ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

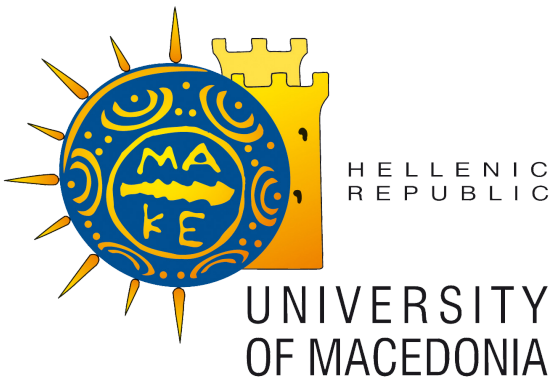
1η Εργασία

Σχολή:
Εφαρμοσμένης Πληροφορικής

Τμήμα:
Εισαγωγή στην Επιστήμη και Τεχνολογία Υπολογιστών

Θέμα: «Regression problems»

Ομάδα:
1. Μιχαήλ Ζέρβας , ics20015
2. Νέστορας Σωτηρίου , ics20012
3. Χρήστος Δρίκος , ics20043



Πίνακας περιεχομένων

Πίνακας περιεχομένων γραφημάτων

Εισαγωγή

Η άσκηση χωρίζεται σε 2 επιμέρους υποπροβλήματα τα οποία όμως έχουν σαν κοινό σκοπό τον εντοπισμό της καλύτερης συνάρτησης πρόβλεψης των αποτελεσμάτων. Οι διαφορές αυτών των δύο αφορούν στη γνώση ή όχι του μοντέλου που δημιουργεί τα δεδομένα. Πιο συγκεκριμένα:

Το πρώτο υποπρόβλημα (1^{ο} μέρος) αφορά την εύρεση της καλύτερης προσέγγισης ανάμεσα σε 2 συναρτήσεις (μία πολυωνυμική και μία συνάρτηση βασισμένη στη συνάρτηση ημιτονου και της εκθετικής) βάση των δεδομένων εισόδου που έχουν θόρυβο, γνωρίζοντας το παραμετρικό μοντέλο. Επισημαίνεται εδώ ότι το μοντέλο που δημιουργεί τα δεδομένα είναι η ίδια ημιτονοειδής συνάρτηση με πριν, ενώ παράλληλα προστίθεται και θόρυβος.

Το δεύτερο υποπρόβλημα (2^{ο} μέρος) έγκειται την εύρεση της καλύτερης δυνατής προσέγγισης ανάμεσα σε 3 διαφορετικές συναρτήσεις γραμμικής παλινδρόμησης (kNN, SVR, decision trees), βάση των δεδομένων εισόδου με θόρυβο που δημιουργήθηκαν πριν, χωρίς να γνωρίζουμε αυτήν την φορά το παραμετρικό μοντέλο.

Μέθοδοι που εφαρμόστηκαν

Γλώσσα που χρησιμοποιήθηκε: Python

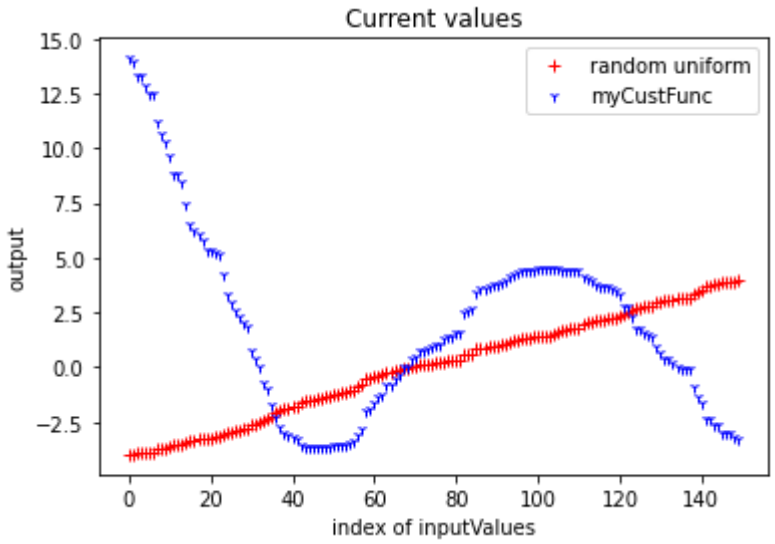
1ª μέρος:

1ο ερώτημα)
Για την δημιουργία των 150 τυχαίων αριθμών στο [-4,4], που ακολουθούν την κανονική κατανομή, χρησιμοποιήθηκε από την βιβλιοθήκη numpy η συνάρτηση random.uniform(), ενώ για την ταξινόμηση των αποτελεσμάτων η συνάρτηση sort. Οι αριθμοί αυτοί στο εξής θα αναφέρονται ως inputValues.

