ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ
HAROKOPIO UNIVERSITY

Εργασία Μηχανικής Μάθησης

Διαχείριση Δεδομένων II - Μηχανική Μάθηση

Μαρίνος Κουβαράς, ap23011

12 Ιουλίου 2024

Πίνακας Περιεχομένων

Μέρος 1

Η τεκμηρίωση των αποτελεσμάτων βρίσκεται στο επισυναπτόμενο αρχείο *ap23011-Part_1-ML.ipynb.*

1.1 Διερεύνηση των δεδομένων (EDA) και προ-επεξεργασία

Απο την ανάλυση των δεδομένων μας παρατηρούμε πώς αρχικά έχουμε ένα πίνακα δεδομένων 20640X10. Κάθε στήλη αποτελείται από αριθμούς τύπου float ενώ παράλληλα η ονομασία κάθε στήλης φαίνεται παρακάτω.Η τελευταία στήλη έχει σαν δεδομένα αντικείμενο. Επίσης παρατηρούμε πως λείπουν δεδομένα στο total bedrooms καθώς η τιμή συνόλου είναι 20433.

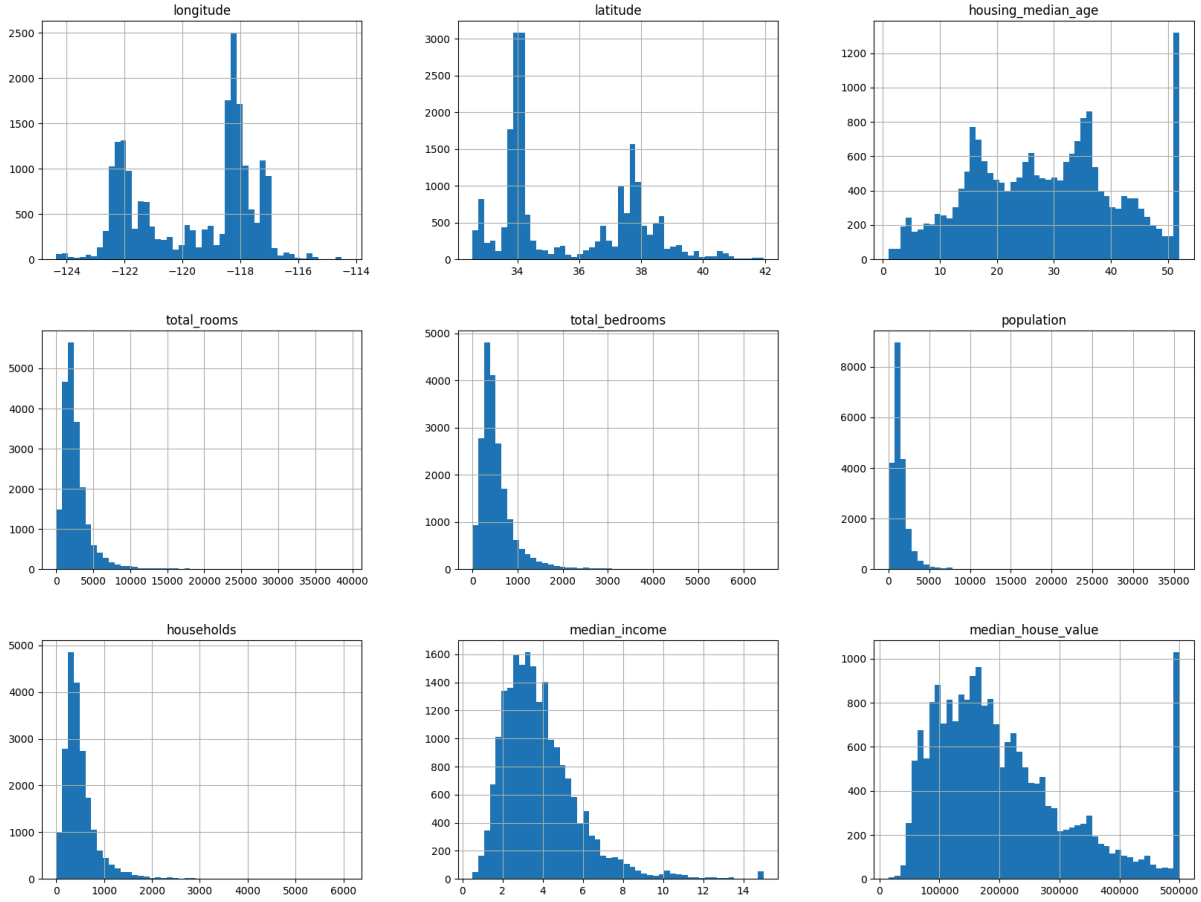
```
#   Column              Non-Null Count  Dtype
---  ------             --------------  -----
 0   longitude          20640 non-null  float64
 1   latitude           20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms        20640 non-null  float64
 4   total_bedrooms     20433 non-null  float64
 5   population         20640 non-null  float64
 6   households         20640 non-null  float64
 7   median_income      20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity    20640 non-null  object
```

Απο την εντολή describe παρατηρούμε ότι:

