Multiple Adaptive Regression Splines (MARS)

By - Divyansh Verma
Subject - Machine Intelligence (MI)
Roll no. - 16CO110
Email - divyanshverma12@gmail.com
Mobile - +91-8950882215

Definition - Multivariate/Multiple Adaptive Regression Splines (MARS) is a form of regression analysis which was introduced by Jerome H. Friedman in 1991. It is a stepwise linear regression algorithm. It can be defined as an attempt to modify linear models to automatically fit over non linearities in a given dataset. So in layman language it is an extension of linear models that can easily model some non linearities.

Terminology

1) *Multivariate* - *Able to generate model based on several input variables*
2) *Adaptive* - *Generates flexible models in passes each adjusting the model*
3) *Regression* - *Estimation of relationship among independent and dependent variables*
4) *Spline* - *A piecewise defined polynomial function that is smooth (possess high order derivatives) where polynomial pieces connect*
5) *Knot* - *The point at which two polynomial pieces connect*

Previous Methods There are various linear modeling techniques like linear regression (https://en.wikipedia.org/wiki/Linear_regression), logistic regression (https://en.wikipedia.org/wiki/Logistic_regression) etc.

Normal Regression

MARS

y' = -37 + 5.1x

y'= 25 + 6.1 max(0,x-13)-3.1 max(0,13-x)

Fig - 1 Image showing a comparison between Linear Regression and Multivariate Adaptive Regression Spline

They are really fast and simple algorithms and many of such linear models can be easily adapted to non linear patterns in the data by adding non-linear terms (like higher order polynomials, interaction effects or any other transformation techniques applied to original features), however to such things we should know the specific nature of the non-linearities and interactions before building such models.

There are many Data Analysis models which are naturally nonlinear and these models can be used to extract non linearity from the given dataset without detecting or identifying non-linearity in such datasets and Multivariate Adaptive Regression Spline (MARS) is one such algorithm (*Fig - 1* shows a comparison between Normal regression model and MARS model). MARS can discover non-linearities in a dataset without explicitly defining or understanding non-linearity (It will search for it).

Why to Use

We need to use such non-linear regression models (MARS) as they are more flexible than linear regression models and although some non-linearity is added to the model, yet the MARS model is easy to understand and interpret and also MARS requires minimal features engineering like feature scaling or feature transformation and automatically performs features selection.

Linear Regression is the most basic regression model. Simple linear regression (SLR) assumes that statistical relationship between two continuous variables (let us say X and Y) is linear and can be defined using a simple equation:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i \quad for\ i = 1, 2, 3, ..., n, \qquad (1)$$

Where Y_i represents the i-th prediction or value or X_i feature value and β_0 and β_1 are fixed but unknown constant and ε_i represent noise or error. So, a simple linear regression model work is to estimate values of β_0 and β_1 such that (1)'s value will have least loss or error sum on a test dataset or real life values. Cost or error sum can be defined in various ways one of the easy and most used formulas to calculate loss in linear regression is Residual sum of squares.

$Let\ \ Ypred_i\ be\ the\ predicted\ value\ from\ SLR\ given\ by\ -$

$$Ypred_i = \beta_0 + \beta_1 X_i \qquad for\ i = 1, 2, 3, ..., n,$$

$and\ true\ value\ given\ by\ Ytrue_i\ and\ the\ Loss\ function\ is\ given\ by\ -$

$$LOSS(\beta_0, \beta_1) = \sum_{i=1}^{n} [Ytrue_i - Ypred_i]^2$$

So, what linear regression does is that is find appropriate values of β_0 and β_1 to get minimum loss over the given data points. Such models can be easily extended for multidimensional data points.

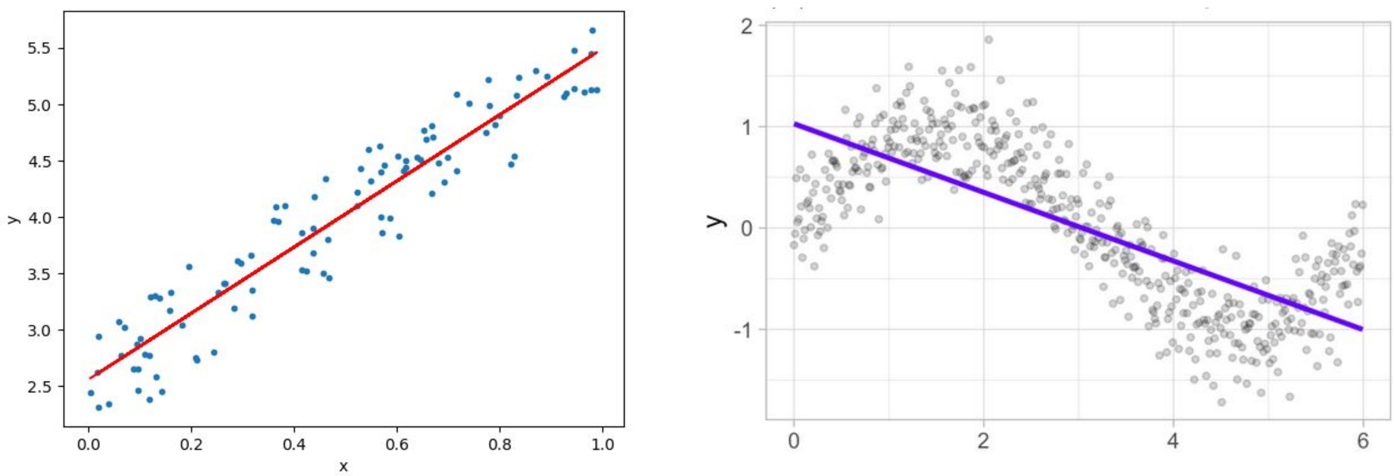

Fig - 2 Images showing output of linear regression on two different datasets

Problem with Logistic Regression - If you see fig - 2, when linear regression regression is applied on a dataset which is not linear (Fig - 2(b)), It under fits the datasets, so doesn't provide a good generalization of the dataset. Such predictions will have little or no use on non-linear distribution of data points. As discussed above there are many regression techniques like polynomial regression which can overcome and can fit over such distribution but for such regressions required pre-knowledge of such data points and give explicit parameters. But MARS doesn't require such explicit parameter initialization or pre-analysis of the dataset. It itself tries various configurations and tries to fit over the distribution.

