Neural Networks

&

Deep Learning

ενδιαμεση εργασια

Κωνσταντίνος Μυλωνάς | 10027 | 13.11.2022

Table of Contents

[Εισαγωγή 3](#_Toc119149093)

[Τεχνικές Προδιαγραφές 3](#_Toc119149094)

[Υλοποίηση 4](#_Toc119149095)

[KNN, NCC 4](#_Toc119149096)

[KNN Cross Validation 4](#_Toc119149097)

[Άλλες τεχνικές βελτιστοποίησης της ακρίβειας 4](#_Toc119149098)

[Γραφική αναπαράσταση μετρήσεων 5](#_Toc119149099)

[Σχολιασμός Αποτελεσμάτων 8](#_Toc119149100)

[KNN vs NCC Accuracy 8](#_Toc119149101)

[CROSS VALIDATION VS KNN ACCURACY 8](#_Toc119149102)

[Παράρτημα 9](#_Toc119149103)

[(Α) Πίνακες Μετρήσεων 9](#_Toc119149104)

[(Β) Κωδικας 13](#_Toc119149105)

[KNN & NCC 13](#_Toc119149106)

[KNN Cross Validation 14](#_Toc119149107)

# Εισαγωγή

Στην συγκεκριμένη εργασία ζητήθηκε η υλοποίηση ενός προγράμματος σε οποιαδήποτε γλώσσα προγραμματισμού, το οποίο να συγκρίνει την απόδοση του κατηγοριοποιητή πλησιέστερου γείτονα με 1 και 3 γείτονες (Κ-Nearest Neighbor Classifier ή KNN) με αυτή του κατηγοριοποιητή πλησιέστερου κέντρου (Nearest Class Centroid Classifier ή NCC). Το πρόγραμμα πρέπει να διαβάζει τα δεδομένα εκπαίδευσης (train) και ελέγχου (test) και να μετράει τις αποδόσεις.

Ο κατηγοριοποιητής ΚΝΝ υπολογίζει τα πλησιέστερα Κ σημεία και κατηγοριοποιεί το στοιχείο που μελετάμε με βάση τη συχνότητα εμφάνισης των κατηγοριών αυτών των Κ σημείων. Είναι σαφές πως για να μην υπάρχει ισοψηφία, επιλέγουμε την παράμετρο Κ να είναι περιττός αριθμός.

Ο κατηγοριοποιητής NCC αρχικά διαχωρίζει σε κλάσεις τα πιθανά αποτελέσματα (10 στη συγκεκριμένη περίπτωση). Στην συνέχεια υπολογίζει το κέντρο (centroid) της κάθε κλάσης με τη χρήση του μέσου όρου. Τελικά, συγκρίνει την απόσταση του σημείου που μελετάμε από αυτά τα κέντρα, και το κατηγοριοποιεί το σημείο στην κλάση που η απόσταση αυτή είναι η ελάχιστη.

# Τεχνικές Προδιαγραφές

Για την υλοποίηση χρησιμοποιήθηκε Python ενώ ως περιβάλλον εργασίας το PyCharm.

Ως βάση δεδομένων επιλέχθηκε η MNIST[[1]](#footnote-1) dataset, η οποία περιλαμβάνεται στο Keras[[2]](#footnote-2). Η MNIST αποτελεί μια συλλογή από χειρόγραφα ψηφία, τα οποία χρησιμοποιούνται για την εκπαίδευση αλγορίθμων επεξεργασίας εικόνας. Το Keras είναι ένα API βαθιάς μάθησης γραμμένο σε Python, που τρέχει πάνω στην πλατφόρμα μηχανικής μάθησης TensorFlow.

Για τους κατηγοριοποιητές (KNN, NCC) χρησιμοποιήθηκε η βιβλιοθήκη μηχανικής μάθησης scikit-learn ή sklearn[[3]](#footnote-3).

Σαφώς χρησιμοποιήθηκαν και Python modules όπως NumPy και Time.