- Οι τιμές του longitude έχουν μείον (-) σαν πρόσημο το οποίο είναι κατανοητό γιατί αφορά συντεταγμένες.
- Επίσης από το πεδίο count παρατηρούμε ότι έχουμε 207 τιμές που μας λείπουν από το total_bedrooms.
- Επιπλέον παρατηρούμε πως το εύρος των τιμών είναι αρκετά μεγάλο.
- Πολύ σημαντικό είναι πως οι τιμές media_income έχουν κλιμακωθεί σε ένα εύρος από 0.499900 έως 15.000100, ενώ κλιμάκωση φαίνεται να έχουμε και στις τιμές housing_median_age και median_house_value.
- Τέλος αξίζει να παρατηρήσουμε την τιμή στόχο (median_house_value) όπου το κύριο εύρος της είναι μεταξύ 119600 και 264725.

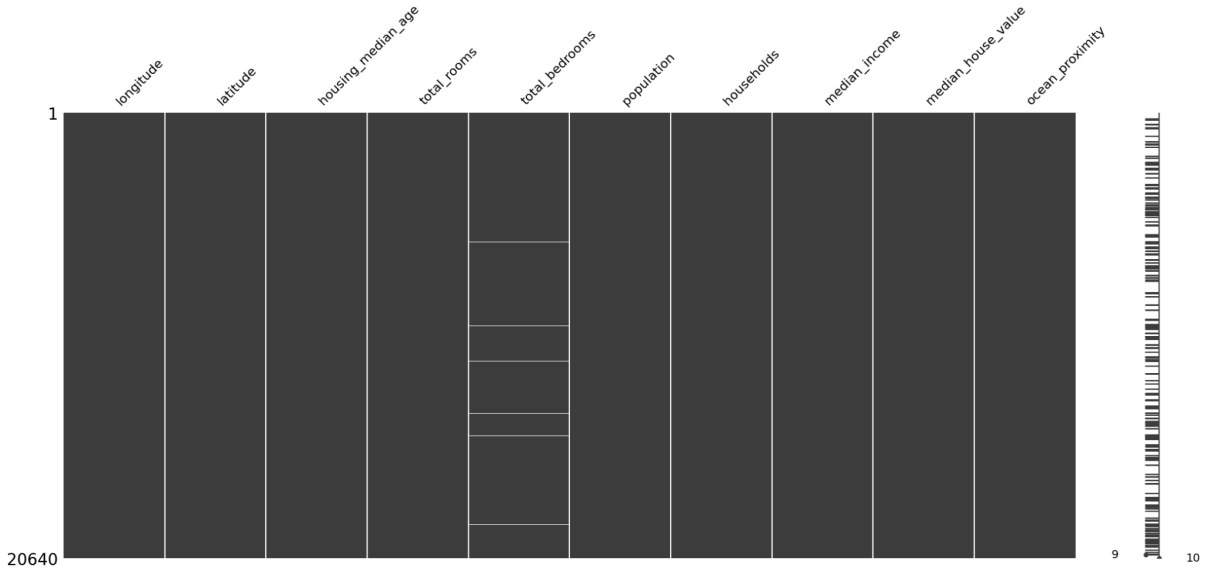
	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	-119.569704	35.631861	28.639486	2635.763081	536.838857	1425.476744	499.539680	3.870671	206855.816909
std	2.003532	2.135952	12.585558	2181.615252	419.391878	1132.462122	382.329753	1.899822	115395.615874
min	-124.350000	32.540000	1.000000	2.000000	1.000000	3.000000	1.000000	0.499900	14999.000000
25%	-121.800000	33.930000	18.000000	1447.750000	297.000000	787.000000	280.000000	2.563400	119600.000000
50%	-118.490000	34.260000	29.000000	2127.000000	435.000000	1166.000000	409.000000	3.534800	179700.000000
75%	-118.010000	37.710000	37.000000	3148.000000	643.250000	1725.000000	605.000000	4.743250	264725.000000
max	-114.310000	41.950000	52.000000	39320.000000	6445.000000	35682.000000	6082.000000	15.000100	500001.000000

- Απο τα ιστογράμματα μπορούμε να παρατηρήσουμε καλύτερα το εύρος των τιμών με βάση αυτό που εκφράζουν.