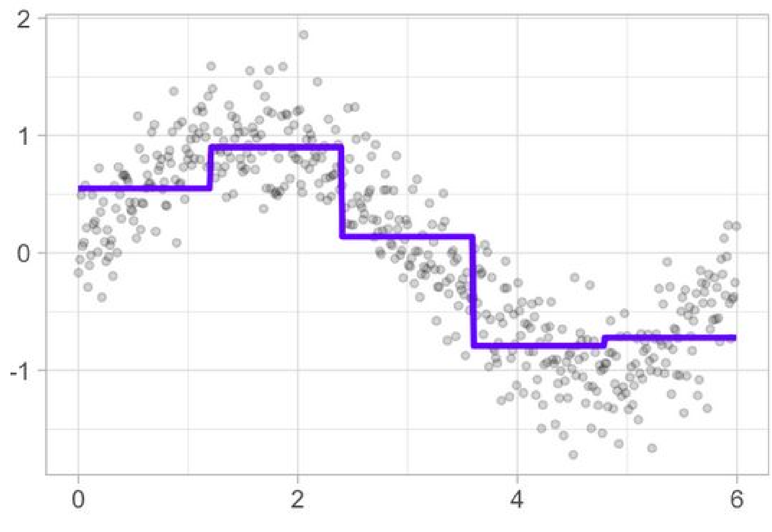

Fig - 3 Image shows fitting of a stepwise model over a non-linear distribution of data points

MARS uses piecewise linear basis functions of the form(given by an equation below)

$$y_i = \beta_0 + \beta_1 C_1(x_i) + \beta_2 C_2(x_i) + \beta_3 C_3(x_i) + ... + \beta_d C_d(x_i) + \varepsilon_i \quad (2)$$

Where $C_d(x_i)$ represents x_i values ranging from $c_{d-1} \leq x_i < c_d$.
Fig - 3 shows an illustration of such stepwise linear basis function

Multivariate adaptive regression splines (MARS) is an easy and simple approach to capture the non-linear relationships in the data by setting the values of knots(cutpoints) similar to step functions also known as hinge functions. The procedure assesses each data point for each predictor as a knot and creates a linear regression model with the candidate feature(s).

