**An Open Problem in Constrained Support Vector Regression**

**Introduction:**

**Support-vector machines** (SVMs, also support-vector networks) are supervised learning models with associated learning algorithms that analyze data for classification and regression analysis. Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Vapnik et al., 1997), SVMs are one of the most robust prediction methods, being based on statistical learning frameworks or VC theory proposed by Vapnik and Chervonenkis (1974) and Vapnik (1982, 1995). In addition to performing linear classification and regression, SVMs can efficiently perform a non-linear classification and regression using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

In machine learning, kernel machines are a class of algorithms for pattern analysis, whose best-known member is the support-vector machine. The general task of pattern analysis is to find and study general types of relations (for example clusters, rankings, principal components, correlations, classifications) in datasets. For many algorithms that solve these tasks, the data in raw representation have to be explicitly transformed into feature vector representations via a user-specified feature map: in contrast, kernel methods require only a user-specified kernel, i.e., a similarity function over pairs of data points in raw representation.

Kernel methods owe their name to the use of kernel functions, which enable them to operate in a high-dimensional, implicit feature space without ever computing the coordinates of the data in that space, but rather by simply computing the inner products between the images of all pairs of data in the feature space. This operation is often computationally cheaper than the explicit computation of the coordinates. This approach is called the "kernel trick". In the case of Support Vector Regression, the choice of kernel comes into play when choosing the dot product to determine the distance between two points in the data.

Through the use of kernel functions, I will be presenting an open problem in Support Vector Regression with the necessary tools needed to solve such a problem. The problem will be to convert constrained Support-Vector Regression problems into standard SVR problems with an updated kernel choice which would make solving constrained optimization problems easier to solve, as currently every additional constraint leads to exponential increases in computation time. This problem was suggested by Ph.D. Chris Bemis of the University of Minnesota. Details on the importance of this problem with calculations are provided after the problem is described in full and the kernel trick is applied to make the problem feasible.

**Standard Support-Vector Regression:**

**Linear SVM Regression: Primal Formula**

Suppose we have a set of training data where xn is a multivariate set of N observations with observed response values yn.

To find the linear function

and ensure that it is as flat as possible, find *f*(*x*) with the minimal norm value (*β*′*β*). This is formulated as a convex optimization problem to minimize

subject to all residuals having a value less than ε; or, in equation form:

It is possible that no such function *f*(*x*) exists to satisfy these constraints for all points. To deal with otherwise infeasible constraints, introduce slack variables *ξn* and *ξ*\**n* for each point. This approach is similar to the “soft margin” concept in SVM classification, because the slack variables allow regression errors to exist up to the value of *ξn* and *ξ*\**n*, yet still satisfy the required conditions.

Including slack variables leads to the objective function, also known as the primal formula:

subject to:

The constant *C* is the box constraint, a positive numeric value that controls the penalty imposed on observations that lie outside the epsilon margin (*ε*) and helps to prevent overfitting (regularization). This value determines the trade-off between the flatness of *f*(*x*) and the amount up to which deviations larger than *ε* are tolerated.

The linear ε-insensitive loss function ignores errors that are within *ε* distance of the observed value by treating them as equal to zero. The loss is measured based on the distance between observed value *y* and the *ε* boundary, called the Hinge-Loss. This is formally described by

**Linear SVM Regression: Dual Formula**

The optimization problem previously described is computationally simpler to solve in its Lagrange dual formulation. The solution to the dual problem provides a lower bound to the solution of the primal (minimization) problem. The optimal values of the primal and dual problems need not be equal, and the difference is called the “duality gap.” But when the problem is convex and satisfies a constraint qualification condition, the value of the optimal solution to the primal problem is given by the solution of the dual problem.

To obtain the dual formula, construct a Lagrangian function from the primal function by introducing nonnegative multipliers *αn* and *α*\**n* for each observation *xn*. This leads to the dual formula, where we minimize

subject to the constraints

The β parameter can be completely described as a linear combination of the training observations using the equation below:

The function used to predict new values depends only on the support vectors:

The Karush-Kuhn-Tucker (KKT) complementarity conditions are optimization constraints required to obtain optimal solutions. For linear SVM regression, these conditions are

These conditions indicate that all observations strictly inside the epsilon tube have Lagrange multipliers = 0 and = 0. If either or is not zero, then the corresponding observation is called a support vector.

**Nonlinear SVM Regression: Primal Formula**

Some regression problems cannot adequately be described using a linear model. In such a case, the Lagrange dual formulation allows the previously-described technique to be extended to nonlinear functions.

|  |  |
| --- | --- |
| **Kernel Name** | **Kernel Function** |
| Linear (dot product) | *G*(*xj*,*xk*)=*xj*′*xk* |
| Gaussian | *G(xj*,*xk*)= |
| Polynomial | *G*(*xj*,*xk*)=(1+*xj*′*xk*)*q*, where *q* is in the set {2,3,...}. |

We obtain a nonlinear SVM regression model by replacing the dot product *x*1′*x*2 with a nonlinear kernel function *G*(*x*1,*x*2) = <*φ*(*x*1),*φ*(*x*2)>, where *φ*(*x*) is a transformation that maps *x* to a high-dimensional space. In MatLab, the Statistics and Machine Learning Toolbox provides the following built-in positive semidefinite kernel functions listed above.

The Gram matrix is an n-by-n matrix that contains elements gi,j = G(xi,xj)=K(*x*i,*x*j). Each element gi,j is equal to the inner product of the predictors as transformed by φ. However, we do not need to know φ, because we can use the kernel function to generate Gram matrix directly. Using this method, nonlinear SVM finds the optimal function f(x) in the transformed predictor space. TO AVOID CONFUSION I WILL BE USING K INSTEAD OF G TO REFER TO THE KERNEL NOTATION, BUT REFERENCE LINKS MAY USE G.

**Nonlinear SVM Regression: Dual Formula**

The dual formula for nonlinear SVM regression replaces the inner product of the predictors (*xi*′*xj*) with the corresponding element of the Gram matrix (*gi,j*).