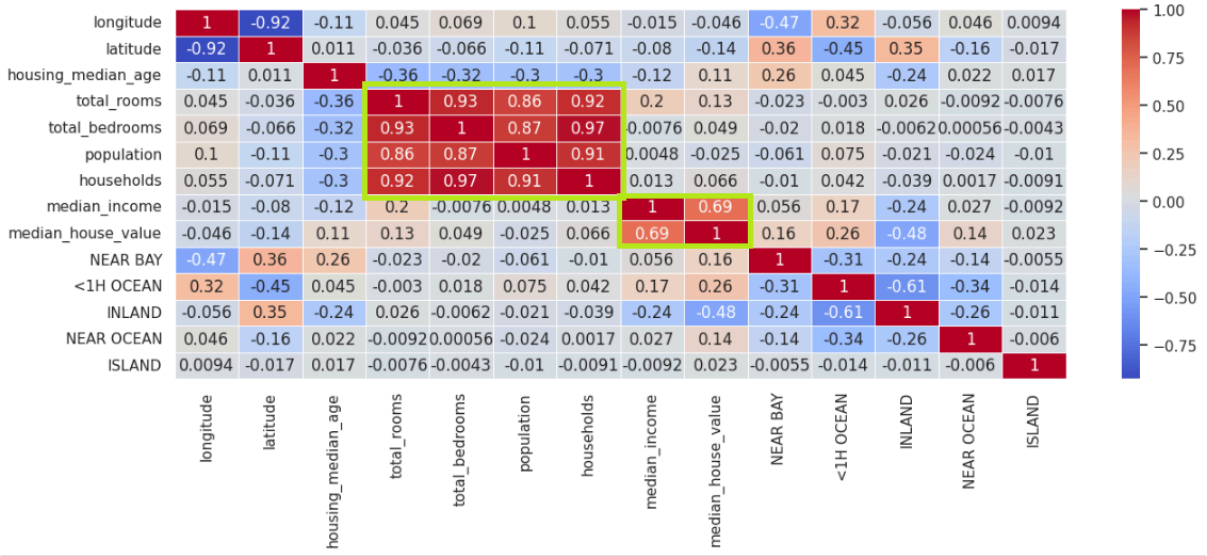


Όπως διαπιστώνεται οι τιμές έχουν ένα μεγάλο εύρος τιμών. Τα περισσότερα ιστογράμματα έχουν “ουρά” που εκτείνεται προς τα δεξιά σε σχέση με τη μέση τιμή.

Όσον αφορά τις τιμές που λείπουν μπορούμε να οπτικοποιήσουμε πού εντοπίζονται οι τιμές που αυτές στο παρακάτω διάγραμμα.



Απο το heatmap μπορούμε να παρατηρήσουμε τις συσχετίσεις των τιμών.



Παρατηρούμε υψηλή συσχέτιση στα δύο ορθογώνια που επισημαίνονται με πράσινο χρώμα. Εξεταζοντας το μεγαλύτερο βλέπουμε πως τα χαρακτηριστικά ενός σπιτιού αλληλοεξαρτώνται όπως είναι λογικό αλλά και συνδέονται με το population, δηλαδή αύξηση ή μείωση του πληθυσμού επηρεάζει αντίστοιχα και τον αριθμό του συνόλου των χώρων ενός σπιτιού.

Εξετάζοντας το μικρότερο παρατηρούμε μερική συσχέτιση μεταξύ το μέσου εισοδήματος και της μέσης αξίας του σπιτιού. Τέλος παρατηρούμε πως στην επάνω αριστερή γωνία υπάρχει ισχυρή συσχέτιση μεταξύ του γεωγραφικού πλάτους και μήκους, και αφορά κυρίως την ιδιαιτερότητα της σχέσης που έχουν τα γεωγραφικά δεδομένα. Το μπλέ χρώμα είναι φυσιολογικό καθώς όπως αναφέρθηκε οι τιμές του longitude είναι αρνητικές.

1.4 Ανάπτυξη γραμμικού μοντέλου

Ερώτηση: Τι συμπέρασμα βγάζετε όσον αφορά την τυποποίηση των δεδομένων;

Επιχειρούμε να εκτελέσουμε δύο παράλληλες διαδικασίες εκπαίδευσης, η πρώτη αφορά τα δεδομένα μας χωρίς κάποια τυποποίηση και η δεύτερη με τυποποίηση.

Model score for not scaled: 0.6470480227253683
Model score for scaled: 0.6470480227253683
<---------->
MSE not scaled 4733529273.092575
MSE scaled 4733529273.092554
<---------->
MAE not scaled 50078.09884156549
MAE scaled 50078.09884156545

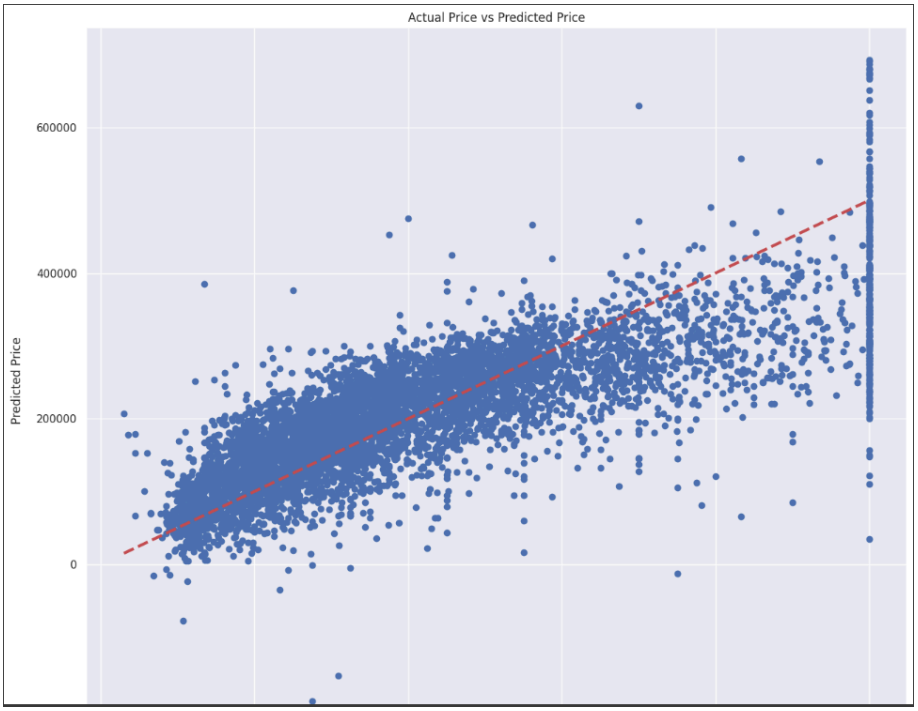
Η τυποποίηση των δεδομένων διαμορφώνει με τέτοιο τρόπο τις τιμές ώστε το μοντέλο να αποδίδει καλύτερα. Πιο συγκεκριμένα βελτιώνουν την απόδοση του μοντέλου, περιορίζουν τις ακραίες τιμές (outliers), βοηθούν το μοντέλο στην καλύτερη κατανόηση της σχέσης των δεδομένων και τη διαμόρφωση των coefficient και περιορίζουν σημαντικά το overfitting.