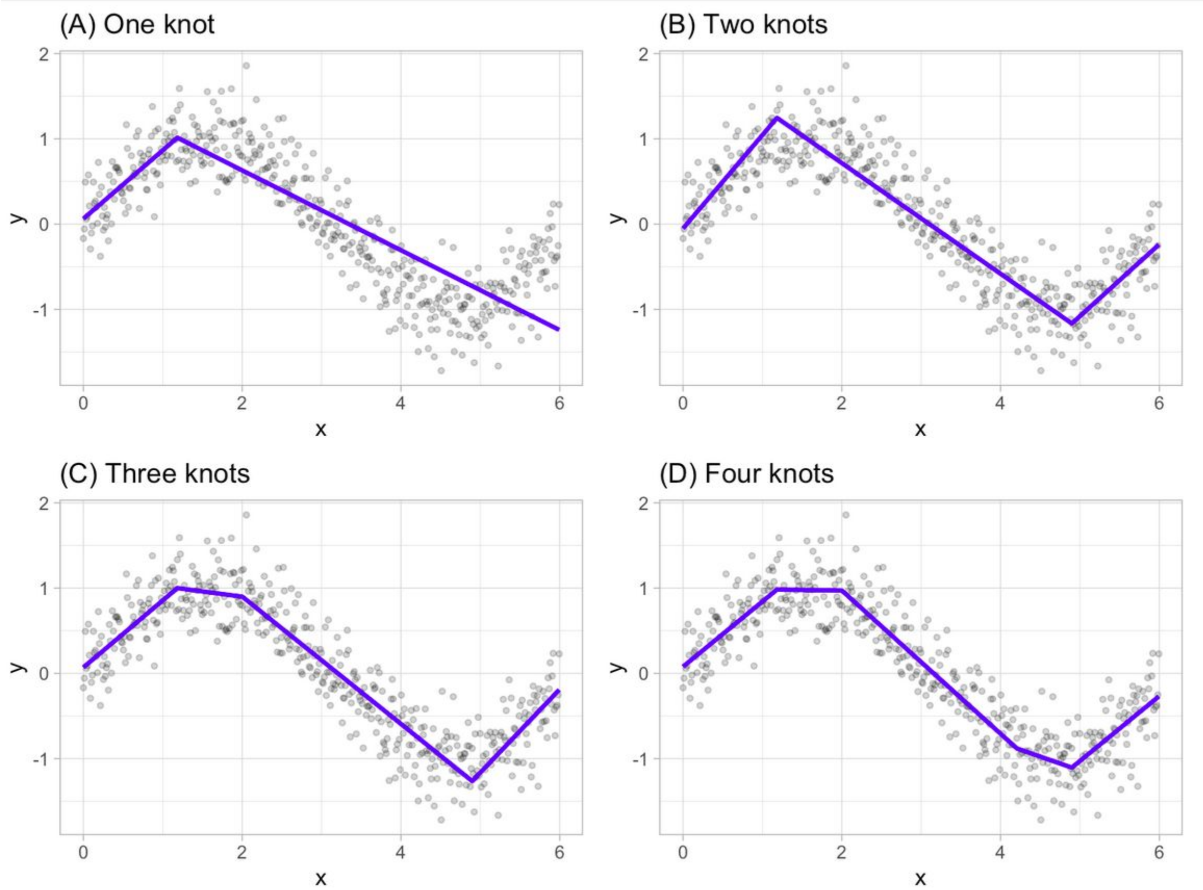

Fig - 4 Image showing fitted regression splines of one (A), two (B), three (C) and four (D) knots

Example/Overview of working of algorithm -

Consider a non-linear, non-monotonic dataset where $Y = f(X)$.

I. Look for the single point across the range of X values where 2 different linear relationships between Y and X achieve smallest error or loss.

II. The result of such finding is known as hinge which is given by $h(x-a)$ where a is the cut-point value.

As shown in Fig - 4(A) our hinge function is $h(x-1.18)$ such that out two linear model for Y will be -

$$Y = \beta_0 + \beta_1(1.18 - x) \; when\, x < 1.18$$
$$Y = \beta_0 + \beta_1(x - 1.18) \; when\, x > 1.18$$

III. Once the first knot is found, algorithm will continue to find 2nd knot which in the given figure fig - 4(B) is x = 4.89 so,

$$Y = \beta_0 + \beta_1(1.18 - x) \; when\, x < 1.18$$
$$Y = \beta_0 + \beta_1(x - 1.18) \; when\, x > 1.18 \text{ and } x < 4.89$$
$$Y = \beta_0 + \beta_1(4.89 - x) \; when\, x > 4.89$$

IV. Step III. continuous as long as many cutpoints(knots) are found, resulting in a good non-linear prediction equation.

Generalization

MARS model generalizes to $\Rightarrow$ $f(X) = \beta_0 + \sum_{m=1}^{M} \beta_m f_m(X)$ (3)

Where $f_m(X)$ is a basis function which is the product of two or more such hinge functions.

Basis function - Each basis function takes one of the three form

- A constant term
- A hinge function which has a form max(0,x-constant) or max(0,constant-x)
- A product of two or more hinge functions

β_i's for i = 0,1,2…,m are the coefficients of hinge functions estimated by minimizing the loss or error function (like defined in (1) above) and these coefficient can be defined as the weights that represent the importance of the variable in the MARS model to fit over a non-linear distribution of data-points.

MARS Model Building Procedure

1. Gather data i.e. x input variables or data-points from the dataset with y output for each x (i.e. input variable).
2. Calculate or find a set of basis functions by setting knots at observed values.
3. Constraint specification i.e. number of terms in the model and maximum allowable degree of interaction.
4. Forward Pass - Try out different or new hinge functions and their product combinations which decreases training error.
5. Backward Pass - Fix Overfitting over the training set.
6. Use of generalized cross validation technique to estimate the number of optimal terms in the MARS model.

MARS Forward Pass

1. MARS starts with a model which consists of an intercept term which can be defined as the mean of the response values.
2. Each step MARS adds a basis function in pairs to the model and finds a pair of basis functions that gives the maximum reduction in loss or error (i.e. sum of square error).
3. Each new basis function consists of a term already in the model multiplied by a new hinge function. As define above hinge function is defined by a variable and a knot so to add a new basis function, and MARS model search over all the combination of following
    a. Existing terms
    b. All variables
    c. All values of each variable
4. To calculate the coefficient of each term MARS applies a linear regression over the terms.

MARS Backward Pass

1. Forward Pass leads to an overfitted model (An overfitted model is a model that gives good accuracy on a test dataset used to build a model but does not generalize well to new data or real world data.
2. So to make a better model, pruning is used which is a major functionality of backward pass.
3. It removes one term at a time from the model.
4. Remove the term which increases the error or loss by minimum amount.
5. Continue removing terms until cross validation is satisfied. The MARS model uses Generalized Cross Validation (GCV).

Generalized Cross Validation

MARS backward pass uses generalized cross validation (GCV) for comparing the output/accuracy of model's subsets in order to choose the best subset. GCV is a form of regularization i.e. it trades off goodness of fit against model complexity (As used in various neural network models). GCV is used to approximate the error or loss that will be there by removing one hinge function or a set of that.

There is nothing wrong in having a lot of hinge functions but a model that fits to noise in the dataset can give poor results on real world data.

Formula of GCV =

$$\left(\sum_{i=1}^{n} [Y\,true_i - Y\,pred_i]^2 \right) / \left(N * (1 - (effective\ number\ of\ parameters)/N)^2\right)$$

The effective number of parameters is defined in MARS context as

$$(effective\ number\ of\ parameters) = (number\ of\ mars\ terms) + (penalty) * ((number\ of\ mars\ terms) - 1)/2$$

Where penalty can be set to 2 or 3 by the analyst or programmer.

Assumptions

No assumptions are made about the environment or distribution of data-points. The only requirement for the MARS model to perform well is that variables should not be highly correlated to one another as this can lead to difficulty in estimation.