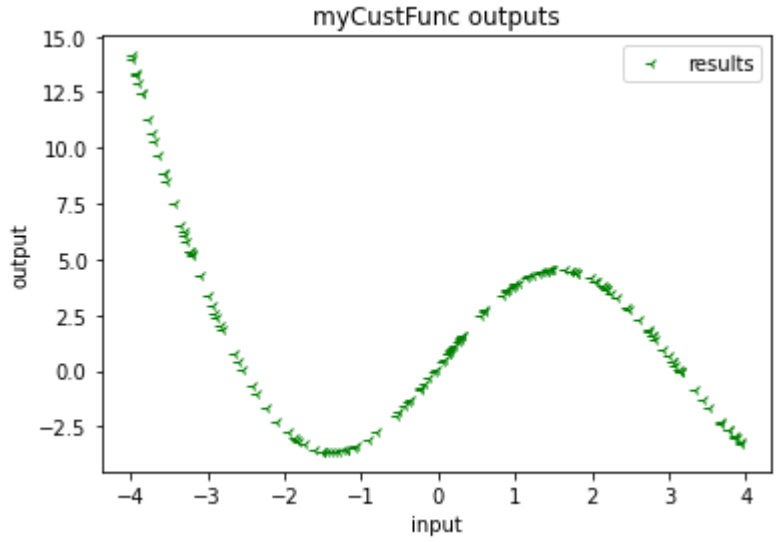
2ο ερώτημα)
Ομοίως για την υλοποίηση της συνάρτησης myCustFunc χρησιμοποιήθηκαν από την βιβλιοθήκη numpy οι έτοιμες συναρτήσεις sin() και exp().

3ο ερώτημα)
Δίνονται τυχαία οι τιμές $\lambda_1 = 0.2$, $\lambda_2 = 4.5$ για τις παραμέτρους της myCustFunc(), και σαν x δίνεται το inputValues. Παράγεται ένα νέο ndarray 150 θέσεων.

4ο ερώτημα)
Σε αυτό το ερώτημα παρότι ζητούσε μία γραφική παράσταση, εμείς παραθέτουμε δύο. Ο λόγος είναι ότι θεωρήσαμε πως δεν διευκρινίζονταν σαφώς αν τα δεδομένα θα έπρεπε να εμφανίζονται συναρτήσει του ενός του άλλου ή όχι. Έτσι καταλήξαμε στα εξής δύο διαγράμματα:



Στο πρώτο διάγραμμα, ο άξονας των x αφορά τη θέση του εκάστοτε στοιχείου στο ταξινομημένο array inputValues, ενώ στον άξονα των y φαίνονται οι τιμές των αντίστοιχων θέσεων του inputValues με χρώμα κόκκινο (οι οποίες δημιουργήθηκαν στο ερώτημα 1) και με χρώμα μπλε οι τιμές που δίνει η myCustFunc με είσοδο την τιμή της αντίστοιχης θέσης κάθε φορά του inputValues.