# Υλοποίηση

## KNN, NCC

Αρχικά εισάγουμε τις απαραίτητες βιβλιοθήκες, ώστε να μπορούμε να χρησιμοποιήσουμε τη βάση δεδομένων, διάφορες μεθόδους και τους κατηγοριοποιητές. Εναλλάσσοντας παραμέτρους στον αλγόριθμο, όπως τον τύπο της απόστασης σημείων ή τον αριθμό των γειτόνων στον ΚΝΝ, παίρνουμε μετρήσεις σχετικά με το ποσοστό επιτυχίας και το χρόνο εκπαίδευσης και πρόβλεψης. Στη συγκεκριμένη εργασία λήφθηκαν μετρήσεις από 1 έως 25 γείτονες στον ΚΝΝ και Ευκλείδεια και Manhattan απόσταση στον NCC (για σκοπούς απλής παρατήρησης). Τελικά, συγκρίνουμε τόσο τον χρόνο όσο και την ακρίβεια κάθε αλγορίθμου και καταλήγουμε σε συμπεράσματα.

## KNN Cross Validation

Η βάση δεδομένων περιλαμβάνει 2 σετ, 1 για εκπαίδευση (train set) και 1 για testing (test set). Το train set περιλαμβάνει 50,000 στοιχεία, ενώ το test set περιλαμβάνει 10,000 στοιχεία. Μελετώντας τον ΚΝΝ περαιτέρω, βλέπουμε πως μπορεί να εμφανιστεί το φαινόμενο overfitting, δηλαδή ο αλγόριθμος να μαθαίνει πολύ εύκολα το dataset που του δίνουμε και να προσαρμόζεται πλήρως σε αυτό, χωρίς να μπορεί να διαχειριστεί διαφορετικά datasets. Επομένως, χρησιμοποιήθηκε και η τεχνική k-fold Cross Validation (CV) για πιο ακριβή αποτελέσματα. Στην CV, ενώνουμε τα train & test sets σε ένα set 60,000 εικόνων. Στη συνέχεια χωρίζουμε σε k folds το συνολικό dataset και χρησιμοποιούμε 1 fold ως test set και τα υπόλοιπα k-1 folds ως train sets. Με αυτόν τον τρόπο εξασφαλίζουμε πως κάθε στοιχείο του συνολικού set των 60,000 στοιχείων θα έχει χρησιμοποιηθεί και για εκπαίδευση και για testing και συνεπώς πως ο αλγόριθμός μας μπορεί να διαχειρίζεται και διαφορετικά datasets.

Στη συγκεκριμένη εργασία χρησιμοποιούμε τον ΚΝΝ με 3 γείτονες για να πάρουμε μετρήσεις. Το πρόγραμμα που υλοποιείται παίρνει μετρήσεις για 2-39 folds.

## Άλλες τεχνικές βελτιστοποίησης της ακρίβειας

Ύστερα από έρευνα γύρω από την απόδοση του ΚΝΝ, βλέπουμε πως υπάρχουν και άλλες τεχνικές αύξησης της ακρίβειας του κατηγοριοποιητή. Εκτός από την Cross Validation, που αφορά στην υλοποίηση της κατηγοριοποίησης, μπορούμε να επέμβουμε και στο ίδιο το dataset και να το επεξεργαστούμε πριν το χρησιμοποιήσουμε στον αλγόριθμό μας. Μπορούμε να απομακρύνουμε το θόρυβο χρησιμοποιώντας denoising autoencoders, ή να κάνουμε blur (defocus, linear horizontal motion blur). Ωστόσο αυτές οι τεχνικές απλώς ερευνήθηκαν, χωρίς να υλοποιηθούν στη συγκεκριμένη εργασία.