Advantages of MARS

1. Automatically detects interactions between variables.
2. Fast and computationally efficient.
3. Easy to handle data with high dimensions.
4. Non-linear relationships are handled well.
5. More Flexible than linear models.
6. Simple to understand and interpret.
7. Both continuous and discrete data can be handled well.
8. Requires no data preparation.
9. As computationally fast, can handle large datasets.

Output of MARS on a dataset

Given dataset can be downloaded from this link (Dataset). When plotted the dataset distribution looks like this

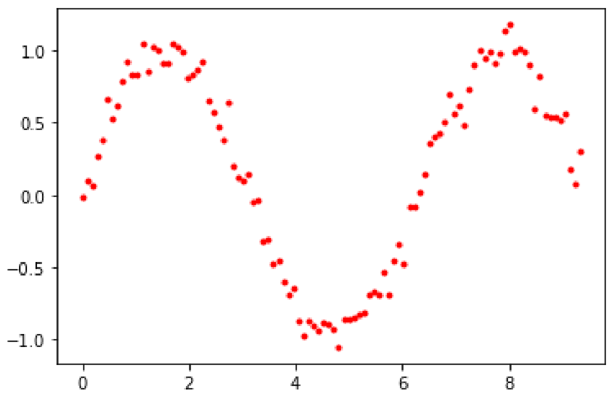

Fig - 6 Image showing Dataset distribution

Now when the MARS model is trained on this dataset with different parameters such as max_degree(i.e. Maximum degree of x in the equation (3)) and max_terms (i.e. Maximum number of allowed hinge functions), we can get different outputs and those outputs were plotted and examined.

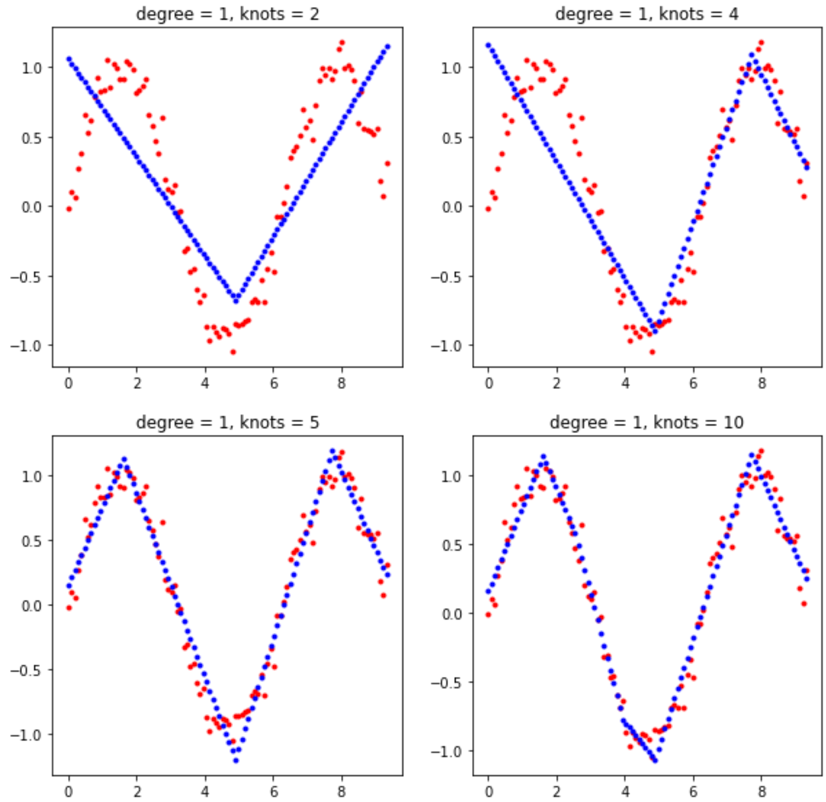

Fig - 7 Image shows 4 different output of MARS model with max_degree = 1 and different values of knots allowed

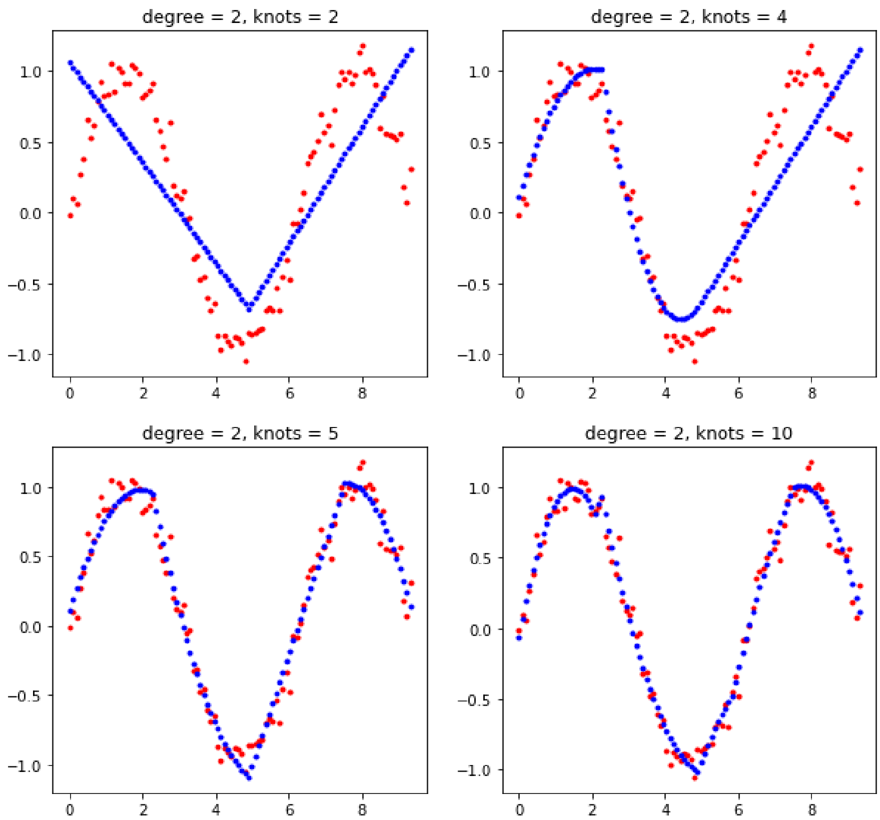

Fig - 8 Image shows 4 different output of MARS model with max_degree = 2 and different values of knots allowed