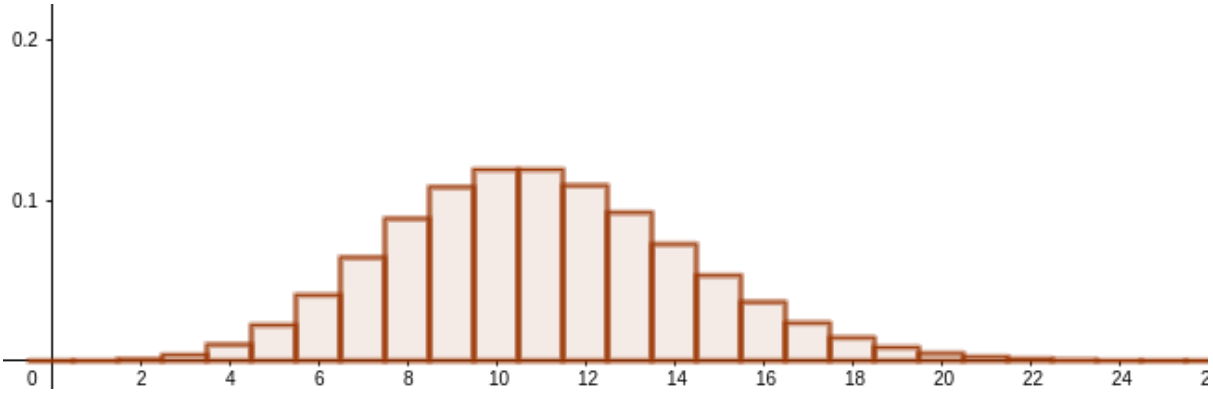


Στο δεύτερο διάγραμμα ο άξονας των x δείχνει την τιμή που δίνονται σαν input και ο άξονας των y δείχνει τα αποτελέσματα που παράγονται από την myCustFunc για τα αντίστοιχα δεδομένα. Τα δεδομένα εισόδου είναι τα ίδια με πριν (inputValues) μόνο που σε αυτό το διάγραμμα φαίνονται πιο καλά οι τιμές εισόδου (x) και εξόδου (y), καθώς και η μεταξύ τους σχέση που προκύπτει μέσω της συνάρτησης myCustFunc.

5ο ερώτημα)
Ο θόρυβος εισήχθη βάση της κατανομής Poisson. Σε κάθε μία εγγραφή του inputValues, προστέθηκε επί αυτής μια τυχαία τιμή από την κατανομή Poisson με λ=11. Για την επίτευξη αυτού έγινε χρήση της αντίστοιχης συνάρτησης από την numpy (random.poisson()).
Η κατανομή Poisson υπολογίζει πόσες φορές μπορεί να συμβεί ένα γεγονός σε ένα συγκεκριμένο χρονικό διάστημα. Αποτελεί μια διακριτή συνάρτηση. Πιο συγκεκριμένα για γεγονότα με αναμενόμενη απόσταση λ η κατανομή Poisson f(k;λ) περιγράφει την πιθανότητα να συμβούν k γεγονότα εντός του παρατηρούμενου διαστήματος λ. Ορίζεται από τον τύπο:

$$f(k;\lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$ και γραφικά έχει την γενική μορφή:

Εμείς χρησιμοποιήσαμε λ=11 και η μορφή της γίνεται ως εξής:



6ο ερώτημα)
Επειδή γνωρίζουμε το παραμετρικό μοντέλο, εδώ μας επιτρέπεται η χρήση της scipy.optimize.curve_fit.

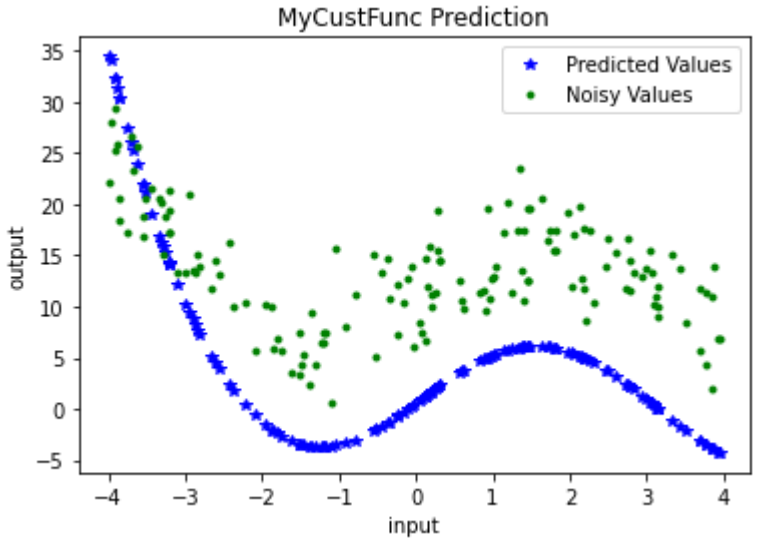
7ο ερώτημα)
Για την δημιουργία του πολυωνύμου χρησιμοποιείται η συνάρτηση polyval της numpy, που παίρνει σαν ορίσματα τον άγνωστο x και τις παραμέτρους του x^n με αύξουσα σειρά για n=0 μέχρι n=4 (στην άσκηση). Έπειτα μετατρέπεται σε ndarray μέσω της εντολής numpy.array() για να μπορεί να χρησιμοποιηθεί από την curve_fit().

