Μηχανική Μάθηση: Μέθοδοι και Αλγόριθμοι

Μεταπτυχιακή απαλλακτική εργασία ΦΕΒ. 2023-24

Καθηγητής:
Ο. Τελέλης

Καμπάσης Παναγιώτης
Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων
Μεγάλα Δεδομένα και Αναλυτική
2 Φεβρουαρίου 2024

Μηχανική Μάθηση: Μέθοδοι και Αλγόριθμοι

Πάνος Καμπάσης
2 Φεβρουαρίου 2024

Περιεχόμενα

Περίληψη—Στο παρόν θα αναλύσουμε τις πωλήσεις προϊόντων από την πλατφόρμα *eBay*. Θα χρησιμοποιήσουμε μοντέλα και τεχνικές μηχανικής μάθησης, ελέγχοντας τα αποτελέσματα και τις επιδόσεις τους. Θα επιλέξουμε το πιο αποδοτικό μοντέλο με βάση τα αποτελέσματα στις διάφορες πρακτικές κατηγοριοποίησης που θα εφαρμόσουμε και θα παρουσιάσουμε συγκριτικά χαρακτηριστικά μεταξύ τους.

Λέξεις κλειδιά:
Μηχανική Μάθηση, Παλινδρόμηση, *classification*, *Random Forest*, *Gradient Boosting Regression*, *Support Vector Regressor*, *Multilayered Perceptron*.

I. Η Βάση Δεδομένων

Χρησιμοποιούμε την βάση *eBay Auction Sales* η οποία αποτελείται από 28 χαρακτηριστικά. Η βάση αφορά αγορές αντικειμένων από πλειστηριασμούς και περιέχει ένα μεγάλο όγκο δεδομένων όσον αφορά την αγόρα και τα διάφορα επί μέρους χαρακτηριστικά του εκάστοτε πλειστηριασμού.

II. Προ-επεξεργασία των Δεδομένων

Για την ανάλυση μας αρχικά μελετούμε την βάση δεδομένων για την αναγνώριση της. Κοιτούμε για ασυνέχειες, κενές τιμές, μορφή των δεδομένων (κατηγορικά, αριθμητικά ή άλλα), εάν υπάρχει μεγάλη διαφορά εύρους κλπ.
η βάση μας έχει αυτή τη μορφή:

[8]:

	EbayID	QuantitySold	Price	PricePercent	StartingBidPercent	SellerName	SellerClosePercent	Category	PersonID	StartingBid
0	4.004762e+11	0.0	0.99	0.5892	0.5892	harryjean0	0.128269	73409.0	8215.0	0.99
1	3.507850e+11	0.0	119.00	1.1427	1.1427	rrsports23	0.116667	27260.0	34.0	119.00
2	3.806299e+11	1.0	1.75	1.0416	0.4464	mojo640	0.531599	73409.0	8215.0	0.75
3	3.008968e+11	1.0	66.00	0.6338	0.0000	mintsignatures	0.971014	27260.0	34.0	0.01
4	2.009189e+11	0.0	9.99	0.5197	0.5197	realdealsignatures	0.187117	27285.0	28504.0	9.99

5 rows × 56 columns

Σχήμα 1: Η βάση δεδομένων

Σαν πρώτη παρατήρηση η βάση μας περιέχει μεικτές αλφαριθμητικές τιμές και πρέπει να αγνωρίσουμε ποιοι είναι οι πιο σημαντικοι παράγοντες που πρέπει να διατηρήσουμε. Για μια κλασική κατηγοριοποίηση προτιμούμε αριθμητικές τιμές και σαν πρώτο βήμα αναζητούμε τα χαρακτηριστικά που θα διατηρήσουμε και αυτά που θα αποβάλουμε από την ανάλυση μας. Αναζητώντας την βάσης για την αναγνώριση χαρακτηριστικών που περιέχουν μόνο χαρακτήρες αφαιρούμε από την ανάλυση τα $'SellerName', 'EndDay'$ (Σε αυτό το σημείο σχολιάζουμε πως δεν είναι πάντοτε καλή πρακτική η αφαίρεση χαρακτηριστικών με αυτόν τον τρόπο. Η λογική πίσω από αυτή μας την κίνηση είναι πως χρειάζεται να δουλέψουμε με αριθμούς ή να μετατρέψουμε τις λέξεις σε διανύσματα ή *dummy variables* με τεχνικές $one - hot\ encoding$. Σε μια πιο αυστηρή ανάλυση θα επιθυμούσαμε την συμβόλη (οσοδήποτε μικρή) κάθε χαρακτηριστικού. Στο παρόν δοκιμάσαμε πολλές μεθόδους μείωσης διαστάσεων όπως θα δούμε παρακάτω και είδαμε την μικρή συμβολή των εν λόγω 2 χαρακτηριστικών σε *gini* και εντροπία, άρα επιλέξαμε να τις αφαιρέσουμε εξ αρχής).

