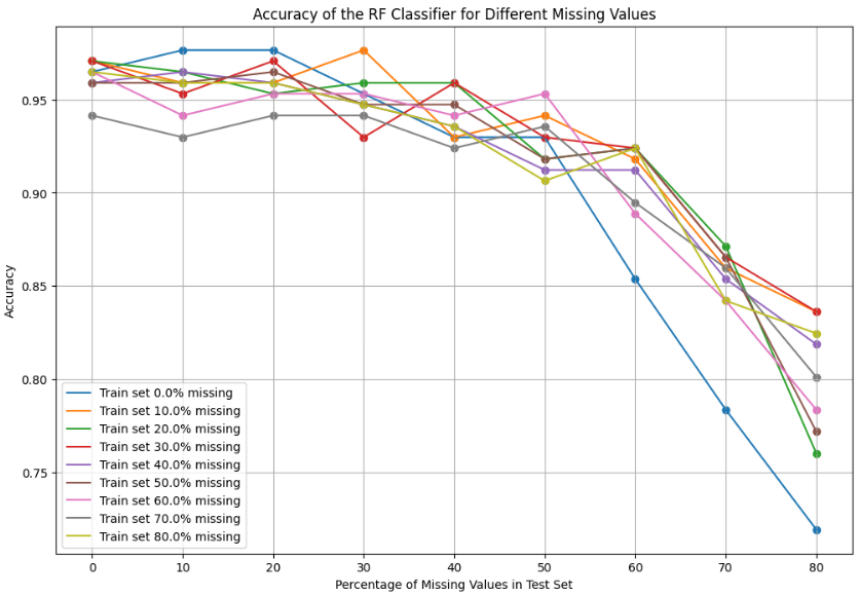
Το RF εκπαιδεύεται και αξιολογείται για όλους του πιθανούς συνδυασμούς missing values σε train και test set (0% έως 80% ανά 10%). Τα αποτελέσματα της ακρίβειας φαίνονται στον παρακάτω πίνακα.



Παρατηρούμε ότι το χαμηλότερο accuracy (71.93%) προκύπτει όταν στο training set δεν υπάρχουν missing values, ενώ στο test set υπάρχει μεγάλος αριθμός missing values (80%). Γενικά, για μεγάλο αριθμό missing values στο test set (πάνω από 50%), η ακρίβεια είναι χαμηλότερη. Για μεγάλο αριθμό missing values στο training set, η ακρίβεια παραμένει σχετικά σταθερή. Το υψηλότερο accuracy (97.66%) προκύπτει όταν στο training set έχω λίγα missing values (0% με 10%) και στο test set έχω μικρό αριθμό missing values (10% με 30%).

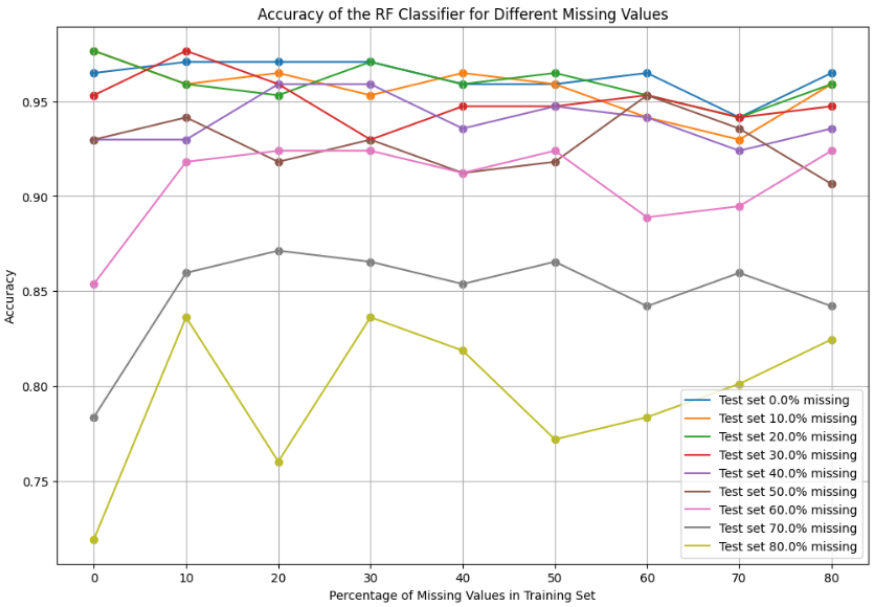
Ο RF είναι περισσότερο ευαίσθητος σε missing values στο test set, καθώς για μεγάλο ποσοστό η ακρίβεια μειώνεται ραγδαία. Ενώ για ίδιο ποσοστό missing values στο test set, καθώς το ποσοστό των missing values μεταβάλλεται στο training set, η ακρίβεια παραμένει σχετικά σταθερή. Εξαίρεση αποτελούν πολύ μεγάλα ποσοστά missing values στο test set, στα οποία αν δεν υπάρχουν missing values στο training set μας δίνουν τις χαμηλότερες τιμές ακρίβειας (πιθανότατα επειδή ο ταξινομητής δεν έχει μάθει να χειρίζεται την επιπλέον τυχαιότητα που εισάγεται).