8ο ερώτημα)
Χρησιμοποιώντας την scipy.optimize.curve_fit, προσεγγίζουμε τις βέλτιστες παραμέτρους του πολυωνύμου.

9ο ερώτημα)
Πριν κάνουμε τα γραφήματα πρέπει να υπολογίσουμε τα outputs χρησιμοποιώντας τις παραμέτρους που προσεγγίσαμε προηγουμένως και με είσοδο τα inputValues.
Στα διαγράμματα συμπεριλάβαμε και τις Noisy Values για να φαίνεται πόσο καλές προβλέψεις πετυχαίνει κάθε συνάρτηση. Έτσι έχουμε:

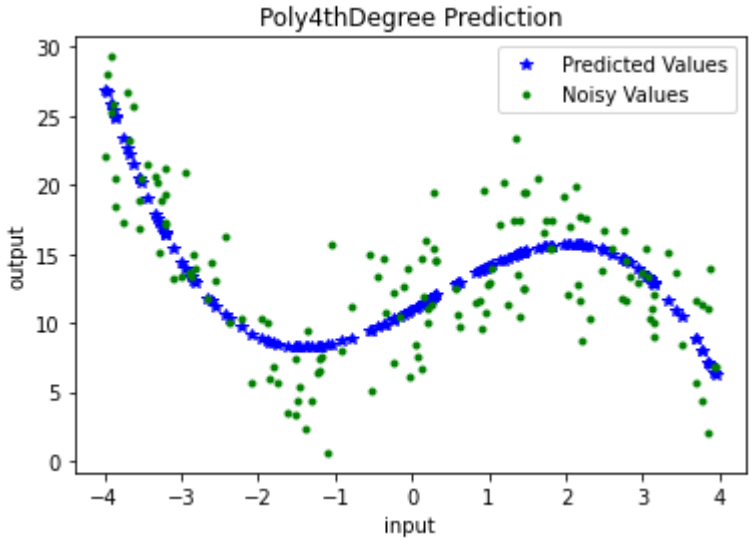
9a)

Το διάγραμμα δείχνει τις τιμές που προβλέπει η συνάρτηση myCustFunc (μπλε χρώμα) και τις Noisy Values που είναι ουσιαστικά οι τιμές θα θέλαμε να προσεγγίζει το μοντέλο μας. Εδώ όμως, βλέπουμε, πως παρόλο στην αρχή τα Predicted Values της MyCustFunc είναι σχετικά καλά, όταν το input περάσει την τιμή κοντά στο -3, τα αποτελέσματα αρχίζουν να απομακρύνονται αρκετά από τα Noisy Values. Έτσι μπορούμε να συμπεράνουμε πως η myCustFunc δεν είναι σχεδόν καθόλου ακριβής στο συγκεκριμένο πρόβλημα. Όσο αυξάνεται ο θόρυβος τόσο θα χειροτερεύει και η ακρίβειά της και αντίστροφα. Σε περίπτωση ελάχιστου θορύβου, η myCustFunc θα μπορούσε να κάνει καλύτερες προβλέψεις λόγω του ότι η συνάρτηση που δημιούργησε τα δεδομένα είναι η ίδια η myCustFunc συνεπώς τα δεδομένα με θόρυβο θα ήταν σχεδόν ίδια με αυτά χωρίς θόρυβο.



9b)

Εδώ βλέπουμε τις αντίστοιχες προβλεφθείσες τιμές της πολυωνυμικής συνάρτησης σε σχέση πάλι με τα Noisy Values. Παρατηρούμε ότι η πρόβλεψη αυτή φαίνεται πιο ακριβής από την προηγούμενη καθώς τα Predicted Values βρίσκονται καλύτερα κατανεμημένα ανάμεσα στα Noisy Values. Η πολυωνυμική συνάρτηση 4^{ου} βαθμού φαίνεται να προσαρμόζεται καλύτερα στα δεδομένα θορύβου.



10ο ερώτημα)

Για τον υπολογισμό των απαιτούμενων μετρικών χρησιμοποιήθηκαν οι έτοιμες συναρτήσεις της sklearn.