# Γραφική αναπαράσταση μετρήσεων

Figure 1: Predict time of each model

Figure 2: Fit time of each model

Figure 3: Total time of each model

Figure 4: Accuracy percentage of KNN models

Figure 5: Accuracy percentage of each model

Figure 6: Accuracy percentage of KNN (3 neighbors) with Cross Validation

Σχολιασμός Αποτελεσμάτων

## KNN vs NCC Accuracy

Συγκρίνοντας τα αποτελέσματα μεταξύ των KNN με διαφορετικό αριθμό γειτόνων, παρατηρούμε στο γράφημα 4 πως πιο ακριβής είναι αυτός με τους 3 γείτονες (97,05%). Ο συνολικός χρόνος κυμαίνεται από 31 μέχρι 37 δευτερόλεπτα, επομένως θεωρείται γενικά αργός αλγόριθμος. Να σημειωθεί πως ο χρόνος ποικίλλει ανάλογα με την επεξεργαστική ισχύ του μηχανήματος που χρησιμοποιούμε. Ο ίδιος κώδικας έτρεξε και σε διαφορετικό μηχάνημα και σε διαφορετικό περιβάλλον, και αυτό είχε σημαντική επιρροή στο χρόνο εκπαίδευσης και πρόβλεψης. Ωστόσο, όλες οι μετρήσεις που φαίνονται στους πίνακες του παραρτήματος Α λήφθηκαν ceteris paribus, και επομένως η σύγκριση μεταξύ των αποτελεσμάτων έχει νόημα.

Ο NCC από την άλλη παρατηρούμε πως είναι αρκετά γρήγορος. Στη περίπτωσή μας έχουμε κάνει reshape(flatten) το dataset από 2D σε 1D (28 \* 28 to 784), επομένως έχει νόημα η ευκλείδεια απόσταση. [Γενικά δεδομένα σε 1D χρησιμοποιούνται σε πλήρως συνδεδεμένα νευρωνικά δίκτυα (όπως MLP), ενώ δεδομένα σε 2D χρησιμοποιούνται για συνελικτικά νευρωνικά δίκτυα.] Για σκοπούς παρατήρησης, υλοποιήθηκε ο NCC και με Manhattan απόσταση, ωστόσο αυτό είχε σημαντική επίδραση (μείωση) στην ακρίβειά του.

Παρατηρούμε επίσης στα γραφήματα 1 και 2 πως ο KNN έχει αρκετά μικρό χρόνο fit και πολύ μεγαλύτερο predict σε σύγκριση με τον NCC. Η μεγάλη διαφορά που βλέπουμε στους χρόνους μεταξύ KNN και NCC έγκειται στο γεγονός πως η περισσότερη δουλειά στον NCC γίνεται κατά τη διαδικασία fit, όπου ο αλγόριθμος υπολογίζει τα κέντρα (centroids) των κλάσεων, και επομένως στη συνέχεια απλώς συγκρίνει τις αποστάσεις αυτών από το σημείο που μας ενδιαφέρει. Δηλαδή στη συγκεκριμένη περίπτωση υλοποιεί 10 συγκρίσεις αφού έχουμε 10 κλάσεις. Ο ΚΝΝ κάθε φορά ψάχνει τους πλησιέστερους Κ γείτονες και υπολογίζει πόσοι από αυτούς ανήκουν στην κάθε κλάση, ώστε να κατηγοριοποιήσει το σημείο που μελετάμε. Για το λόγο αυτό παρατηρούμε και μια μικρή αύξηση στο χρόνο όσο η παράμετρος Κ αυξάνεται.

Τελικά, ο NCC είναι σαφώς πιο γρήγορος από τον KNN. Ωστόσο, η ακρίβεια των προβλέψεων του NCC είναι αρκετά πιο χαμηλή (στην καλύτερη περίπτωση 82,03% έναντι 97,05% του KNN).

## CROSS VALIDATION VS KNN ACCURACY

Χρησιμοποιήσαμε τη τεχνική Cross Validation στον ΚΝΝ με 3 γείτονες καθώς αυτός είχε τη μεγαλύτερη ακρίβεια στη προηγούμενη σύγκριση. Βλέπουμε, από το γράφημα 6, πως η ακρίβεια αυξάνεται για μεγαλύτερα K-Folds, ενώ για Κ=35 φτάνουμε στην μέγιστη ακρίβεια που παρατηρήθηκε, 0,98664 ή 98,664%. Ακόμη όμως παρατηρούμε πως με τη τεχνική CV, οι περισσότερες μετρήσεις μας έχουν ακρίβεια μεγαλύτερη από 97,05%, η οποία ήταν η μέγιστη στο μοντέλο ΚΝΝ χωρίς CV. Τελικά, συμπεραίνουμε πως με τη τεχνική CV, η ακρίβεια στα αποτελέσματά μας είναι σαφώς καλύτερη, ειδικά όσο αυξάνουμε τα folds.