Απο την ανάλυση παρατηρούμε πως η απόδοση παραμένει η ίδια. Μία εξήγηση θα μπορούσε να είναι πως τα δεδομένα μας, όπως είδαμε και παραπάνω έχουν δεχτεί μία κλιμάκωση. Επίσης από την ανάγνωση της βιβλιογραφίας θα μπορούσαμε να πούμε πως το μοντέλο linear_model.LinearRegression() μπορεί να προσαρμόζεται εύκολα στα δεδομένα, τα οποία όπως είδαμε απο το heatmap σε αρκετά σημεία έχουν υψηλή συσχέτιση. Για επιβεβαίωση και τεκμηρίωση μπορούμε να εξετάσουμε το παρακάτω αποτέλεσμα:

Cross-validated MSE for not scaled: 4763194358.520627
Cross-validated MSE for scaled: 4763396853.845527

Σε αυτή την περίπτωση οι αριθμοί δεν είναι ίδιοι και έχουμε μία ισχυρή ένδειξη πως το μοντέλο προσαρμόστηκε και στα μη τυποποιημένα δεδομένα. Ωστόσο η τυποποίηση είναι μία διαδικασία που θα πρεπει να εφαρμόζεται.

Με τυποποιημένα δεδομένα η τιμή score (R^2 prediction) του μοντέλου είναι 0.64. Αν θέλουμε να εξηγήσουμε την απόδοση αυτή, μία “προβλεψη” για την τιμή στόχο (median_house_value) μπορεί να φτάνει σε απόκλιση έως και 68,376. Παρακάτω φαίνεται η αξιολόγηση του μοντέλου ενώ στο αρχείο τεκμηρίωσης περιλαμβάνονται διαγράμματα απεικόνισης για κάθε μεταβλητή.



1.5 Παλινδρόμηση με Random Forest

Ερώτηση: Τι παρατηρείτε σχετικά με την επίδοση του Random Forest σε σύγκριση με τα προηγούμενο μοντέλο; Πώς εξηγείτε τη διαφορά στην επίδοση;

Απο την εκτέλεση του σχετικού κώδικα έχουμε τις τιμές:

Score 0.8175530027602743
Train MSE: 340352066.7029687
Test MSE: 2394690014.2734513
Train MAE: 11990.939782668884
Test MAE: 31879.238661175714

Η τιμή Score είναι υψηλότερη σε σύγκριση με το γραμμικό μοντέλο ενώ οι τιμές MSE και MAE σε όλες τις περιπτώσεις είναι χαμηλότερες για το μοντέλο Random Forest σε σύγκριση με το γραμμικό μοντέλο. Αυτό συμβαίνει επειδή το Random Forest είναι ένα πιο σύνθετο μοντέλο από το γραμμικό. Το Random Forest αντιμετωπίζει καλύτερα δεδομένα με πολυπλοκότητα όπως είναι το δικό μας dataset. Φαίνεται πως εξάγει πιο ακριβείς προβλέψεις, ειδικά σε προβλήματα που η σχέση μεταξύ των χαρακτηριστικών και της μεταβλητής εξαρτημένης δεν είναι σε όλες τις περιπτώσεις γραμμική. Γενικά, το Random Forest έχει καλύτερη επίδοση σε αυτά τα δεδομένα σε σύγκριση με το απλό γραμμικό μοντέλο.

Μέρος 2

Η τεκμηρίωση των αποτελεσμάτων βρίσκεται στο επισυναπτόμενο αρχείο *ap23011-Part_2-ML.ipynb.*

Πως προκύπτει ο αριθμός των παραμέτρων του κάθε επιπέδου.