Έχουμε:

	Mean Absolute Error (MAE)	Root Mean Square Error (RMSE)
original_values - noisy_values	11.38	11.83
noisy_values - myCustFunc	9.43	10.24
noisy_values - poly4thDegree	2.72	3.31

- Το πολυώνυμο έχει περίπου 67% μικρότερο MAE από την myCustFunc
- Το πολυώνυμο έχει περίπου 71% μικρότερο RMSE από την myCustFunc

Στόχος είναι η επίτευξη όσο των δυνατών χαμηλότερων τιμών για τις μετρικές. Έτσι συγκρίνοντας κάθε φορά ίδιες μετρικές, παρατηρούμε πως η πολυωνυμική συνάρτηση βρίσκει πολύ καλύτερα αποτελέσματα. Τα συμπεράσματα που εξάγαμε από τα διαγράμματα επαληθεύονται και από τις μετρικές.

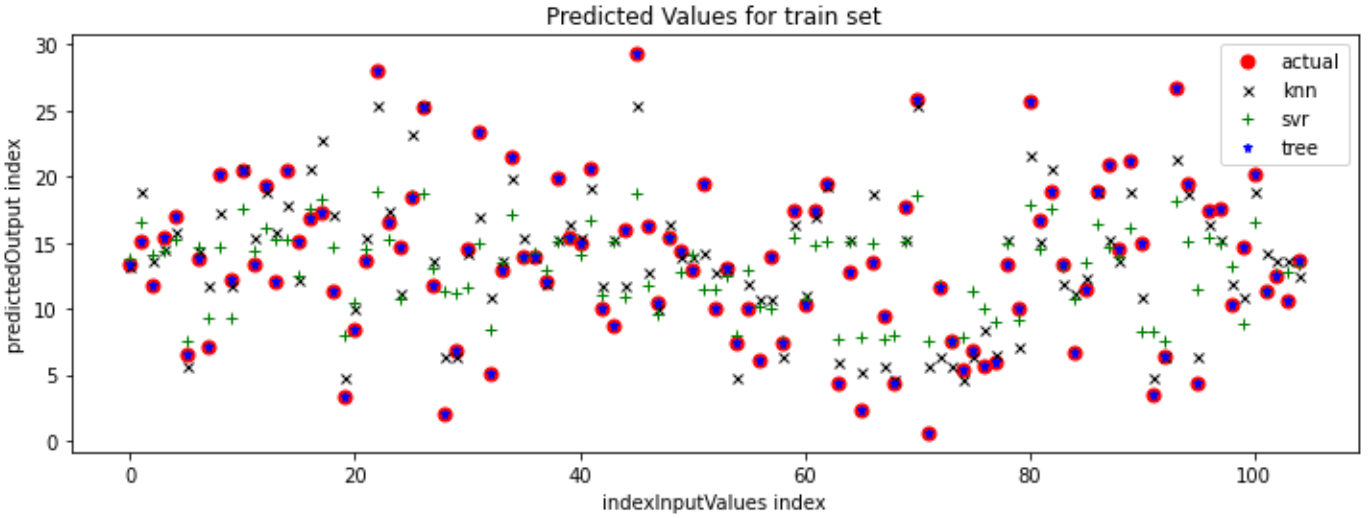
2ο μέρος:

2ο ερώτημα)
Ο διαχωρισμός των δεδομένων σε train, validation και test έγινε ως εξής:

- 70% χρησιμοποιήθηκε για train
- 7% χρησιμοποιήθηκε για validation και
- 23% χρησιμοποιήθηκε για test