# Παράρτημα

## (Α) Πίνακες Μετρήσεων

Table 1: Μετρήσεις fit time και predict time KNN & NCC

|  |  |  |
| --- | --- | --- |
| **Model** | **Fit time (s)** | **Predict time (s)** |
| KNN 1 | 0,004010677337646484 | 30,921151638031006 |
| KNN 3 | 0,003977298736572266 | 32,90735363960266 |
| KNN 5 | 0,0053746700286865234 | 31,293059825897217 |
| KNN 7 | 0,0 | 34,9021737575531 |
| KNN 9 | 0,01564168930053711 | 35,49791479110718 |
| KNN 11 | 0,0 | 36,85045766830444 |
| KNN 13 | 0,004988431930541992 | 34,69837188720703 |
| KNN 15 | 0,006287336349487305 | 34,118040561676025 |
| KNN 17 | 0,003983736038208008 | 34,49005627632141 |
| KNN 19 | 0,0057260990142822266 | 34,049394607543945 |
| KNN 21 | 0,010051488876342773 | 32,45123600959778 |
| KNN 23 | 0,005588054656982422 | 31,26777744293213 |
| KNN 25 | 0,00497126579284668 | 37,37762904167175 |
| NCC Euclidean | 0,08776473999023438 | 0,06194138526916504 |
| NCC Manhattan | 0,3364279270172119 | 0,07768750190734863 |