Παρατηρούμε τους αριθμούς των παραμέτρων του διασυνδεδεμένου νευρωνικού δικτύου:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_6 (Dense)             (None, 20)                61460

 dense_7 (Dense)             (None, 20)                420

 dense_8 (Dense)             (None, 10)                210

=================================================================
Total params: 62090 (242.54 KB)
Trainable params: 62090 (242.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Στο πρώτο επίπεδο έχουμε για είσοδο εικόνα 32*32 με 3 κανάλια (χρώματα) επομένως 3072 εισαγόμενα δεδομένα με 20 νευρώνες επομένως 3072*20 = 61440 και προσθέτουμε και τα Bias για κάθε νευρώνα 61440+20 = 61460.

Στο δεύτερο επίπεδο έχουμε 20 νευρώνες εισόδου από το πρώτο επίπεδο που τα δεδομένα τους εισέρχονται σε 20 νευρώνες του δεύτερου επιπέδου επομένως 20*20 = 400 και προσθέτουμε και τα Bias για κάθε νευρώνα 400+20 = 420.

Στο τελευταίο επίπεδο έχουμε 20 νευρώνες από το δεύτερο επίπεδο που τα δεδομένα τους εισέρχονται σε 10 νευρώνες του τρίτου επιπέδου επομένως 20*10=200 και προσθέτουμε και τα Bias για κάθε νευρώνα 200+10 = 210.

Πως προκύπτει ο αριθμός των παραμέτρων του κάθε επιπέδου.

Παρατηρούμε τους αριθμούς των παραμέτρων του συνελικτικού νευρωνικού δικτύου:

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 16)	448
max_pooling2d (MaxPooling2D)	(None, 15, 15, 16)	0
conv2d_1 (Conv2D)	(None, 13, 13, 32)	4640
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten (Flatten)	(None, 1152)	0
dense_3 (Dense)	(None, 10)	11530

Total params: 16618 (64.91 KB)
Trainable params: 16618 (64.91 KB)
Non-trainable params: 0 (0.00 Byte)

Στο πρώτο επίπεδο έχουμε 3 κανάλια από την είσοδο και 16 φίλτρα με το μέγεθος του kernel να είναι 3*3, επομένως 3*16*3*3=432 και επειδή έχουμε και 16 bias τότε 432+16=448.

Στο δεύτερο επίπεδο έχουμε 16 κανάλια εισόδου από το πρώτο επίπεδο και 32 φίλτρα με το μέγεθος του kernel να είναι 3*3, επομένως 16*32*3*3=4608 και επειδή έχουμε και 32 bias τότε 4608+32=4640.

Στο τρίτο επίπεδο έχουμε 1152 κανάλια εισόδου από το δεύτερο επίπεδο που πηγαίνουν σε 10 νευρώνες επομένως 1152*10=11520 και επειδή έχουμε και 10 bias 11520+10=11530.

Τι παρατηρείτε σχετικά με τον αριθμό παραμέτρων και την επίδοση του κάθε δικτύου; Πως εξηγείτε τις παρατηρήσεις σας;

Όσον αφορά των αριθμό των παραμέτρων τα συνελικτικά δίκτυα έχουν σαφώς περισσότερες παραμέτρους. Απο τη μελέτη της βιβλιογραφίας τα συνελικτικά δίκτυα είναι καλύτερη επιλογή για δεδομένα εικόνων όπως είναι τα δικά μας στο συγκεκριμένο παράδειγμα καθώς με τη χρήση της συνέλιξης εντοπίζουν με μεγαλύτερη ακρίβεια χαρακτηριστικά στις εικόνες. Αντίθετα τα νευρωνικά δίκτυα αποδίδουν καλύτερα σε δεδομένα που έχουν συνέχεια.

Ποιες παράμετροι φαίνεται να επηρεάζουν τα αποτελέσματα; Τι παρατηρείτε όσο αυξάνεται η συνθετότητα ενός μοντέλου;

Για να απαντήσουμε σε αυτό το ερώτημα θα πρέπει παράλληλα να πειραματιστούμε με τις αρχιτεκτονικές του δικτύου, όπως αναφέρεται και σε σχετικό ερώτημα.

Όσον αφορά το νευρωνικό δίκτυο τα αποτελέσματα των δοκιμών που πραγματοποιήθηκαν είναι τα εξής:

Πιο “βαθύ” δίκτυο

```
loss, accuracy = deep_model.evaluate(x_test_fc, y_test_fc)
313/313 [==============================] - 1s 2ms/step - loss: 2.1759 - accuracy: 0.2294
```

Πιο “πλατύ” δίκτυο

```
loss, accuracy = wide_model.evaluate(x_test_fc, y_test_fc)
313/313 [==============================] - 1s 4ms/step - loss: 1.7621 - accuracy: 0.3880
```

Χρήση Adam optimizer

```
[48] loss, accuracy = adam_model.evaluate(x_test_fc, y_test_fc)
313/313 [==============================] - 1s 2ms/step - loss: 1.8896 - accuracy: 0.2719
```

Χρήση dropout

```
loss, accuracy = dropout_model.evaluate(x_test_fc, y_test_fc)
313/313 [==============================] - 1s 2ms/step - loss: 2.1707 - accuracy: 0.2393
```