Παρ' όλα αυτά αν ο αριθμός των missing values είναι σχετικά μικρός κυρίως στο test set (κάτω από 40%), τότε η ακρίβεια του ταξινομητή δεν επηρεάζεται (δεν μειώνεται) και σε ορισμένες ευνοϊκές περιπτώσεις μπορεί και να αυξηθεί.



Στο διάγραμμα, παρατηρούμε πως μεταβάλλεται η ακρίβεια καθώς το ποσοστό missing data στο test set αυξάνεται, για διαφορετικά ποσοστά missing data στο training set. Μέχρι περίπου το 30% missing data στο test set, η ακρίβεια για όλες τις γραμμές παραμένει σχετικά σταθερή και κυμαίνεται σε παρόμοιες τιμές. Για ποσοστά πάνω από 30% missing data στο test set, η ακρίβεια μειώνεται για τις περισσότερες γραμμές και για ποσοστά πάνω από 50% η μείωση είναι ραγδαία.

Σε αυτό το διάγραμμα, μπορούμε να παρατηρήσουμε πως μεταβάλλεται η ακρίβεια καθώς το ποσοστό missing data στο training set αυξάνεται, για διαφορετικά ποσοστά missing data στο test set. Η ακρίβεια για ποσοστά missing data στο test set μέχρι 40% παραμένει σχετικά σταθερή και κυμαίνεται σε παρόμοιες τιμές. Για μεγαλύτερο ποσοστό missing data στο test set η ακρίβεια μειώνεται ραγδαία, αλλά για ενδιάμεσα ποσοστά missing data στο training set παραμένει σχετικά σταθερή.



ΣΤ. Να εκπαιδεύσετε έναν δενδρικό ταξινομητή και έναν RF ταξινομητή όπως στα προηγούμενα, για 10% missing data στο training set. Υπολογίστε τις καμπύλες precision-recall για τους δύο ταξινομητές (χωρίς missing data στο test set). Ποιος ταξινομητής έχει καλύτερη συμπεριφορά και γιατί; Δεδομένου ότι οι ταξινομητές που εκπαιδεύσατε αποτελούν συστήματα πρόβλεψης της κακοήθειας όγκων, ποια χαρακτηριστικά της καμπύλης έχουν περισσότερη βαρύτητα κατά την πρακτική εφαρμογή;

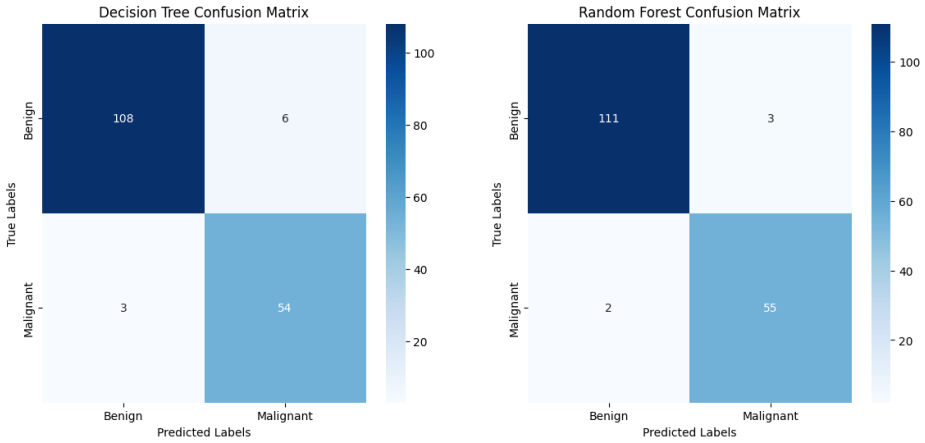
Όπως και πριν τα αποτελέσματα για 10% missing data στο training set για το δενδρικό ταξινομητή είναι:

```
Train with 10.0% missing values. Accuracy of the tree classifier is 94.74%
```

Για το RF είναι:

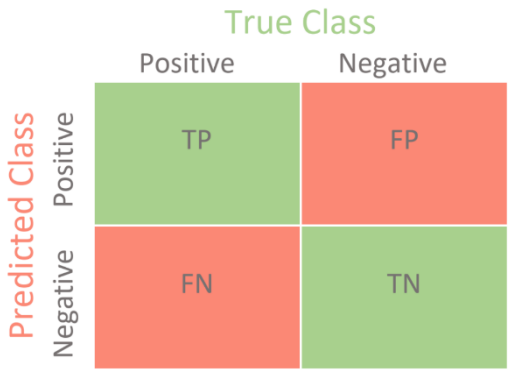
```
Train with 10.0% missing values. Accuracy of the random forest classifier is 97.08%
```

Οι πίνακες σύγχυσης είναι οι ακόλουθοι.



Παρατηρούμε ότι οι λανθασμένες προβλέψεις είναι πολύ λίγες. Ο RF φαίνεται να είναι καλύτερος, γιατί κάνει λιγότερα λάθη.