Table 2: Μετρήσεις συνολικού χρόνου και ακρίβειας KNN & NCC

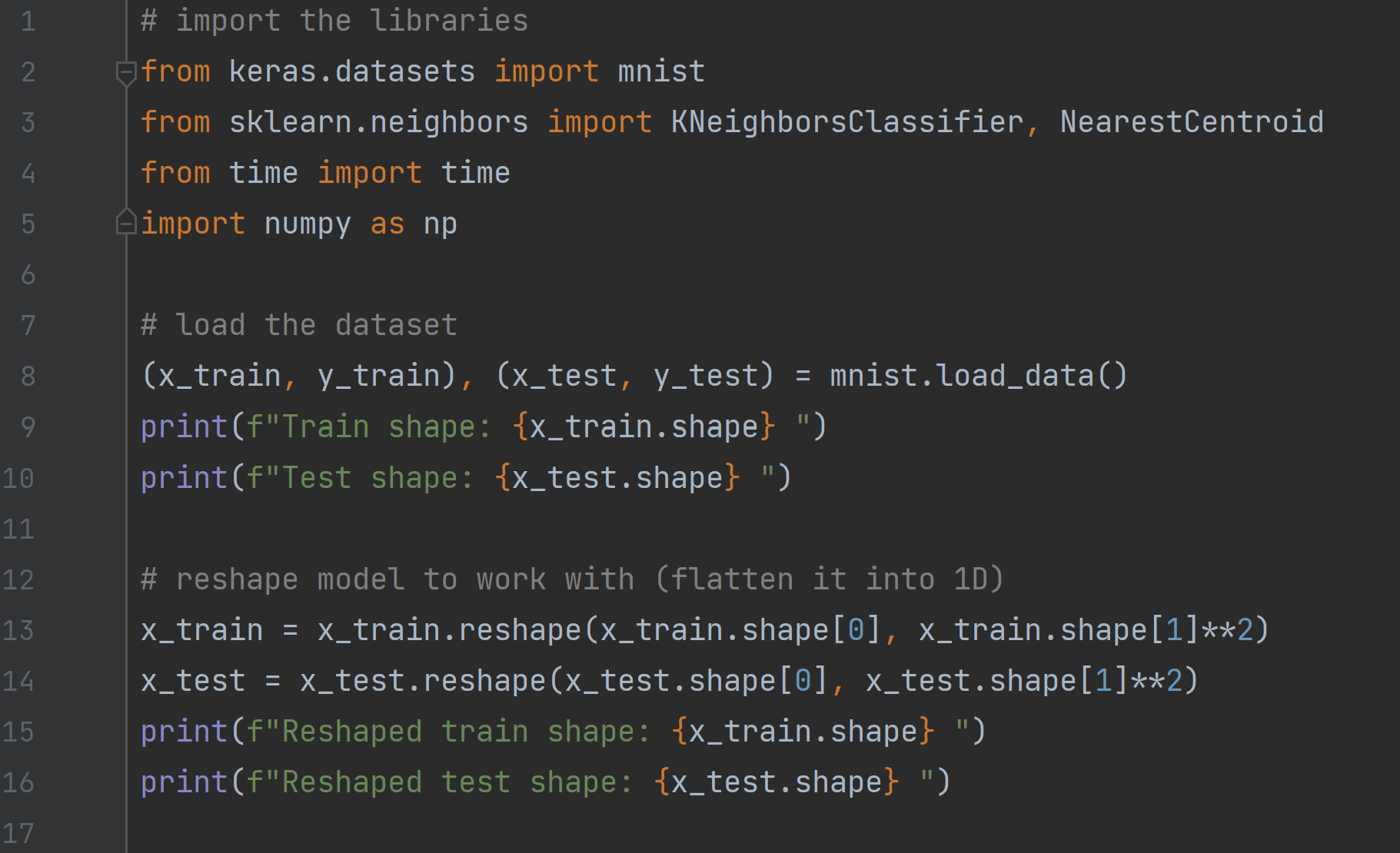
|  |  |  |
| --- | --- | --- |
| **Model** | **Total time (s)** | **Score / Accuracy** |
| KNN 1 | 30,925162315368652 | 0,9691 or 96,91% |
| KNN 3 | 32,91133094 | 0,9705 or 97,05% |
| KNN 5 | 31,2984345 | 0,9688 or 96,88% |
| KNN 7 | 34,90217376 | 0,9694 or 96,94% |
| KNN 9 | 35,51355648 | 0,9659 or 96,59% |
| KNN 11 | 36,85045767 | 0,9668 or 96,68% |
| KNN 13 | 34,70336032 | 0,9653 or 96,53% |
| KNN 15 | 34,1243279 | 0,9633 or 96,33% |
| KNN 17 | 34,49404001 | 0,963 or 96,3% |
| KNN 19 | 34,05512071 | 0,9632 or 96,32% |
| KNN 21 | 32,4612875 | 0,963 or 96,3% |
| KNN 23 | 31,2733655 | 0,9619 or 96,19% |
| KNN 25 | 37,3826003074646 | 0,9609 or 96,09% |
| NCC Euclidean | 0,14970612525939942 | 0,8203 or 82,03% |
| NCC Manhattan | 0,41411542892456053 | 0,7535 or 75,35% |

Table 3: Μετρήσεις ακρίβειας τεχνικής Cross Validation σε ΚΝΝ Classifier με 3 γείτονες