Nonlinear SVM regression finds the coefficients that minimize

subject to the constraints

The function used to predict new values depends only on the support vectors:

The KKT complementarity conditions are:

**Additional Constraints:**

Now suppose we wish to apply monotonicity constraints the linear transformation *φ*(*x*) in the following way:

First, we must define monotonicity:

Let . Suppose a partial ordering  exists in the input space and a linear ordering is defined over the space R. The function f is said to be monotonic if the following expression holds: x y for any x and y; where the partial order, , is defined such that for x=(x1; x2; ...; xn), and y =(y1; y2; ...; yn), we say x y if and only if i for

i = 1,...,n.

Then we want our monotonicity constraints to be:

There is obviously no function that will achieve this on randomly disturbed data such as stock returns but we would like to minimize the violations.

To do this we want to minimize the following error function:

If we recall though, the we are trying to restrict to be monotonic actually has the form where K() is an inner product of the , thus we want to preserve monotonicity in the kernel which should be easier.

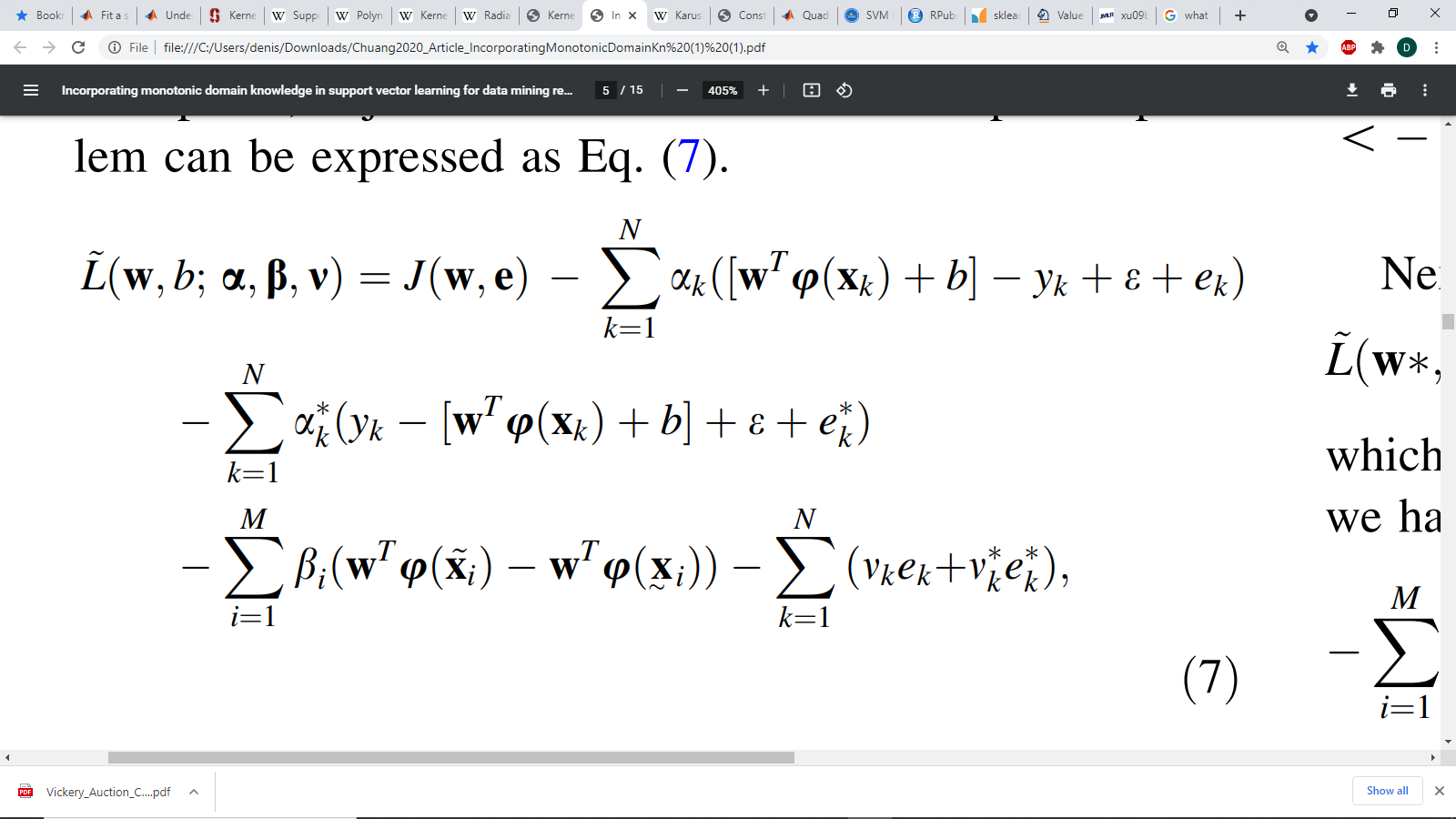
Without taking into account this notion of Kernels we have a solution to the linear Monotonicity Constrained SVR, derived in “Incorporating monotonic domain knowledge in support vector learning for data mining regression problems” [reference 1, see page 41] with the problem statement given as: Text