III. Πρόβλεψη του χαρακτηριστικού *Price*

Χρησιμοποιώντας $RandomForest$ για να βρούμε τα πιο σημαντικά χαρακτηριστικά καταλήγουμε στα παρακάτω:

$Feature$	$Importance$
$PriceBuckets$	0.998804
$StartingBid$	0.000583
$PricePercent$	0.000103
$BidCount$	0.000100
$AvgPrice$	0.000083
$AuctionMedianPrice$	0.000051
$HitCount$	0.000025
$AuctionHitCountAvgRatio$	0.000025
$ItemAuctionSellPercent$	0.000023
$StartingBidPercent$	0.000021
$SellerAvg$	0.000018

Φαίνεται ότι το χαρακτηριστικό $PriceBuckets$ συγκρατεί σχεδόν στην ολότητα της την πληροφορία που είναι πιο καθοριστική για την παλινδρόμηση μας. Επιχειρούμε ωστόσο να δοκιμάσουμε ένα μοντέλο παλιδρόμησης για να ελέγξουμε τα αποτελέσματα του.

A′. *Gradient Boosting Regression*

Εκπαιδεύοντας το μοντέλο έχουμε:

$$MeanSquaredError : 1.29$$

$$R - squared : 0.99$$

Το οποίο είναι αρκετή ένδειξη ότι έχουμε $overfit$. Φαίνεται άλλωστε και από το $Importace$ παραπάνω πόσο συγκετρωμένο είναι σε ένα μόνο χαρακτηριστικό. Στο παρόν δεν χρειάζονται 11 χαρακτηριστικά καθώς μόνο τα

πρώτα 2 μπορούν να δώσουν τον ίδιο συντελεστή συσχέτισης.
Έτσι επιλέγουμε να αφαιρέσουμε το χαρακτηριστικό και να δούμε την επίπτωση των υπολοίπων στην εκπαίδευση του μοντέλου μας.
Μετά την αφαίρεση του *PriceBuckets* η *RandomForest* δίνει:

Feature	*Importance*
AuctionMedianPrice	0.385428
AvgPrice	0.305747
PricePercent	0.302389
StartingBid	0.002108
SellerAvg	0.001598
BidCount	0.000515
IsInMedianRatio25Percent	0.000257
HitCount	0.000240
IsInMedianRatio20Percent	0.000181
SellerItemAvg	0.000146
AuctionHitCountAvgRatio	0.000144

Από αυτά θα επιλέξουνε να κρατήσουμε τα 3 πρώτα καθώς στα επόμενα πέφτει σημαντικά το *Importance*. Επομένως εκπαιδεύοντας και πάλι το μοντέλο:

$$MeanSquaredError : 17.5895$$

$$R - squared : 0.99$$

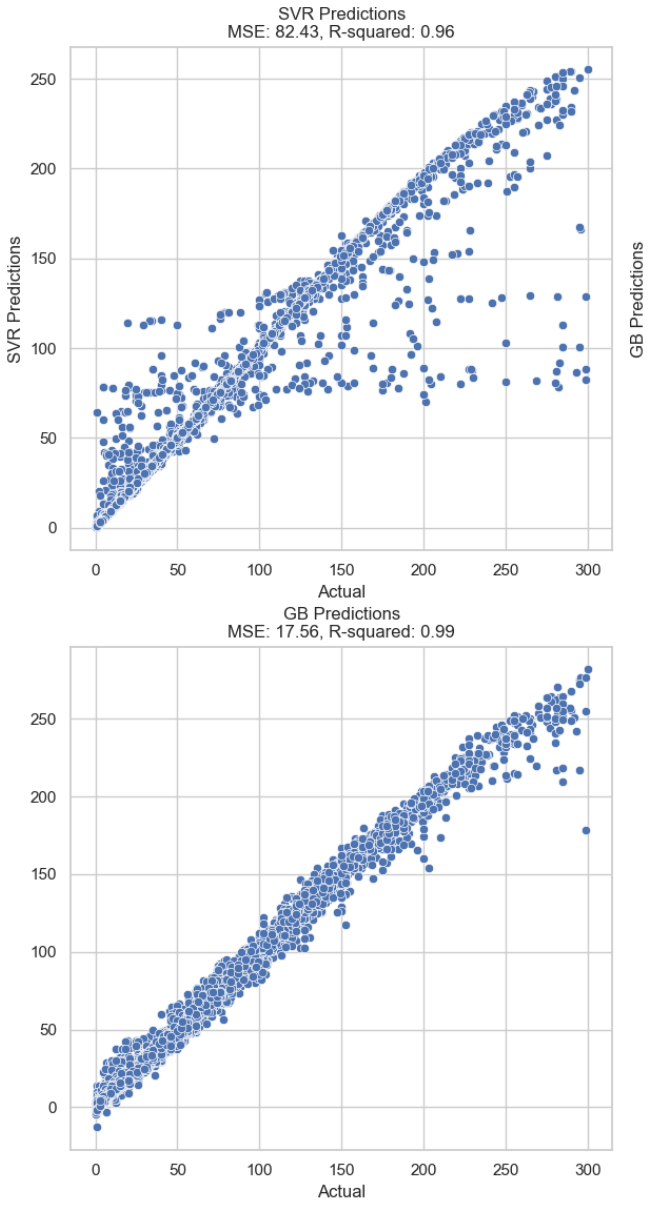
Τα αποτελέσματα αυτά δείχνουν την μεγάλη αποδοτικότητα του μοντέλου δεδομένου ότι προέκυψαν χωρίς ρύθμιση των υπερπαραμέτρων. Συνάγουμε επίσης ότι το $overfit$ μειώθηκε λόγο της διαφοράς στο *MeanSquaredError*.

Β′. *Support Vector Regressor*

$$MeanSquaredError : 82.4256$$

$$R - squared : 0.95$$

Εδώ βλέπουμε ότι το μοντέλο *Support Vector Regressor* πετυχαίνει το ίδιο ικανοποιητικά αποτελέσματα, συγκρίνοντας το παρακάτω γράφηματα θα καταλήγαμε στη *Gradient Boosting Regression*. Δεν θα απορρίπταμε την *Support Vector Regressor* καθώς επιτυχνάνει πολύ καλά αποτελέσματα χώρις ιδιαίτερη ρύθμιση των υπερπαραμέτρων. Επί της ουσίας, αν ο στόχος ήταν η εξοικονόμηση χρόνου θα επίλεγαμε την *Support Vector Regressor* αφού πετυχαίνει ανάλογη ακρίβεια σε πολύ πιο σύντομο χρόνο.



Σχήμα 2: *Support Vector Regressor*
Gradient Boosting Regression

Βλέπουμε την πάρα πολύ καλή πρόβλεψη της *Support Vector Regressor* , και την σχεδόν τέλεια πρόβλεψη της *Gradient Boosting Regression*. Αυτό είναι μια ένδειξη που μας παραπέμπει να εξετάσουμε εάν προσπέσαμε σε $overfit$. Αυτό φαίνεται χαρακτηριστικά από το *Importance* που εμφανίζουν τα χαρακτηριστικά όπως το *PriceBuckets*, *AvgPrice* και γενικά δοκιμάζοντας την *RandomForest* και αφαιρώντας τα 3 πρώτα χαρακτηριστικά *Auction Median Price* ,*AvgPrice* και *Price Percent* για να αποφύγουμε *bias* εκπαιδεύουμε και πάλι τα μοντέλα μας και καταφτάνουμε στην παρακάτω