|  |  |
| --- | --- |
| **Folds** | **Score / Accuracy** |
| 2 | 0,9593788941437034 |
| 3 | 0,9682804674457429 |
| 4 | 0,9632764167285326 |
| 5 | 0,966621788919839 |
| 6 | 0,9699498327759198 |
| 7 | 0,9666155676764869 |
| 8 | 0,9688492063492065 |
| 9 | 0,9760831937465103 |
| 10 | 0,9766325263811299 |
| 11 | 0,9738651667052085 |
| 12 | 0,9760700969425803 |
| 13 | 0,9771941644009209 |
| 14 | 0,9705019725913621 |
| 15 | 0,9777170868347338 |
| 16 | 0,9777328934892543 |
| 17 | 0,9794091221394217 |
| 18 | 0,982182940516274 |
| 19 | 0,9810868155831909 |
| 20 | 0,9827340823970039 |
| 21 | 0,9838512149045665 |
| 22 | 0,9849648243957188 |
| 23 | 0,983312871315073 |
| 24 | 0,9821471471471472 |
| 25 | 0,9838341158059468 |
| 26 | 0,9821866539257845 |
| 27 | 0,9838350335862772 |
| 28 | 0,9827352335164835 |
| 29 | 0,9838618501431463 |
| 30 | 0,9855084745762712 |
| 31 | 0,9849735573639326 |
| 32 | 0,9849819862155391 |
| 33 | 0,9866340169370473 |
| 34 | 0,9849419448476053 |
| 35 | 0,9866408101702218 |
| 36 | 0,986077097505669 |
| 37 | 0,9861072807501378 |
| 38 | 0,9849640724150803 |
| 39 | 0,9861120994330986 |

## (Β) Κωδικας

### KNN & NCC



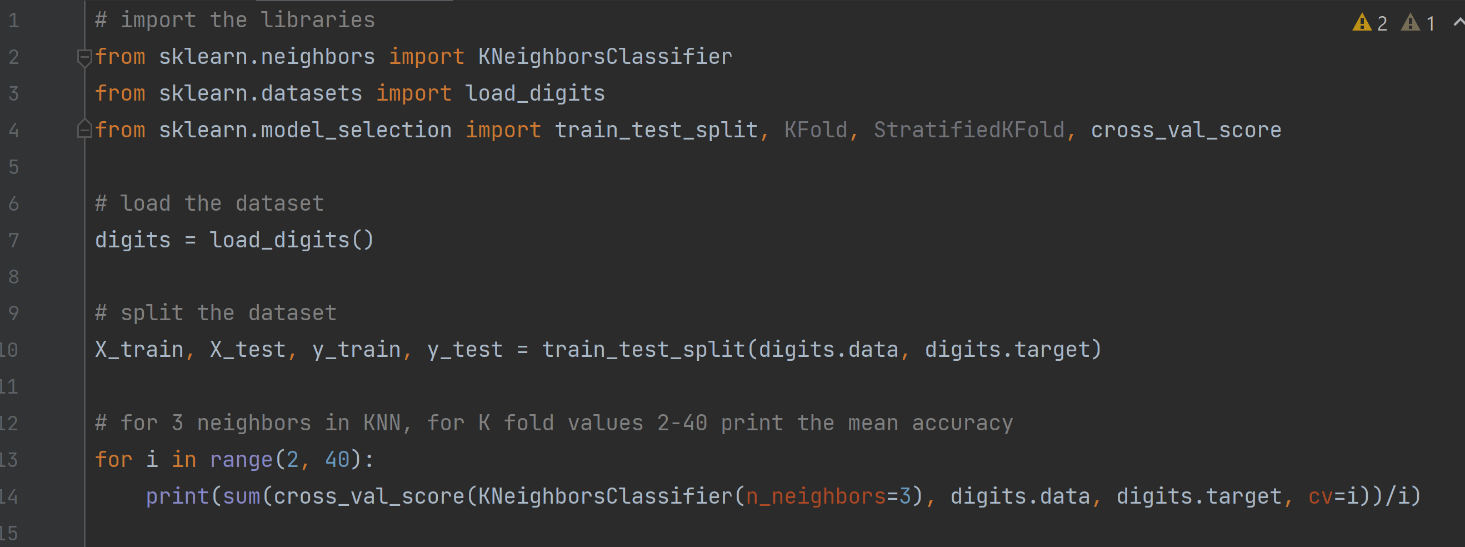
Text

Description automatically generated

Text

Description automatically generated

### KNN Cross Validation



1. MNIST Dataset: <http://yann.lecun.com/exdb/mnist/> [↑](#footnote-ref-1)
2. Keras: <https://keras.io/about/> [↑](#footnote-ref-2)
3. Sklearn: <https://scikit-learn.org/stable/index.html> [↑](#footnote-ref-3)