Description automatically generated

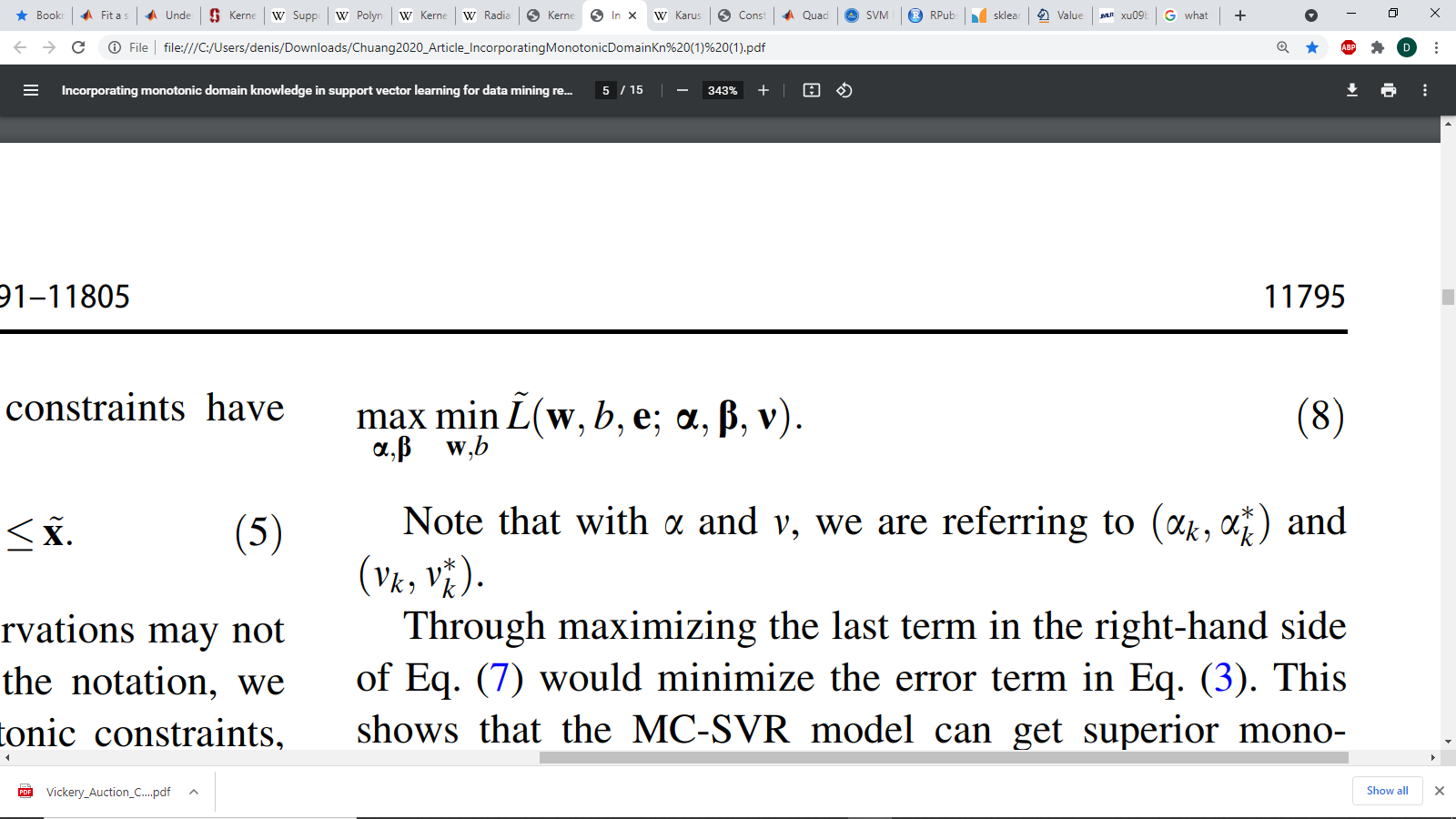
Note that they have used W instead of as used in my formulation of the problem.

Then the problem is trying to preserve the Monotonicity of M observations in the data for which an ordering can be place on them.