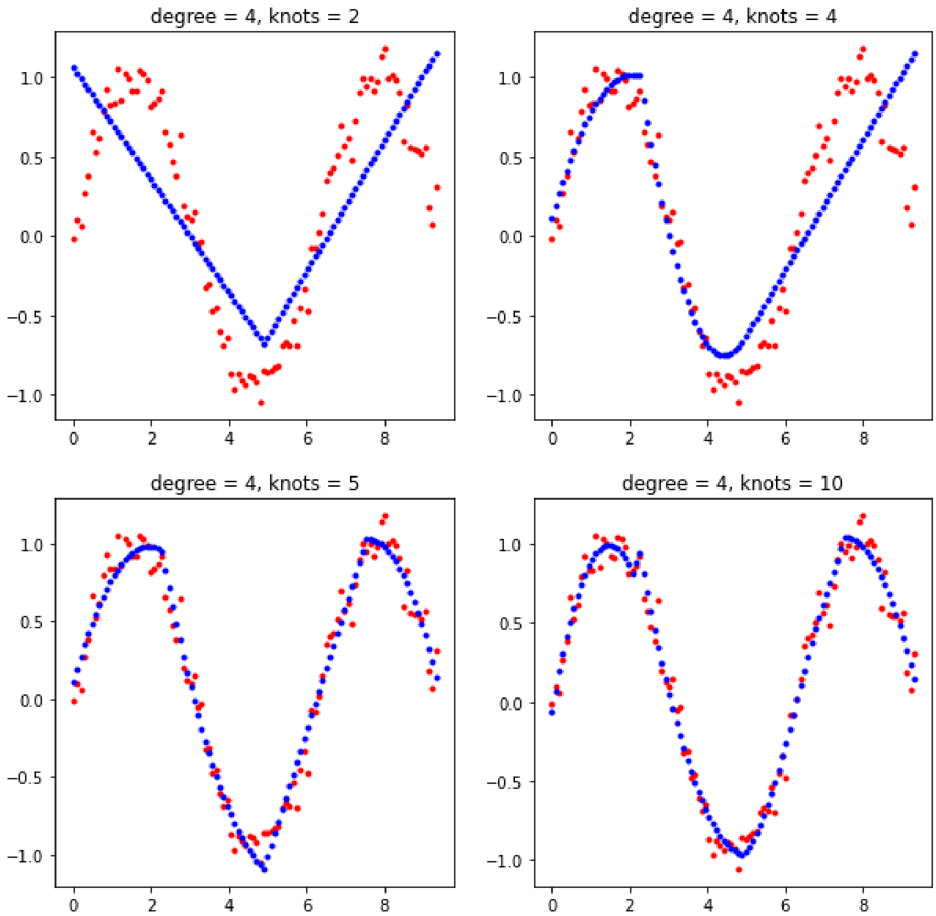

Fig - 9 Image shows 4 different output of MARS model with max_degree = 4 and different values of knots allowed

From Fig - 7,8,9 we can examine that increasing the number of knots helps in better fitting of data distribution and increasing degree brings smoothness in the model's prediction (i.e. helps in fitting curves in the data distribution). By comparing all the models above (Fig 7,8,9) it can be found that the model with knots = 10 and degree = 4 fits the dataset best.

Basis Function	Pruned	Coefficient
(Intercept)	No	1.07464
h(x0-4.90088)	No	1.47705
h(4.90088-x0)	Yes	None
h(x0-2.26195)*h(4.90088-x0)	No	-0.245643
h(2.26195-x0)*h(4.90088-x0)	No	-0.487797
h(x0-7.53982)*h(x0-4.90088)	No	-0.283587
h(7.53982-x0)*h(x0-4.90088)	Yes	None
h(x0-2.07345)	No	-0.719911
h(2.07345-x0)	No	2.06014
h(x0-6.78584)*h(7.53982-x0)*h(x0-4.90088)	Yes	None
h(6.78584-x0)*h(7.53982-x0)*h(x0-4.90088)	No	-0.119208

Fig - 10 Image showing final MARS model output for dataset with max_degree = 3 and knots(max_terms) = 4

MARS with Logistic Regression

Mars Model can be used with logistic regression to compute non-linear boundaries. Here are the examples on three different types of dataset.

Steps -

1) Get a nonlinear equation output from the MARS model.
2) Apply logistic regression for decision boundaries.