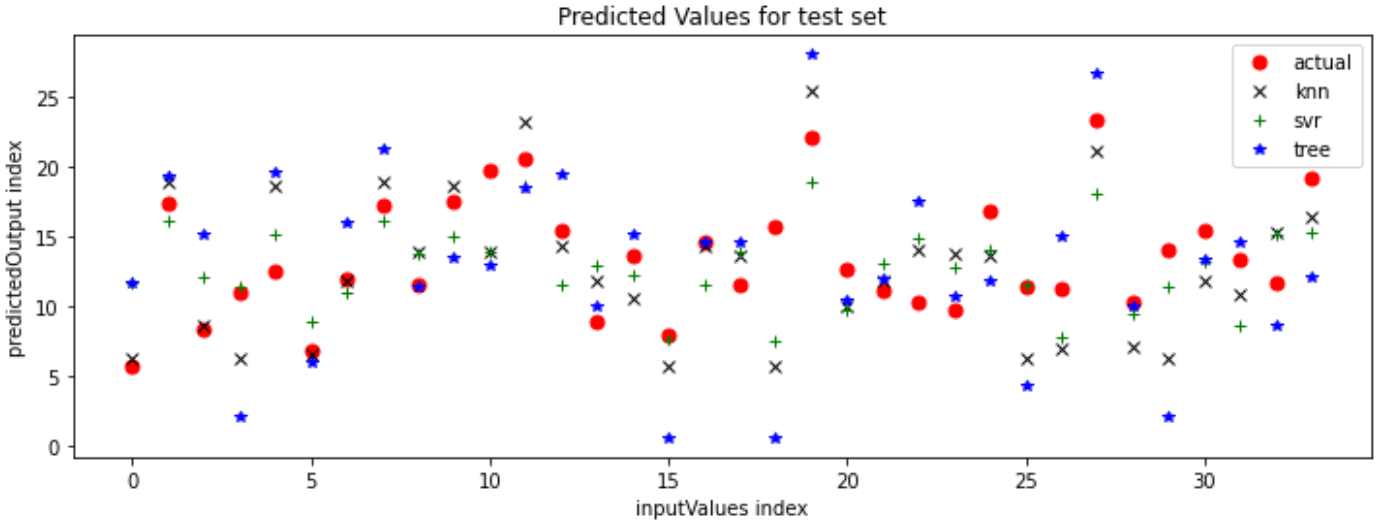
3ο ερώτημα)
Χρησιμοποιήθηκαν οι kNN, SVR και Decision Tree Regressors.
Για την εκπαίδευση όλων των regressors χρησιμοποιήθηκαν η αντίστοιχη συναρτήσεις fit() της sklearn πάνω στα δεδομένα inputValues με noisyInput, στα αντίστοιχα train sets που δημιουργήθηκαν.
Επιπλέον, ο kNN αρχικοποιείται με n=5 γείτονες και ο SVR με kernel την radial basis function (RBF).

4ο ερώτημα)
Για την αξιολόγηση των αποδόσεων χρησιμοποιήθηκε η predict της sklearn, με παράμετρο τα inputValues που αντιστοιχούν στο train set.
Στο πρώτο διάγραμμα παρατηρούμε την ακρίβεια των Predicted Values του κάθε regressor ξεχωριστά σε σχέση με τα actual values. Διακρίνουμε πως το Decision Tree έχει 100% ευστοχία στο train set, δηλαδή κάνει overfitting στα δεδομένα. Αυτό είναι αρκετά αρνητικό και θα μειώσει σημαντικά την απόδοση του στο test set. Αντίθετα οι άλλοι δύο regressors φαίνεται να αποδίδουν εξίσου καλά στο train set.



Στο δεύτερο διάγραμμα φαίνεται το πόσο καλά αποδίδει κάθε regressor σε "άγνωστα" δεδομένα σε σχέση με αυτά της εκπαίδευσης, δηλαδή στο test set. Όπως αναφέρθηκε

προηγουμένως, βλέπουμε πως υπάρχουν ορισμένα αποτελέσματα όπου το prediction του decision tree είναι αρκετά μακριά από την κανονική τιμή κάτι που δεν συμβαίνει στον ίδιο βαθμό και με τους άλλος 2 regressors.