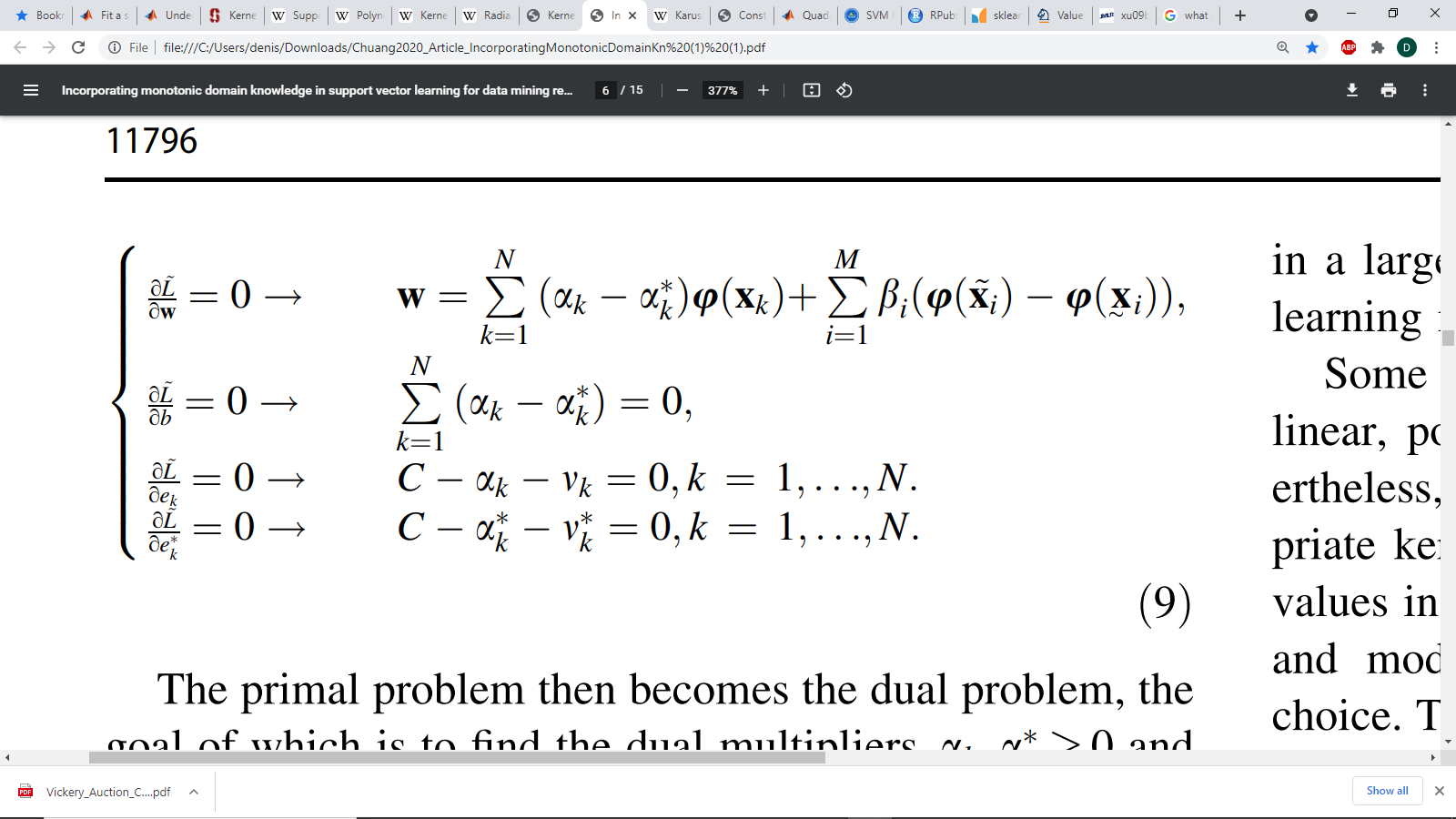
From this the Lagrangian becomes



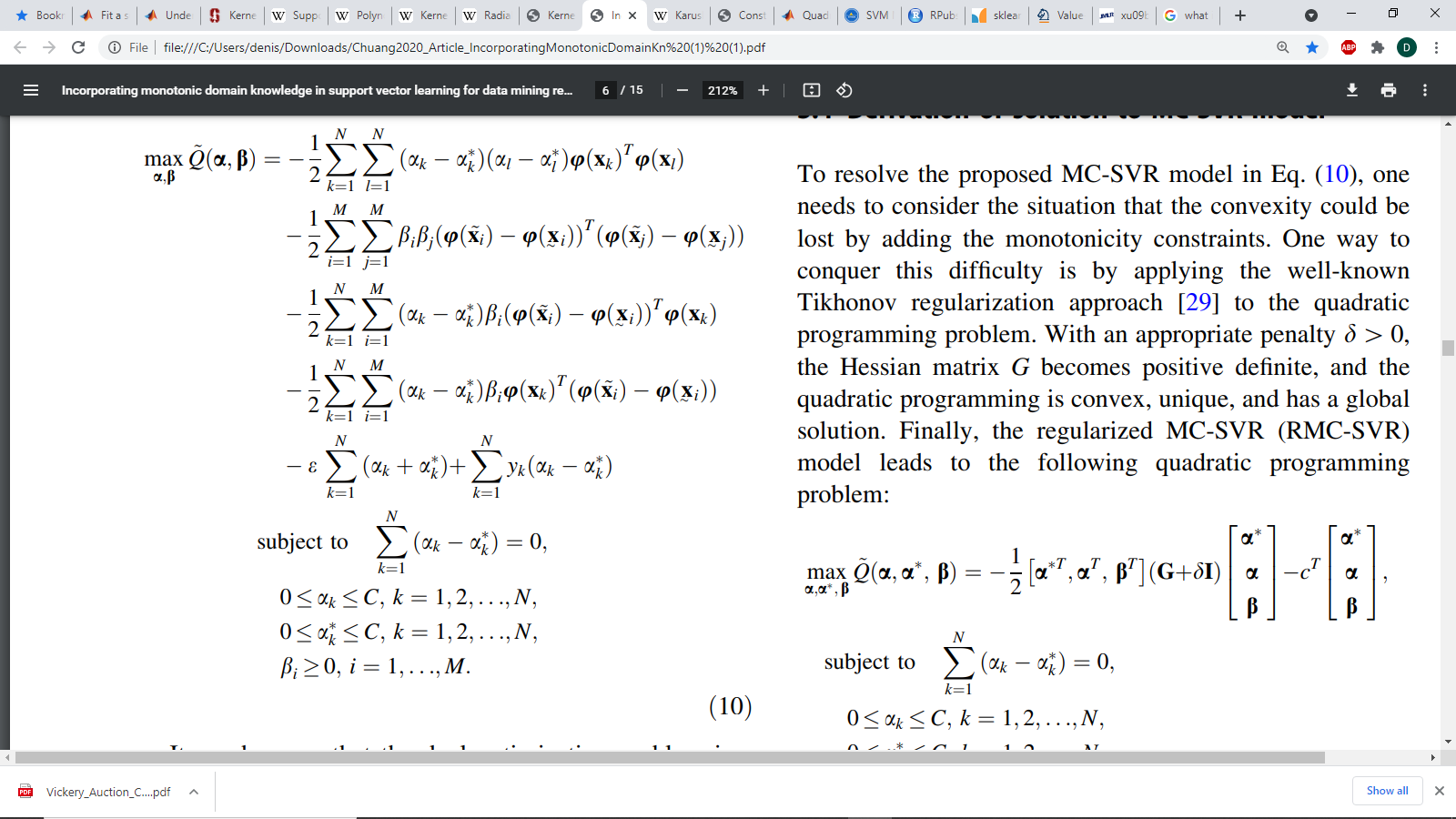
Where are Lagrange multipliers, and e is the monotonicity error.