Example 1 - (Simplified Iris dataset (petal length and sepal length)

Basis Function	Pruned	Coefficient
(Intercept)	No	5.20613
h(x1-6.1)	Yes	None
h(6.1-x1)	No	-0.893801
x1*h(6.1-x1)	No	-0.0950754
x0	No	-0.42067
x0*h(6.1-x1)	No	0.0730401
x0*x0	Yes	None

Fig - 11 Showing output of MARS model

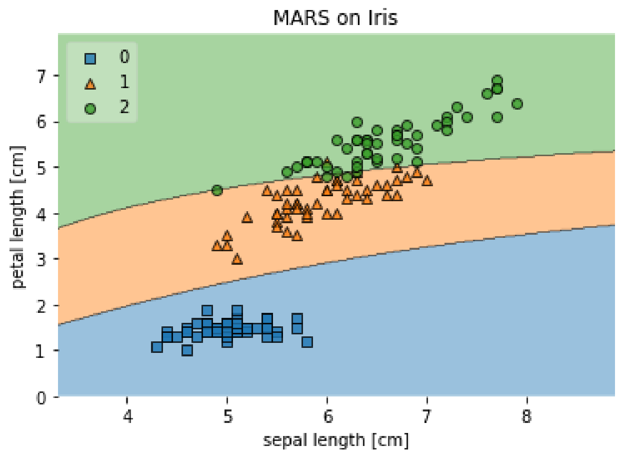

Fig - 12 showing output of logistic regression On equation given by MARS model

Example 2 - (make-moons dataset)

Basis Function	Pruned	Coefficient
(Intercept)	No	1.07715
x1	No	-0.689925
h(x0+0.518393)	No	-0.407693
h(-0.518393-x0)	No	-1.50049
h(x0-0.981559)*h(x0+0.518393)	No	0.642912
h(0.981559-x0)*h(x0+0.518393)	No	1.14643
h(x1-0.0640702)*x1	No	-0.714748
h(0.0640702-x1)*x1	No	-3.74363
h(x0-1.6723)*h(0.0640702-x1)*x1	No	-32.0155
h(1.6723-x0)*h(0.0640702-x1)*x1	No	1.93279
h(x1+0.458668)*h(x0+0.518393)	No	-0.107631
h(-0.458668-x1)*h(x0+0.518393)	No	3.54922
h(x1-0.5)	Yes	None
h(0.5-x1)	No	-0.789851

Fig - 13 Showing output of MARS model

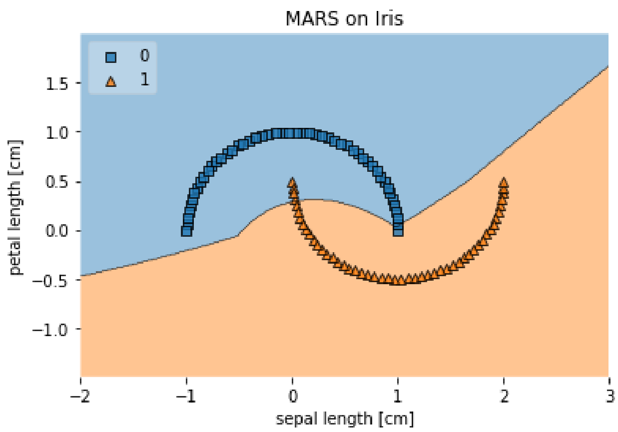

Fig - 14 showing output of logistic regression On equation given by MARS model

Implementation (Python)

- MARS model is present in python pyearth library under name **EARTH.** It can be imported using this piece of code. Pyearth library can be downloaded from this link - https://pypi.org/project/sklearn-contrib-py-earth/ and to learn more about this library - https://contrib.scikit-learn.org/py-earth/

```
from pyearth import Earth
```

- To test and train with different parameters, we can define different knots (hinge functions) -

```
#Number of different
#hinge functions allowed
knots = [2,4,5,10]
```

- Creating a simple MARS model using pyearth library with parameters such as max_terms(i.e. Maximum allowed different hinge functions), max_degree(i.e. Highest degree of polynomial function allowed) and verbose(which when set to 1 gives complete detail how our model learns different beta's ($\beta's$)).

```
model = Earth(max_terms=10,max_degree=4,verbose=0)
```

- Now to train or fit ours MARS model, we just need to write 1 function

```
model.fit(X, y)
```

- To know the parameters that MARS model learns while training on a dataset use model.summary() function.
- To trace pruning of different functions use model.trace().

```
#Summary of model
print(model.summary())

#How model learns
print(model.trace())
```

- *Complete code can be found here on the Google Colab Link -* [**https://colab.research.google.com/drive/1G-QeE9Fcr2qOaWimspiMTQdKfUrHyktd?usp=sharing**](https://colab.research.google.com/drive/1G-QeE9Fcr2qOaWimspiMTQdKfUrHyktd?usp=sharing)

```
--------------------------------------------------
Basis Function                 Pruned  Coefficient
--------------------------------------------------
(Intercept)                    No      -1.01476
h(x0-4.90088)                  No      0.869125
h(4.90088-x0)                  Yes     None
h(x0-2.26195)*h(4.90088-x0)    No      -0.840014
h(2.26195-x0)*h(4.90088-x0)    No      0.463652
h(x0-6.78584)*h(x0-4.90088)    No      -0.180146
h(6.78584-x0)*h(x0-4.90088)    No      -0.270381
h(x0-1.13097)*h(4.90088-x0)    No      0.634655
h(1.13097-x0)*h(4.90088-x0)    No      -0.753823
h(x0-7.91681)*h(x0-4.90088)    No      -0.111945
h(7.91681-x0)*h(x0-4.90088)    Yes     None
--------------------------------------------------
MSE: 0.0098, GCV: 0.0157, RSQ: 0.9786, GRSQ: 0.9665
Forward Pass
---------------------------------------------------------------
iter  parent  var  knot  mse       terms  gcv    rsq    grsq
---------------------------------------------------------------
0     -       -    -     0.459042  1      0.468  0.000  0.000
1     0       0    52    0.194660  3      0.220  0.576  0.530
2     2       0    24    0.100973  5      0.127  0.780  0.728
3     1       0    72    0.012693  7      0.018  0.972  0.962
4     2       0    12    0.010991  9      0.018  0.976  0.962
5     1       0    84    0.009803  11     0.018  0.979  0.962
---------------------------------------------------------------
Stopping Condition 0: Reached maximum number of terms

Pruning Pass
----------------------------------------------
iter  bf  terms  mse   gcv    rsq     grsq
----------------------------------------------
0     -   11     0.01  0.019  0.978   0.960
1     10  10     0.01  0.017  0.979   0.964
2     2   9      0.01  0.016  0.979   0.966
3     9   8      0.01  0.017  0.976   0.965
4     6   7      0.02  0.021  0.967   0.955
5     3   6      0.09  0.125  0.797   0.734
6     5   5      0.19  0.243  0.581   0.481
7     7   4      0.25  0.301  0.451   0.357
8     8   3      0.35  0.392  0.246   0.163
9     4   2      0.40  0.428  0.132   0.086
10    1   1      0.46  0.468  -0.000  -0.000
```