Φυσικά δοκιμάζονται στοχευμένες αλλαγές για λόγους παρουσίασης. Η αλλαγές στο μοντέλο θα πρέπει να είναι συνδυαστικές. Έπειτα από δοκιμές ένα μοντέλο που φαίνεται να αποδίδει είναι το εξής:

```
# Define a better model
better_model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(x_train_fc.shape[1],)),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(10, activation='softmax')
])

# Compile the deeper model
better_model.compile(
    optimizer=tf.keras.optimizers.SGD(),
    loss=CategoricalCrossentropy(),
    metrics=['accuracy']
)

# Train the deeper model
better_model.fit(x_train_fc, y_train_fc, epochs=20, batch_size=50, validation_data=(x_test_fc, y_test_fc),callbacks=[my_scheduler])
```

```
loss, accuracy = better_model.evaluate(x_test_fc, y_test_fc)
313/313 [==============================] - 1s 4ms/step - loss: 1.6699 - accuracy: 0.4128
```

Αυτό που μπορούμε να συμπεράνουμε είναι πως:

- Το βάθος και το πλάτος βοηθούν στη βελτίωση της απόδοσης του μοντέλου, έως ένα σημείο διαφορετικά οδηγούμαστε σε overfitting.

- Το learning rate παίζει πολύ σημαντικό ρόλο και για τις δοκιμές το βάλαμε μεταβαλλόμενο.
- Η αλλαγή του optimizer έχει σημαντική επίπτωση με τον Adam να μας επιστρέφει πιο γρήγορα αποτελέσματα αλλα με τον SDG να μπορεί να πετύχει καλύτερα αποτελέσματα.
- Τέλος η χρήση του dropout μπορεί να βελτιώσει μία κατάσταση overfitting

Όσον αφορά το συνελικτικό δίκτυο με τη μέθοδο των δοκιμών και αλλάζοντας διάφορες παραμέτρους μπορούμε να καταλήξουμε στο εξής μοντέλο:

```
model_cnn_test = tf.keras.models.Sequential([
  tf.keras.layers.Conv2D(64, (3, 3), activation='relu', input_shape=(32, 32, 3)),
   tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
   tf.keras.layers.MaxPooling2D((2, 2)),
   tf.keras.layers.Dropout(0.25),

   tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
   tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
   tf.keras.layers.MaxPooling2D((2, 2)),
   tf.keras.layers.Dropout(0.25),

   tf.keras.layers.Flatten(),
   tf.keras.layers.Dense(512, activation='relu'),
   tf.keras.layers.Dropout(0.5),
   tf.keras.layers.Dense(10, activation='softmax')
])
```

Με την επίδοσή του να είναι:
loss: 0.6207 - accuracy: 0.8007

Απο τις διάφορες μεταβολες των παραμέτρων παρατηρήσαμε πως όσο πιο σύνθετο γίνεται ένα μοντέλο η απόδοση του βελτιώνεται. Αυτό οφείλεται στο γεγονός πως περισσότερα χαρακτηριστικά εξάγονται και έτσι το μοντέλο μας εκπαιδεύεται καλύτερα. Ωστόσο από ένα σημείο και έπειτα, περαιτέρω σύνθεση του μπορεί να οδηγήσει σε overfitting. Η εισαγωγή του dropout μπορεί να μας βοηθήσει να μειώσουμε αυτή την περίπτωση. Επιπλέον βλέπουμε πως όσο πιο σύνθετο το μοντέλο τόσο μεγαλύτερη υπολογιστική ισχύς απαιτείται.

Η επιλογή του learning_rate επηρεάζει την απόδοση του μοντέλου ενώ μεταβάλλει και το χρόνο που απαιτείται για την εκπαίδευσή του. Δοκιμές έγιναν για διάφορες τιμές όπως π.χ. 0.01 και το μοντέλο έφτασε σε accuracy≈0.4 χωρίς να βελτιώνεται. Μία τιμή της τάξης 0.001 είναι καλή.

Η τιμή του batch_size επίσης έχει μεγάλη επίδραση στον χρόνο και την υπολογιστική ισχύ. Αύξηση του batch size μπορούν να μειώσουν το χρόνο εκτέλεσης αλλά οδηγούν σε χειρότερη γενίκευση. Δοκιμές έγιναν για διάφορες τιμές όπως π.χ. 24 και το μοντέλο έφτασε σε accuracy≈0.56. Μία τιμή της τάξης 64 είναι καλή.

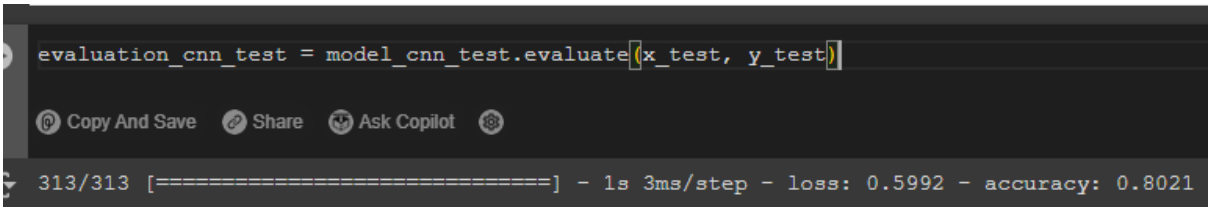
Η τιμή του epoch βοηθάει στη βελτίωση της απόδοσης του μοντέλου και όλες οι δοκιμές έγιναν για τιμή 30. Σίγουρα μεγαλύτερος αριθμός εποχών θα είχε θετική επίδραση, ωστόσο

στα πλαίσια αρκετών δοκιμών περιοριστηκαμε απο τη δυνατότητα χρήσης της gpu του colab (πεπερασμένη διάθεση πόρων).