And we seek 

Lastly applying the derivatives



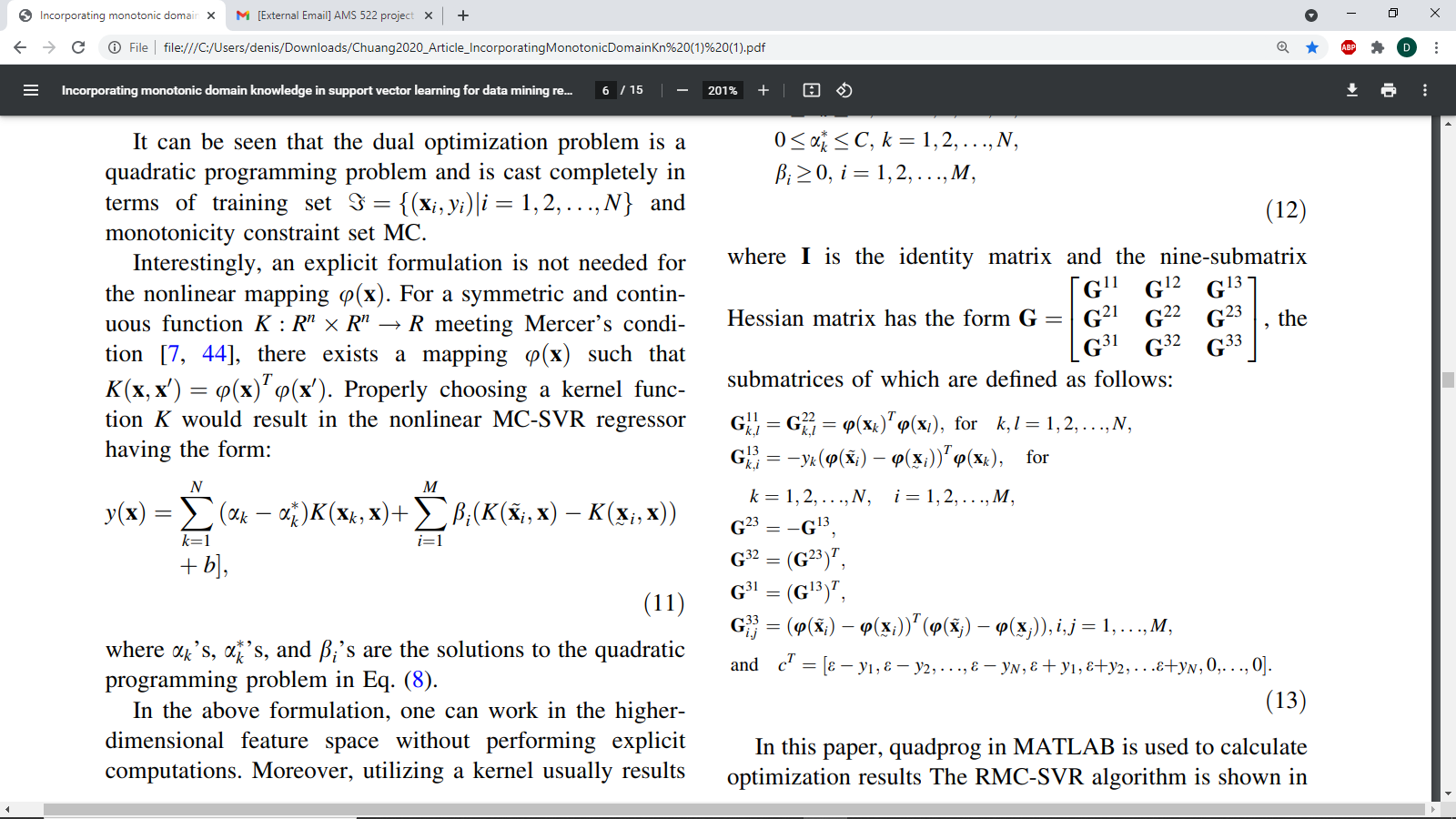
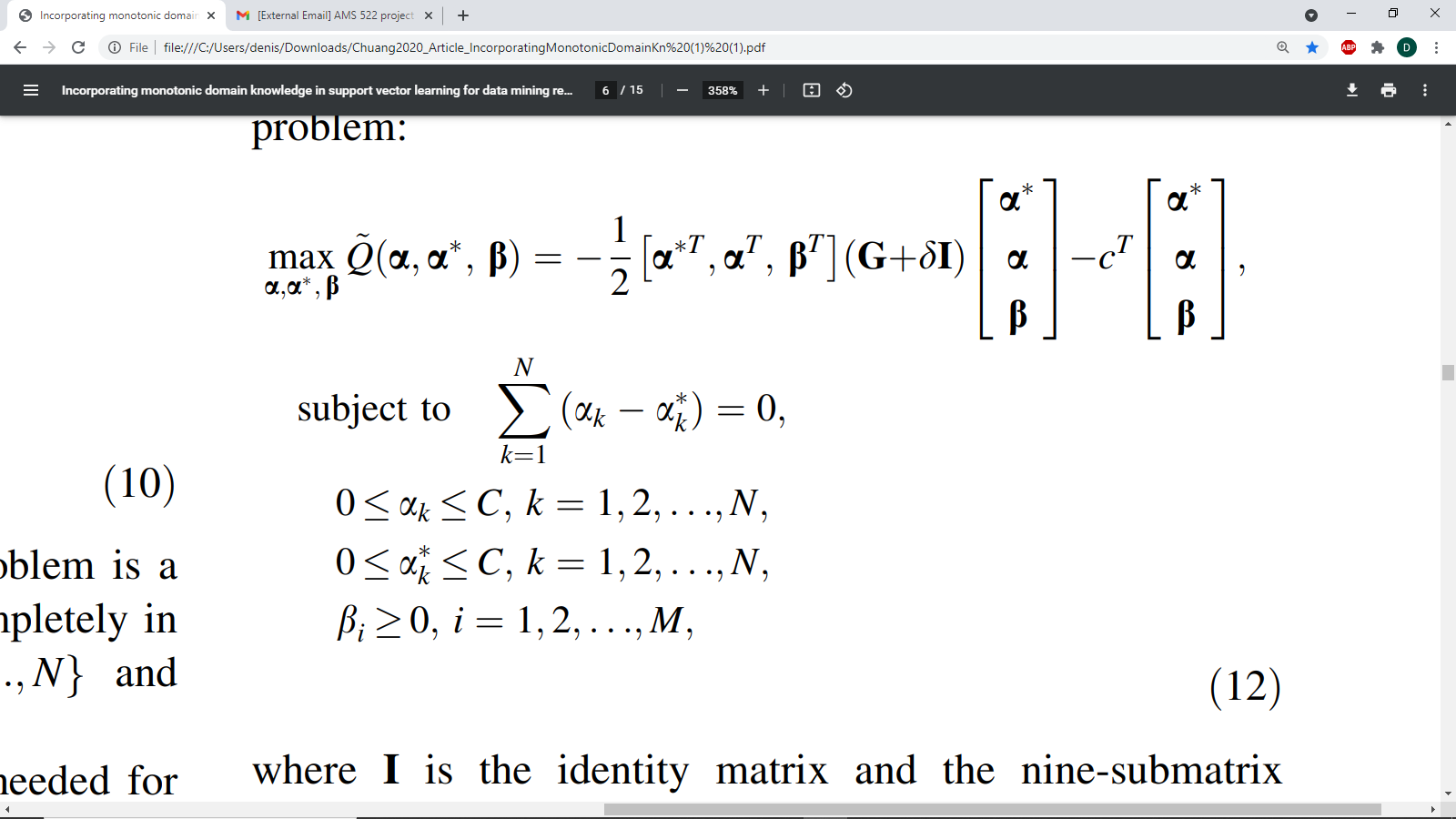
we get to the dual problem