Fig - 15 Image showing sample output of model.summary() and model.trace() functions .

Demo -

A demo of MARS created using which can be downloaded from here - Link. It is created in python using Tkinter GUI library. To download and play with it a ReadMe file has been attached in the github repo.

Here are some screenshots from demo………

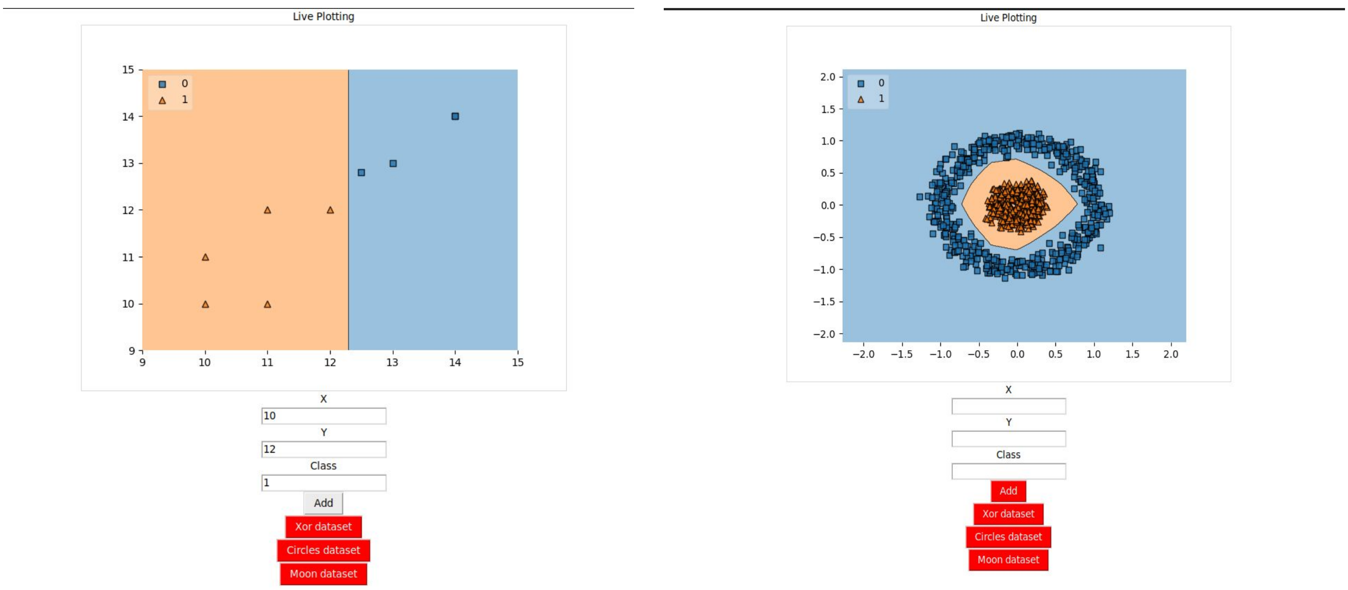

Fig -16 Two Images showing GUI and working of Demo created using Python Tkinter library.