Η επιλογή του optimizer επίσης μπορεί να διαφοροποιήσει τα αποτελέσματά μας. Εφόσον πειραματιστήκαμε με διάφορες αλλαγές τελευταία αφήσαμε τον optimizer. Παρακάτω φαίνονται τα αποτελέσματα από τη χρήση του Adam και του SGD αντίστοιχα.

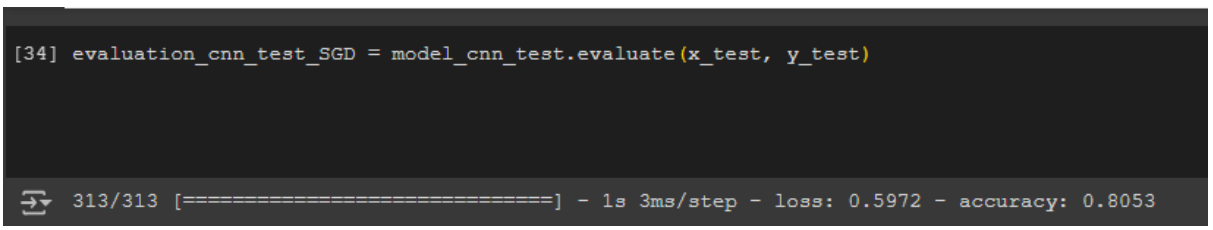
```
model_cnn_test.compile(optimizer=tf.keras.optimizers.Adam(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

###### TRAIN MODEL HERE ( .fit() ) ######
history_cnn_test = model_cnn_test.fit(x_train, y_train, epochs=30,
batch_size=64,             validation_data=(x_test,             y_test),
callbacks=[my_scheduler])
```




```
model_cnn_test.compile(optimizer=tf.keras.optimizers.SGD(),
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

###### TRAIN MODEL HERE ( .fit() ) ######
history_cnn_test_SGD = model_cnn_test.fit(x_train, y_train, epochs=30,
batch_size=64,    callbacks=[my_scheduler],    validation_data=(x_test,
y_test))
```



Μέρος 3

Η τεκμηρίωση των αποτελεσμάτων βρίσκεται στο επισυναπτόμενο αρχείο *ap23011-Part_3-ML.ipynb*

Αρχικά κάνουμε ανάγνωση των δεδομένων μας και αντικαθιστούμε τον ειδικό χαρακτήρα “?”. Τα δεδομένα μας είναι ένας πίνακας 961X6.

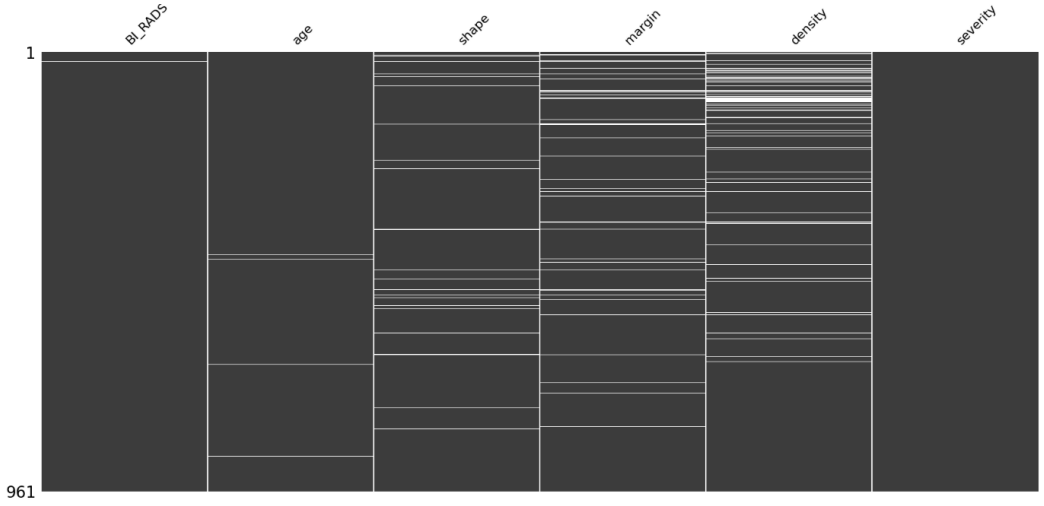
Απο τον πίνακα δεδομένων παρατηρούμε τις τιμές και τις κατηγορίες των δεδομένων μας.