Then we almost have this in a form where we can use the kernel trick!

We can see that under this formulation we can derive the problem with respect to kernel functions as:

Note that we were able to remove 1 one of the sums due to the symmetric property of inner products on Real vector spaces

This leads to the Kernel formulation of the problem: 

Note it was left off in the paper, but G12 = G21 = 0\*G11

Ie the first two off diagonal terms are 0 matricies of size NxN

**Problem Statement:**

We wish to find a new kernel inner product such that these new constraints are included in the original Support Vector Regression formulation. In other words, we would like to find a relation between the choice of kernel inner product function and the monotonicity constraints on the transformations. In a mathematical framework:

Given a transformation and an inner product *K*(*x*1,*x*2) = <*φ*(*x*1),*φ*(*x*2)>, can we find a new inner product K\*( *x*1,*x*2) = <*φ*(*x*1),*φ*(*x*2)>\* such that the solution to the Monotonicity Constrained SVM Regression optimization problem under inner product K is equivalent to the solution of the unconstrained Nonlinear SVM Regression optimization problem under inner product K\*.

To approach the problem, I will consider the case of starting with the most commonly used Kernel Inner Product Functions for Support Vector Regression which are as follows:

1. The Linear Kernel which uses the Euclidean Dot Product:
2. The Polynomial Kernel:

Where is a free parameter and d is the chosen degree of the polynomial Most often d=2 to avoid over fitting the model.

1. The Radial Basis Function (RBF) Kernel :

Where is the Euclidean distance between x & y and is a free parameter

For the purposes of this problem we will be using as the Normal RBF Kernel.

In the case of these 3 models we will see if there exists a new kernel such that the solution for the unconstrained model is equivalent.

We then must consider what transformation would produce such an output by the chosen kernel function.

1. For the linear kernel we have:

which leads to the simple conclusion that is the Identity function.

1. For the Polynomial Kernel we require some actual work:

To get the transformation we must expand this polynomial, for convenience I will only show this for d=2

Therefore,

Where h is the dimension of the output (h is called the hyper dimension) .

1. For the Normal RBF Kernel:

By Taylor series expansion of the first exponential function we get

There is no closed form for the transformation in this case as the dimension of the feature space we are mapping into is infinite dimensional. (This should have been obvious as the RBF can be thought of as a polynomial kernel of dimension d= when expanded as a Taylor Series.

For this reason, the RBF kernel is in practice approximated using a transformation z(x) such that z is estimated by randomly sampling from the Fourier transformation of the kernel. Another approach uses the Nyström method to approximate the eigen decomposition of the Gram matrix G=K, using only a random sample of the training set. In either case due to the inability to actually determine the true function makes the problem we are seeking to solve impossible in this case as we cannot make a meaningful transformation of the kernel based on the constraints on if we do not know . However if we remain in the kernel form we avoid this issue entirely, but to consider the relation between Kernels, one may need to be able to make this jump back to

This is not great as in practice the RBF kernel out-performs the Polynomial kernel on new data, but although this will not work, we can get an idea of how to do such a transformation, by looking at the polynomial kernels due to the aforementioned relation between the two kernels by the Taylor Series expansion.

Lastly, we will look into the MC-SVR problem under these choices for to see how they relate to the original unconstrained problem

**Problem Approach:**

To approach the problem we must now consider the resulting problem statement for the monotonicity constrained problem for each case.

Case 1 Linear Kernel:

subject to the constraints

Note that the first line is just the normal Linear SVR problem.

The problem is to then find a new kernel which incorporates the following two sums in the first sum, with new values for the & incorporating the values of the so that the monotonicity constraint may be removed without losing the property of monotonicity.

Case 2 Polynomial Kernel of dimension d=2 with c=0:

For the unconstrained problem we arrive at the following problem statement:

replacing the with the function it represents into the nonlinear SVR problem resulting in:

subject to the constraints

Note that the first three lines represent the sum

The unconstrained problem would thus require writing out:

subject to the constraints

To expand this out fully would clearly become a nightmare to write down and it is clear why the kernel trick was developed.