Comparison With Other Non linear classifiers -

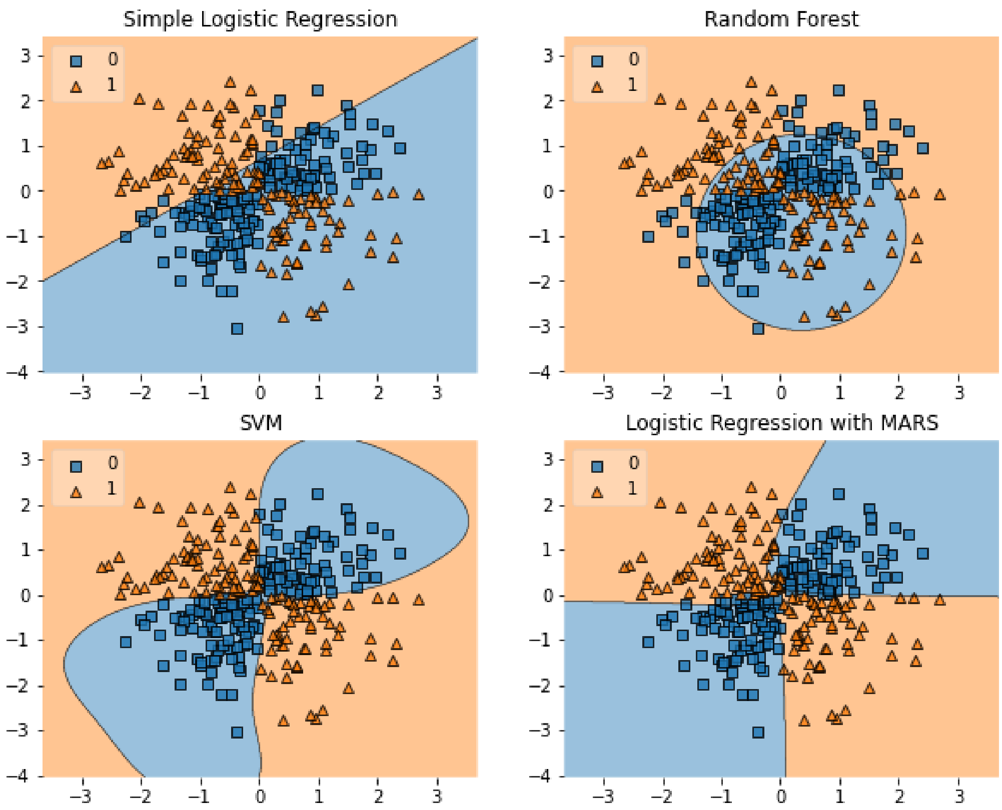

Fig - 14 Showing output of various model on a given XOR dataset

From Fig - 14 it can be seen that Logistic Regression with MARS clearly outperformed Simple Logistic Regression and Random forest and produces equivalent good results as SVM (yet MARS is a simple model than SVM)

Research Implementation Related Papers -

1. C. Briand and Bernd Freimut (2004). “Using multiple adaptive regression splines to support decision making in code inspections”. https://www.sciencedirect.com/science/article/pii/S0164121204000068
2. De Veaux, R.D., Psichogios, D.C., Ungar, L.H., 1993. A comparison of two nonparametric estimation schemes: MARS and neural networks. Computers Chemical Engineering 17 (8), 819–837.
3. Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines". *The Annals of Statistics*. **19** (1): 1–67. CiteSeerX 10.1.1.382.970. doi:10.1214/aos/1176347963. JSTOR 2241837. MR 1091842. Zbl 0765.62064. http://www.stat.yale.edu/~lc436/08Spring665/Mars_Friedman_91.pdf
4. Chi-Jie Lu ; Chih-Hsiang Chang ; Chien-Yu Chen ; Chih-Chou Chiu ; Tian-Shyug Lee “Stock index prediction: A comparison of MARS, BPN and SVR in an emerging market” https://ieeexplore.ieee.org/document/5373010
5. Wengang Zhang, Anthony T.C.Goh. “Multivariate adaptive regression splines and neural network models for prediction of pile drivability”. https://www.sciencedirect.com/science/article/pii/S1674987114001364
6. Prasenjit Dey, Ajoy K.Das. “Application of Multivariate Adaptive RegressionSpline-Assisted”. https://www.sciencedirect.com/science/article/pii/S1738573316300985

Other Links -

1. Github Repo - https://github.com/failedcoder12/MARS-Multivariate-Adaptive-Regession-Spline-
2. Graphs - https://colab.research.google.com/drive/1G-QeE9Fcr2qOaWimspiMTQdKfUrHyktd?usp=sharing
3. MARS Model - https://colab.research.google.com/drive/1sW2pCjWeoJKQ0YHLYl26kLRfTRm1iRHV?usp=sharing
4. GUI builder - https://colab.research.google.com/drive/1f8GPYn-Tz-hcKvVAw1MxOrBW55pfXDSP?usp=sharing

References

1. Friedman, J. H. (1991). "Multivariate Adaptive Regression Splines". *The Annals of Statistics*. **19** (1): 1–67. CiteSeerX 10.1.1.382.970. doi:10.1214/aos/1176347963. JSTOR 2241837. MR 1091842. Zbl 0765.62064.
2. https://bradleyboehmke.github.io/HOML/mars.html#final-thoughts-3
3. http://www.ideal.ece.utexas.edu/courses/ee380l_ese/2013/mars.pdf
4. https://support.bccvl.org.au/support/solutions/articles/6000118097-multivariate-adaptive-regression-splines
5. Milborrow S (2015) Notes on the earth package. http://www.milbo.org/doc/earth-notes.pdf
6. Trevor Hastie, Stephen Milborrow. Derived from mda:mars by, and Rob Tibshirani. Uses Alan Miller's Fortran utilities with Thomas Lumley's leaps wrapper. 2019. *Earth: Multivariate Adaptive Regression Splines*. https://CRAN.R-project.org/package=earth.
7. http://media.salford-systems.com/library/MARS_V2_JHF_LCS-108.pdf
8. Multivariate Adaptive Regression Splines. Wikipedia. http://en.wikipedia.org/wiki/Multivariate_adaptive_regression_splines
9. M. Nash and D. Bradford. Parametric and Nonparametric Logistic Regressions for Prediction of Presence/Absence of an Amphibian. EPA Oct. 2001. http:// www.epa.gov/esd/land-sci/pdf/008leb02.pdf.
10. The Elements of Statistical Learning (2nd ed.). Springer, 2009. http://www-stat.stanford.edu/~hastie/pub.htm.
11. TkInter - https://wiki.python.org/moin/TkInter
12. Mlxtend-http://rasbt.github.io/mlxtend/user_guide/plotting/plot_decision_regions/
13. GUI Creation - How to create a real-time plot with matplotlib and Tkinter
14. Pyearth library - https://pypi.org/project/sklearn-contrib-py-earth/
15. Pyearth documentation - https://contrib.scikit-learn.org/py-earth/