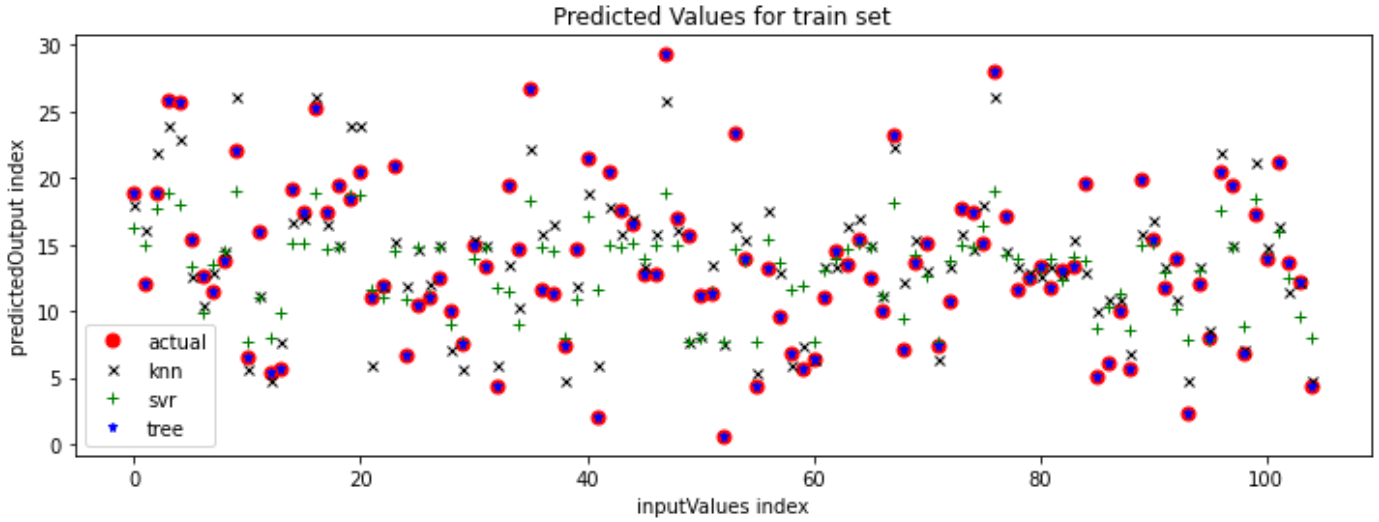


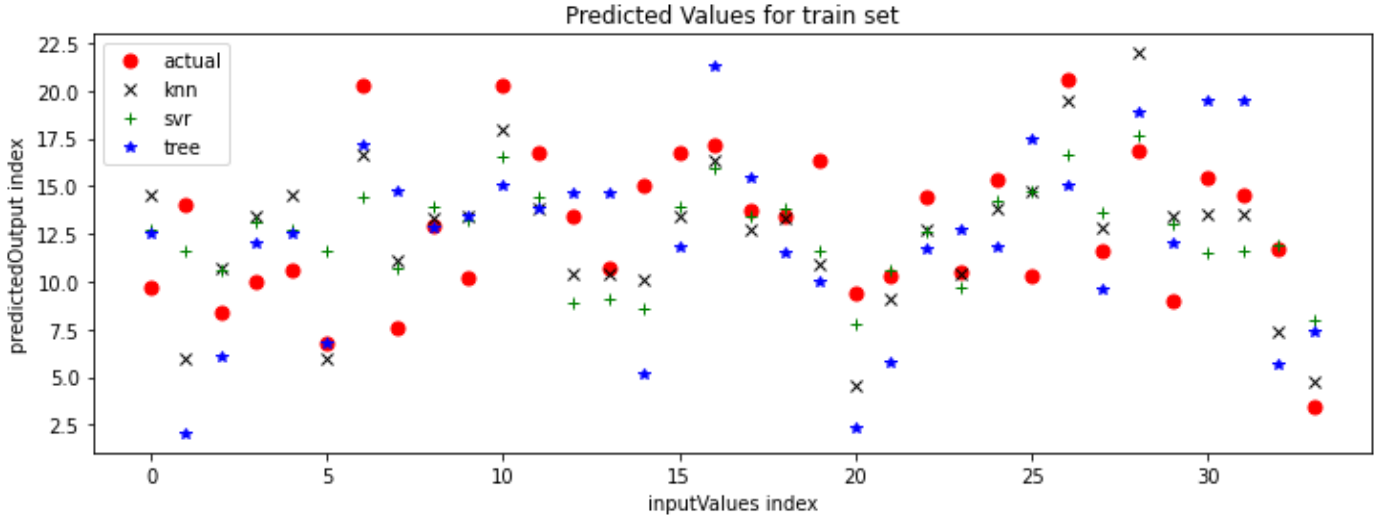
Οι μετρικές υπολογίστηκαν βάση των actual τιμών και των predicted τιμών.