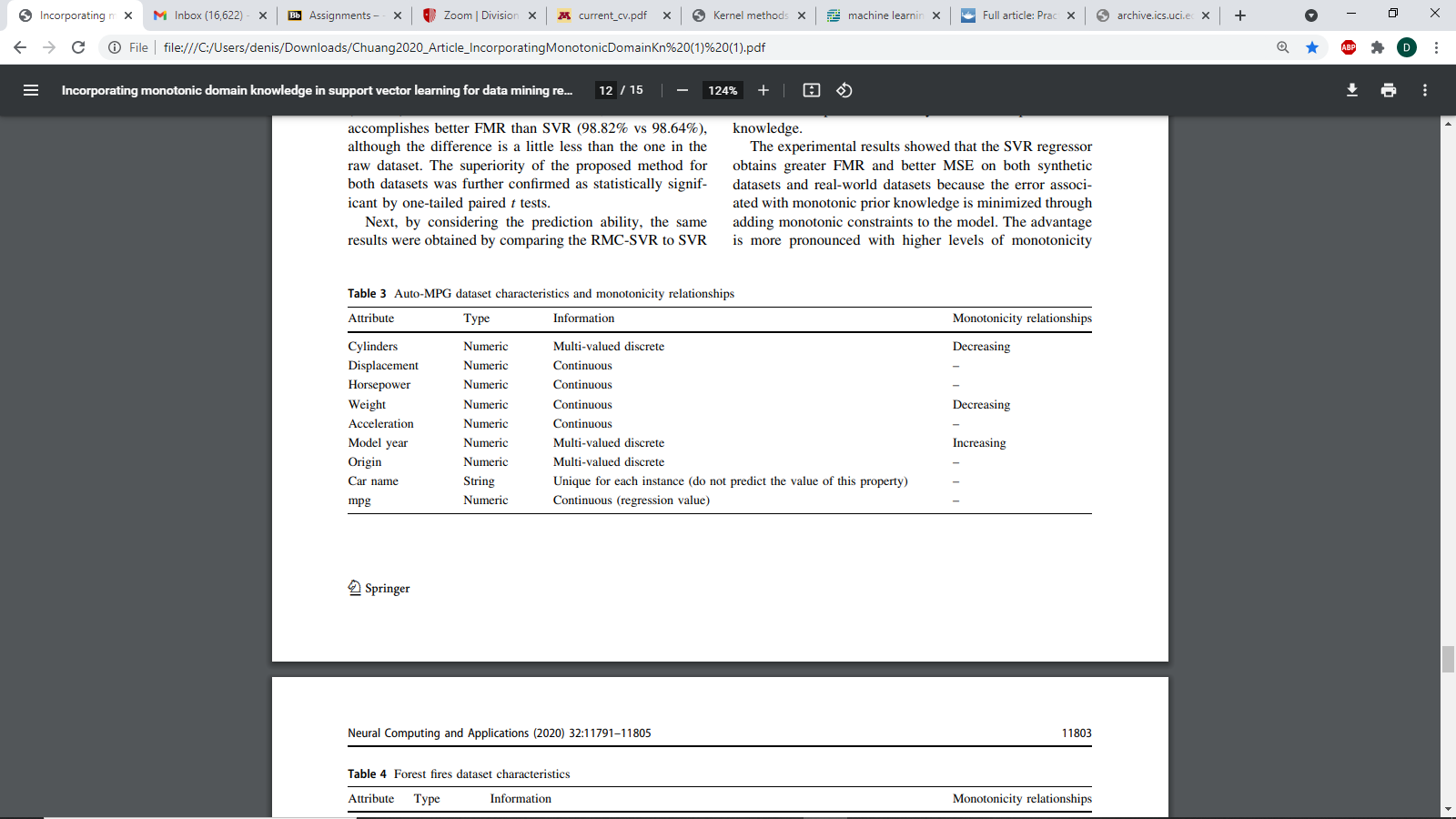
From this we can see that although the problem looked very close to a solution in the outset, there is still a gap in the mathematical framework in this area to make a conclusive connection between the unconstrained SVR problem and the Monotonicity problem based on transformations to the kernel. If one seeks to embark on solving this problem themselves, I recommend the following paper:

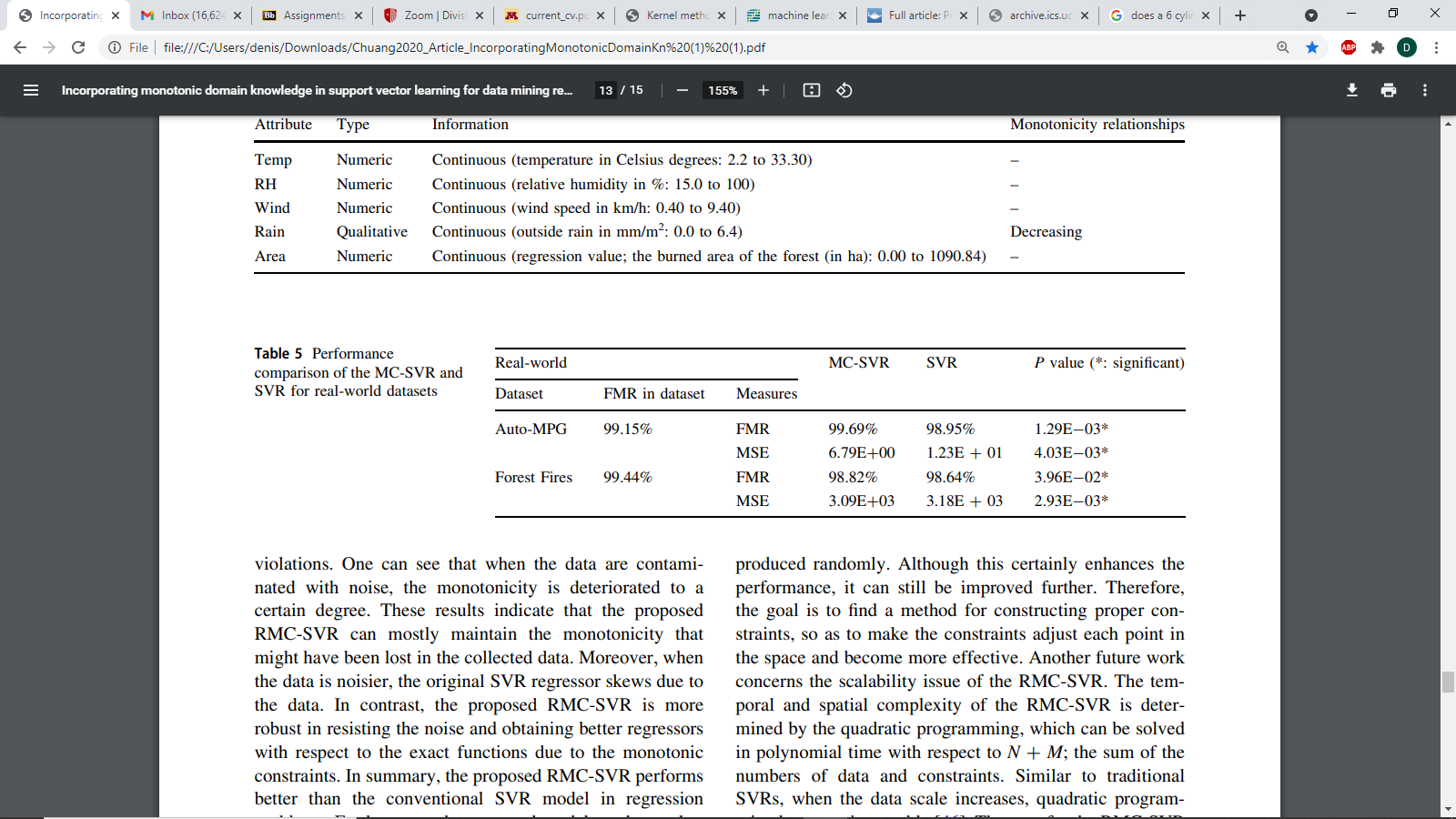
“Kernel Methods In Machine Learning” By Thomas Hofmann, Bernhard Schölkopf