Γνωρίζουμε ότι ισχύει:

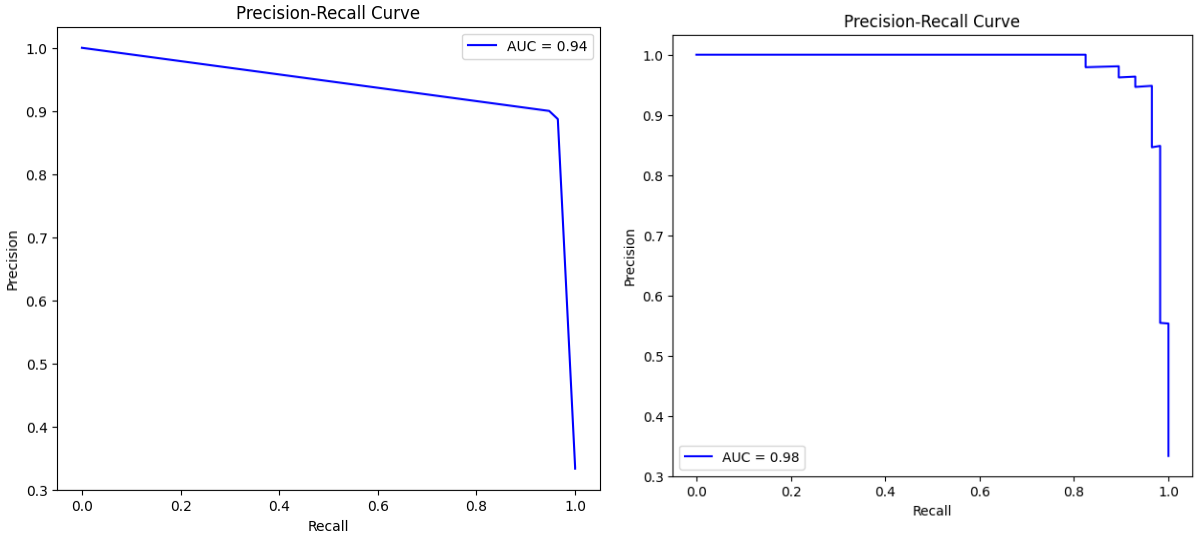


Άρα και για τους δύο ταξινομητές, όσον αφορά τις λανθασμένες προβλέψεις έχουμε περισσότερα FP (False Positive). Δηλαδή, προβλέπουμε ότι υπάρχει μια σοβαρή ασθένεια, ενώ στην πραγματικότητα κάνουμε λάθος.

Τα FN (False Negative) σημαίνουν ότι, ενώ υπάρχει μια σοβαρή ασθένεια, εμείς δεν την προβλέψαμε.

Οι καμπύλες precision-recall για το δενδρικό ταξινομητή φαίνονται παρακάτω. Το AUC είναι η περιοχή που περικλείεται από την κάθε καμπύλη. Όσο μεγαλύτερο, τόσο καλύτερος ο ταξινομητής (λιγότερα λάθη). Οι καμπύλες αναφέρονται στην θετική κλάση, την κλάση 1 (Malignant). Οπότε, εφόσον το RF έχει μεγαλύτερο AUC από τον δενδρικό ταξινομητή, αυτό σημαίνει ότι κάνει λιγότερα λάθη, άρα είναι καλύτερος.

Η καμπύλη precision-recall για το δενδρικό ταξινομητή είναι αυτή στο αριστερό διάγραμμα και για το RF στο δεξί.



Οι ταξινομητές που εκπαιδεύσαμε αποτελούν συστήματα πρόβλεψης της κακοήθειας όγκων που είναι μια πολύ σοβαρή ασθένεια, την οποία αν υπάρχει δεν θέλουμε σε καμία περίπτωση να μην την εντοπίσουμε, διότι μπορεί να τεθεί σε κίνδυνο η ζωή του ασθενούς αν δε δράσουμε άμεσα. Οπότε, σε αυτήν την περίπτωση πρέπει να δοθεί μεγαλύτερη έμφαση στην ελαχιστοποίηση των FN (ενώ υπάρχει η ασθένεια, δεν βρέθηκε). Δεν μας πειράζει να έχουμε περισσότερα FP (να προβλέπουμε την ασθένεια, ενώ στην πραγματικότητα δεν υπάρχει), σε αυτήν την περίπτωση θα γίνουν περισσότερες εξετάσεις και θα φανεί ότι δεν υπάρχει ασθένεια. Επομένως, θέλουμε να έχουμε υψηλό recall, άρα μπορούμε να κατεβάσουμε το confidence threshold, ώστε να προβλέπουμε πιο εύκολα δείγματα στη θετική κλάση (ασθένεια), παρά στην αρνητική. Το πιο σημαντικό στοιχείο του διαγράμματος είναι οι τιμές της καμπύλης που έχουν υψηλό recall, οι τιμές αυτές αναφέρονται σε χαμηλότερο threshold.