(test set)	MAE	RMSE	MAX ERROR
kNN (k=5)	3.00	3.67	9.97
SVR	2.90	3.38	8.09
Decision Tree	4.31	5.94	15.07

Το MAX ERROR του decision tree είναι αρκετά μεγαλύτερο από τον kNN και του SVR και αυτό οφείλεται στο overfitting. Κατά συνέπεια επηρεάζονται και οι άλλες 2 μετρικές με την RMSE να επηρεάζεται περισσότερο διότι "τιμωρεί" περισσότερο τα outliers, λαμβάνοντας κάθε φορά τα τετράγωνα των αποστάσεων.
Συμπεραίνουμε πως με τα συγκεκριμένα δεδομένα ο SVR είναι πιο αποτελεσματικός από τον kNN και τα Decision Trees, με τον kNN να είναι πιο αποδοτικός μόνο από τα Decision Trees. Όμως επειδή ο kNN δεν αποκλίνει πολύ από τον SVR θα μπορούσαμε να συμπεράνουμε πως μπορούμε να χρησιμοποιήσουμε και τους 2 στο συγκεκριμένο πρόβλημα ή θα μπορούσαμε επίσης να το ξανατρέξουμε πολλές φορές και να δούμε συγκεντρωτικά ποιος είναι καλύτερος.

5ο ερώτημα)
Η κανονικοποίηση έγινε στα inputValues στο διάστημα [0,1] μέσω του MinMaxScaler.





(test set - normalized)	MAE	RMSE	MAX ERROR
kNN (k=5)	2.71	3.29	8.02
SVR	2.68	3.14	6.45
Decision Tree	3.97	4.72	11.95

Μετά την κανονικοποίηση βλέπουμε πως έχουν βελτιωθεί οι 3 μετρικές και των 3 regressors. Αυτό οφείλεται στο ότι με την κανονικοποίηση εξομαλύνθηκαν οι αποστάσεις μεταξύ των σημείων. Βέβαια δεν σημαίνει πως θα μειώνονται οι τιμές κάθε φορά γιατί τα αποτελέσματα εξαρτώνται από τον τυχαίο διαχωρισμό του dataset σε train,test και validation. Γενικά όμως μπορούμε να συμπεράνουμε πως με την κανονικοποίηση υπάρχει μεγαλύτερη πιθανότητα να πετύχουμε μια καλύτερη προσέγγιση.

Συμπεράσματα

Όσον αφορά το πρώτο μέρος, όπου το παραμετρικό μοντέλο είναι γνωστό, η πολυωνυμική συνάρτηση $4^{ου}$ βαθμού αποδίδει πολύ καλύτερα από την myCustFunc. Η μόνη περίπτωση που η myCustFunc θα απέδιδε καλύτερα θα ήταν αν ελαχιστοποιούσαμε τον θόρυβο.
Στο δεύτερο μέρος, όπου το παραμετρικό μοντέλο είναι άγνωστο, τόσο ο SVR όσο και ο kNN αποδίδουν εξίσου καλά. Βελτιώσεις πιθανόν να μπορούσαν να φανούν αν βρίσκαμε καλύτερο k στον kNN και χρησιμοποιούσαμε διαφορετικό kernel στο SVR. Με την κανονικοποίηση αποδίδουν ακόμη καλύτερα. Το Decision Tree από την άλλη κάνει overfit στα δεδομένα και κατά συνέπεια δεν είναι αποδοτικό. Ο λόγος που συμβαίνει αυτό είναι επειδή πιθανόν το βάθος του δέντρου να είναι μεγάλο. Προκειμένου να το αποφύγουμε αυτό μπορούμε να καταφύγουμε σε 2 τεχνικές. Η πρώτη είναι το pre-pruning όπου δημιουργούμε ένα δέντρο με λιγότερα κλαδιά από όσα θα έπρεπε (με καθορισμό για παράδειγμα ενός max depth) . Η δεύτερη είναι το post-pruning όπου δημιουργείται πρώτα το πλήρες δέντρο και έπειτα αφαιρούνται τμήματά του.

Πηγές

https://www.w3schools.com/python/numpy/numpy_random_poisson.asp
https://www.geeksforgeeks.org/numpy-random-poisson-in-python/
https://numpy.org/doc/stable/reference/random/generated/numpy.random.poisson.html
https://el.wikipedia.org/wiki/%CE%9A%CE%B1%CF%84%CE%B1%CE%BD%CE%BF%CE%BC%CE%AE_%CE%A0%CE%BF%CF%85%CE%B1%CF%83%CF%83%CF%8C%CE%BD
https://mathworld.wolfram.com/PoissonDistribution.html
https://www.geogebra.org/m/PUS7ZYW8
https://link.springer.com/chapter/10.1007/978-1-4471-4884-5_9
https://towardsdatascience.com/decision-trees-and-random-forests-df0c3123f991