μορφή:

$$SVRpredictions$$

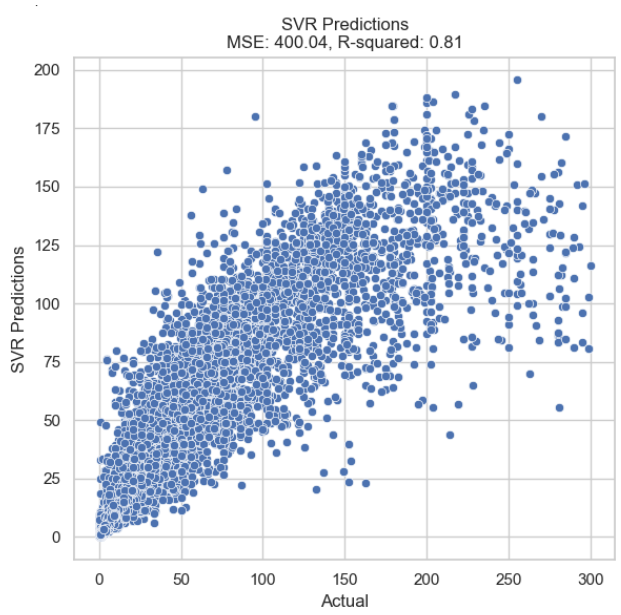
$$MeanSquaredError : 400.0419$$

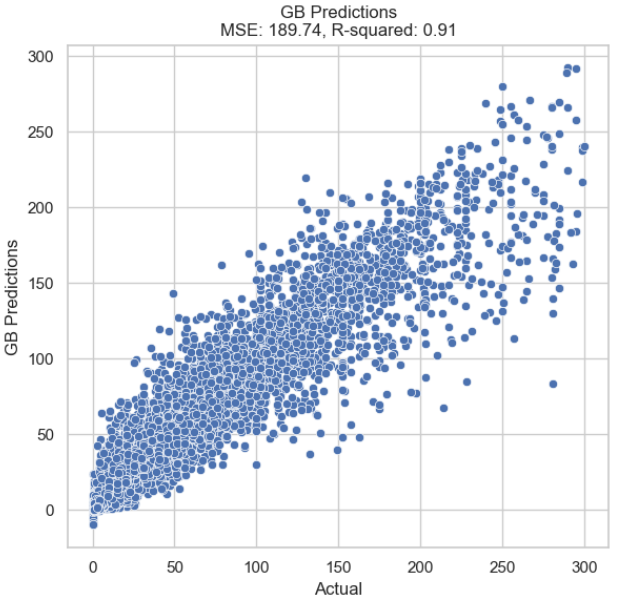
$$R - squared : 0.80$$

$$GBpredictions$$

$$MeanSquaredError : 189.7446$$

$$R - squared : 0.90$$





Σχήμα 3: *Support Vector Regressor*
Gradient Boosting Regression

Έτσι καταλήγουμε και πάλι στο *Gradient Boosting Regression* και επιτρέπουμε το σφάλμα που αποδίδει καθώς το μοντέλο μας είναι καλύτερο στη γενίκευση. Επίσης σημειώνουμε πως επιτυγχάνεται $R^2 = 90\%$, μόνο από τις τιμές που δεν είχαν ισχυρή συσχέτιση με το *Price*.

IV. Πρόβλεψη του χαρακτηριστικού *QuantitySold*

Η *RandomForest* προτείνει τις:

Feature	*Importance*
HitCount	0.448238
SellerClosePercent	0.099886
StartingBid	0.087800
SellerSaleAvgPriceRatio	0.039442
StartingBidPercent	0.036516
BestOffer	0.031771
AuctionCount	0.028522
PricePercent	0.027228
Price	0.025125
AuctionMedianPrice	0.022461
AvgPrice	0.020987