	BI_RADS	age	shape	margin	density	severity
count	959.000000	956.000000	930.000000	913.000000	885.000000	961.000000
mean	4.348279	55.487448	2.721505	2.796276	2.910734	0.463059
std	1.783031	14.480131	1.242792	1.566546	0.380444	0.498893
min	0.000000	18.000000	1.000000	1.000000	1.000000	0.000000
25%	4.000000	45.000000	2.000000	1.000000	3.000000	0.000000
50%	4.000000	57.000000	3.000000	3.000000	3.000000	0.000000
75%	5.000000	66.000000	4.000000	4.000000	3.000000	1.000000
max	55.000000	96.000000	4.000000	5.000000	4.000000	1.000000

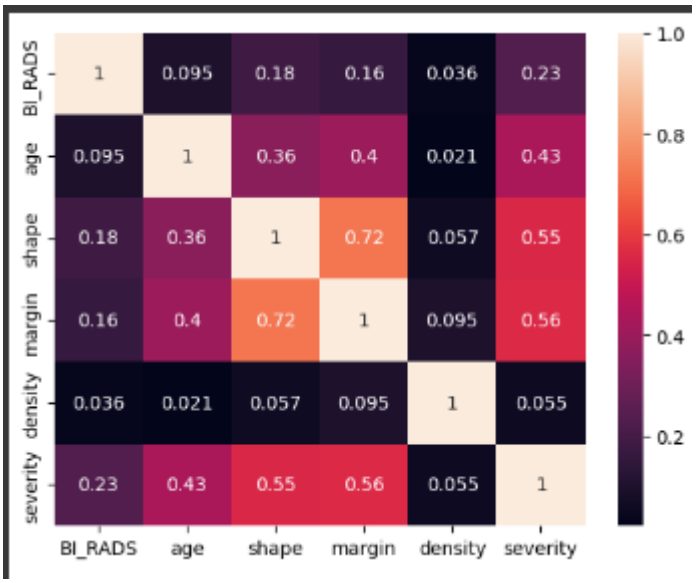
Πιο συγκεκριμένα:
1.BI-RADS assessment: 1 to 5 (ordinal)
2.Age: patient's age in years (integer)
3.Shape: mass shape: round=1 oval=2 lobular=3 irregular=4 (nominal)
4.Margin: mass margin: circumscribed=1 microlobulated=2 obscured=3 ill-defined=4 spiculated=5 (nominal)
5.Density: mass density high=1 iso=2 low=3 fat-containing=4 (ordinal)
6.Severity: benign=0 or malignant=1 (binominal)

Η κατηγορία στόχος είναι το Severity. Η κατανομή των δεδομένων είναι της κατηγορίας είναι 516 καλοήθη αποτελέσματα και 445 κακοήθη.

Παρατηρούμε πως μερικά δεδομένα λείπουν και θα πρέπει να αντικατασταθούν κατάλληλα.



Απο το heatmap παρατηρούμε μία συσχέτιση μεταξύ shape και margin



Για τα συγκεκριμένα δεδομένα θα δοκιμάσουμε τα μοντέλα όπως φαίνεται παρακάτω:

Γραμμικής κατηγοριοποίησης

```
[23] # Two different ways to compute accuracy
     acc = metrics.accuracy_score(y_test, y_hat_test)
     acc2 = np.mean(y_test == y_hat_test)
     print('Accuracy = {}'.format(acc))
     print('Accuracy (alternative) = {}'.format(acc2))

     Accuracy = 0.8477508650519031
     Accuracy (alternative) = 0.8477508650519031
```

Random Forest

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, yhat_rf))

Accuracy: 0.8166089965397924
```

Sequential model

```
loss_and_metrics = model.evaluate(X_test_sc, y_test)
print(loss_and_metrics)
print('Loss = ',loss_and_metrics[0])
print('Accuracy = ',loss_and_metrics[1])

10/10 [==============================] - 0s 5ms/step - loss: 0.3985 - accuracy: 0.8339
[0.39850202202796936, 0.8339100480079651]
Loss =  0.39850202202796936
Accuracy =  0.8339100480079651
```

Ερώτηση: Ποιες παράμετροι φαίνεται να επηρεάζουν τα αποτελέσματα; Τι παρατηρείτε όσο αυξάνεται η συνθετότητα ενός μοντέλου;

Γενικά παρατηρούμε σχετικά καλές επιδόσεις με καλύτερη να είναι αυτή της γραμμικής κατηγοριοποίησης. Απο διάφορες δοκιμές παρατηρούμε πως τα μοντέλα όσο πιο σύνθετα τόσο καλύτερα αποδίδουν, ωστόσο η πολλή σύνθεση μπορεί να τα οδηγήσει σε overfitting. Το βάθος και το πλάτος επηρεάζουν θετικά το μοντέλο όσο αυξάνονται, το learning rate μπορεί να βελτιώσει την απόδοση του μοντέλου αλλα θα πρέπει να αποφεύγονται πολύ μεγάλες ή μικρές τιμές. Ο Adam φαίνεται να μας επιστρέφει πιο γρήγορα αποτελέσματα ειδικά για μικρά dataset οπως αυτό. Η τιμή του batch size έχει επίδραση στη γενίκευση του μοντέλου ενώ η εποχές επίσης μπορούν να βελτιώσουν την απόδοση. Η εισαγωγή validation είναι μια τεχνική που επίσης βοηθάει το μοντέλο.