Τα αποτελέσματα της κατηγοριοποίησης χρησιμοποιόντας *Logistic Regression with Gradient Descent*:

$$MeanSquaredError(MSE) : 0.1208$$

$$RootMeanSquaredError(RMSE) : 0.34$$

$$R - squared(R^2) : 0.42$$

$$ClassificationReport :$$

	precision	*recall*	$f1 - score$	*support*
0	0.88	0.95	0.92	41363
1	0.87	0.70	0.78	17847
accuracy			0.88	59210
macroavg	0.88	0.83	0.85	59210
weightedavg	0.88	0.88	0.88	59210

Με την παραπάνω μέθοδο πετυχαίνουμε ακρίβεια 88%. Από την ακρίβεια αλλά και τις παραπάνω μετρικές τα αποτελέσματα είναι ικανοποιητικά όμως θα προσπαθήσουμε να ενισχύσουμε την εκπαίδευση του μοντέλου.
Τέλος εφαρμόζουμε $Grid - search$ για να βελτιώσουμε τις υπερπαραμέτρους του μοντέλου και μετά από πολλές εκπαιδεύσεις καταλήγουμε στις παρακάτω μετρικές:

$$BestHyperparameters :$$

$$'C' : 100,' max_iter' : 100,' penalty' :' l1',' solver' :' liblinear'$$

$Classification Report$:

$precision$	$recall$	$f1-score$	$support$	
0	0.89	0.97	0.93	41363
1	0.92	0.72	0.81	17847
$accuracy$			0.90	59210
$macroavg$	0.90	0.85	0.87	59210
$weightedavg$	0.90	0.90	0.89	59210

Α′. $Support\ Vector\ Classifier$

Με τις παρακάτω παραμέτρους η εκπαίδευση του μοντέλου δίνει τις παρακάτω μετρικές:

$$C = 1.0, kernel =' rbf', degree = 3$$

$$Accuracy : 0.89$$

$$Mean\ Squared\ Error\ (MSE) : 0.1057$$

$$Root\ Mean\ Squared\ Error\ (RMSE) : 0.32$$

$$R - squared\ (R^2) : 0.49$$

$ClassificationReport$:

$precision$	$recall$	$f1-score$	$support$	
0	0.89	0.97	0.93	41363
1	0.92	0.72	0.80	17847
$accuracy$			0.89	59210
$macro\ avg$	0.90	0.84	0.87	59210
$weighted\ avg$	0.90	0.89	0.89	59210

Η λογιστική με $Gradient\ Boosting$ φαίνεται να αποδίδει καλύτερα στο συγκεκριμένο πρόβλημα κατά ελάχιστα (1% ακρίβεια κατηγοριοποίησης). Ωστόσο, αυτό επιτυγχάνεται μετά από $grid-search$ και επιπρόσθετα δεν έχει καλύτερο MSE και R^2 από την $Support\ Vector\ Classification$. Η $Support\ Vector\ Classification$ αντίθετα πετυχαίνει ελαφρώς καλύτερα αποτελέσματα χώρις ρύθμιση των παραμέτρων και έτσι την προτιμούμε για το παραπάνω πρόβλημα καθώς φαίνεται να αποδίδει σωστά στη συγκεκριμένη βάση δεδομένων.

V. Νέο $feature$ για την πρόβλεψη των πωλήσεων άνω του μέσου όρου

$Model1 : Logistic\ Regression\ with\ Gradient\ Descent$

$Model2 : Support\ Vector\ Classifier$

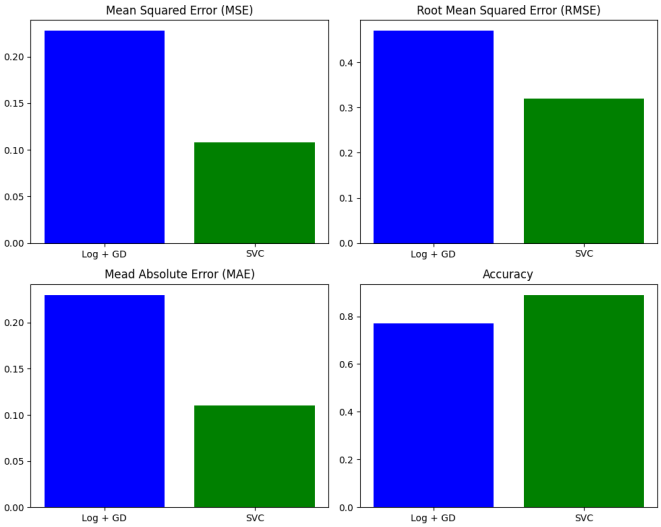
Δημιουργούμε μια νέα στήλη που ονομάζουμε $target$. Η στήλη αυτή θα περιέχει τις τιμές A.A. και B.A. που σημαίνουν αντίστοιχα $AboveAverage$ και $BelowAverage$. Στόχος να κάνουμε μια νέα εκπαίδευση και νέα κατηγοριοποίηση στο αν πωλήθηκε κάτι άνω του μέσου όρου τιμής πωλησης του η όχι.

Με την εντολή

$$data['target'] = (data['Price'] >= data['AvgPrice']).astype(int)$$

προσθέτουμε το νέο μας χαρακτηριστικό και δοκιμάζουμε τους δύο προηγούμενους κατηγοριοποιητές και τα αποτελέσματα τους είναι:

```
Log + GD
Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.84      0.84     12314
           1       0.63      0.63      0.63      5511

    accuracy                           0.77     17825
   macro avg       0.73      0.73      0.73     17825
weighted avg       0.77      0.77      0.77     17825

SVC
Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.94      0.92     12314
           1       0.85      0.79      0.82      5511

    accuracy                           0.89     17825
   macro avg       0.88      0.86      0.87     17825
weighted avg       0.89      0.89      0.89     17825
```



$$\begin{bmatrix} & MSE & RMSE & MAE & Accuracy \\ Log\ with\ GD & 0.2278 & 0.47 & 0.23 & 0.77 \\ SVC & 0.1082 & 0.32 & 0.11 & 0.89 \end{bmatrix}$$

Τα αποτελέσματα της λογιστικής παλινδρόμησης (με $Gradient\ boosting$) είναι πολύ καλά, όμως η $Support\ Vector\ Classification$ αποδίδει καλύτερα σύμφωνα με τον παραπάνω πίνακα. Ουσιαστικά, αφαιρώντας τα χαρακτηριστικά που σχετίζονται άμεσα με το $target$ πετυχαίνουμε να προβλέπουμε την τιμή του με σφάλμα μόλις 10%. Για αυτούς τους λόγους προτιμούμε την SVC για το συγκεκριμένο πρόβλημα.

VI. Κατηγοριοποίηση εικόνων

Α΄. Προεπεξεργασία

Η ιδιαίτερη δομή της βάσης δεδομένων $CIFAR-10$ μας οδηγεί σε ειδικά βήματα για την προεπεξεργασία της. Με τον κώδικα που αναπτύσσουμε αρχικά κάνουμε *flat* (προβάλουμε τα πολυδιάστατα δεδομένα σε μια διάσταση) τα χαρακτηριστικά μας και κάνουμε κανονικοποίηση στις τιμές που αντιστοιχούν στα *pixel*. Την ίδια προβολή θα κάνουμε και στα *labels*. Οι κατηγοριοποιητές εκπαιδεύονται πάνω σε αυτά τα διανύσματα και βγάζουμε την παρακάτω ακρίβεια. Καθώς η βάση δεδομένων παρουσιάζει ιδιαίτερη πολυπλοκότητα δοκιμάζουμε τα μοντέλα με 10,000 δεδομένα και έχουμε:

Β΄. Λογιστική Παλινδρόμηση και *Support Vector Classification*

Πίνακας I: *Classification Reports for Logistic Regression and SVC Models*

Model	*Metrics (per class)*			*Accuracy*
Logistic Regression	0.39	0.40	0.39	0.3345
	0.40	0.38	0.39	
	0.25	0.24	0.25	
	0.23	0.23	0.23	
	0.27	0.26	0.27	
	0.26	0.28	0.27	
	0.35	0.36	0.35	
	0.38	0.36	0.37	
	0.42	0.49	0.45	
	0.38	0.35	0.37	
SVC	0.53	0.55	0.54	0.4734
	0.56	0.58	0.57	
	0.37	0.34	0.35	
	0.33	0.35	0.34	
	0.43	0.36	0.39	
	0.43	0.37	0.40	
	0.45	0.55	0.49	
	0.53	0.48	0.50	
	0.57	0.63	0.60	
	0.51	0.53	0.52	

	Logistic Regression	*SVC*
MSE:	12.6634	9.8929
$RMSE$:	3.5586	3.1453
MAE:	84.7676	67.3473

Τα αποτελέσματα δεν είναι ικανοποιητικά αλλά πρακτικά χρησιμοποιούμε το $\frac{1}{6}$ της βάσης οπότε αναμένα χαμηλή απόδοση. Ο χρόνος εκτέλεσης ήταν 23 λεπτά. Φαίνεται πως η *Support Vector Classification* επιτυγχάνει καλύτερα αποτελέσματα
Δοκιμάζουμε και πάλι με μεγαλύτερο πλήθος δεδομένων(30.000).

Πίνακας II: *Classification Reports for Logistic Regression and SVC Models*

Model	*Accuracy*	*MSE*	*RMSE*	*MAE*
Logistic Regression	0.3752	12.1103	3.47999	78.7415
SVC	0.5206	8.888	2.98127	61.8422

Πίνακας III: *Classification Reports (per class) for Logistic Regression and SVC Models*

Model	*Precision*	*Recall*	*F1-Score*	*Support*
Logistic Regression	0.43	0.45	0.44	1000
	0.44	0.42	0.43	1000
	0.30	0.27	0.28	1000
	0.27	0.24	0.26	1000
	0.32	0.30	0.31	1000
	0.30	0.31	0.30	1000
	0.39	0.43	0.41	1000
	0.42	0.40	0.41	1000
	0.46	0.51	0.48	1000
	0.41	0.41	0.41	1000
SVC	0.59	0.60	0.59	1000
	0.60	0.62	0.61	1000
	0.40	0.38	0.39	1000
	0.36	0.36	0.36	1000
	0.45	0.45	0.45	1000
	0.47	0.41	0.43	1000
	0.51	0.60	0.55	1000
	0.60	0.52	0.56	1000
	0.63	0.67	0.65	1000
	0.58	0.59	0.58	1000

Ολοκληρώνοντας παραθέτουμε τα αποτελέσματα του μοντέλου *Multilayered Perceptron*:

$$Test\ Accuracy : 0.40$$

$$Mean\ Squared\ Error\ (MSE) : 11.7267$$

$$Root\ Mean\ Squared\ Error\ (RMSE) : 3.42$$

$$Mean\ Absolute\ Error\ (MAE) : 2.18$$

Επομένως και εδώ η ακρίβεια μας οδηγει στο SVC ωστόσο η MLP πετυχαίνει καλύτερες μετρικές σε MSE (και $rMSE$), MAE από την λογιστική. Η ακρίβεια είναι λογικό να μην είναι πολύ υψηλή και αυτό λόγω των δέκα κλάσεων που χρειάστηκε να κατανήμει ο κατηγοριοποιητής. Επομένως η *Perceptron* είναι η καλύτερη επιλογή αμέσως μετά την SVC.

Γ′. Βήματα εκπαίδευσης του MLP

Πίνακας IV: *Training History of MLP Model*

Epoch	*Training Loss*	*Training Accuracy*	*Validation Loss*	*Validation Accuracy*
1	1.9898	0.2690	1.8533	0.3132
2	1.8607	0.3148	1.7823	0.3618
3	1.8154	0.3387	1.7593	0.3646
4	1.7819	0.3545	1.7053	0.3910
5	1.7654	0.3612	1.6974	0.4038
6	1.7552	0.3660	1.7214	0.3794
7	1.7348	0.3725	1.6658	0.4122
8	1.7277	0.3728	1.7114	0.4000
9	1.7169	0.3800	1.6544	0.4164
10	1.7046	0.3849	1.6572	0.4114

Αναφορές

[1] *Learning Multiple Layers of Features from Tiny Images*, *Alex Krizhevsky*, 2009.