And Alexander J. Smola, which focuses on SVR for the majority of the paper (however it also discusses Principal Component Analysis toward the end) and formulates the useful properties of kernel functions and the kernel trick.

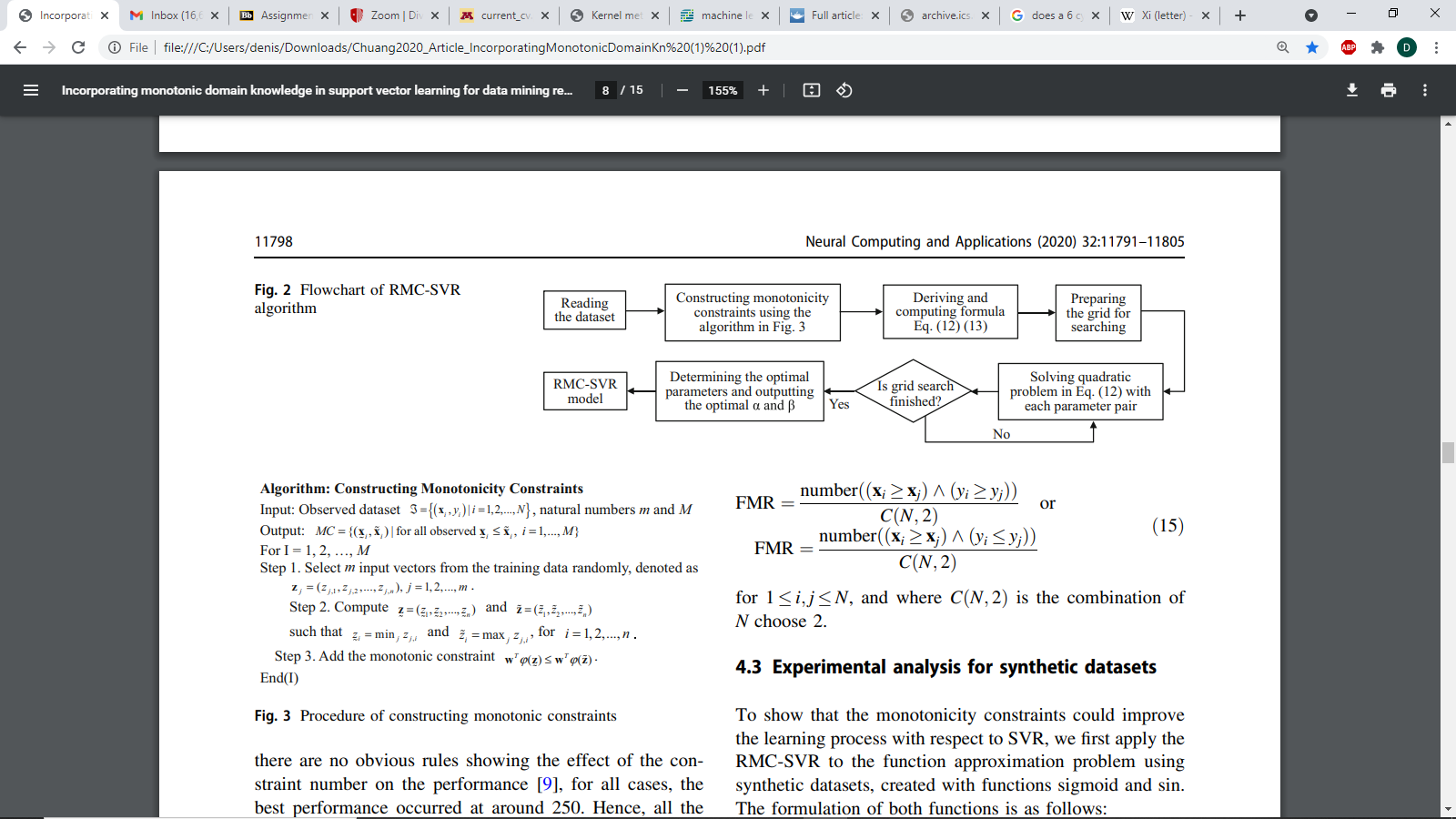
**Importance of Monotonicity and this Problem in General:**

In the paper “Incorporating monotonic domain knowledge in support vector

learning for data mining regression problems” which this problem is based on it is shown that monotonicity constrained SVR problems provide better estimates for nonlinear relations between data and response such as sinusoidal relationships between data and response as shown in the paper using the MSE as a measurement of model accuracy on synthetic data. The reason for this is that we are attempting to preserve the monotonicity of the sinusoidal function on specific intervals, which help to determine the true relationship, such as changes in slope. More importantly however, we are attempting to incorporate prior knowledge about a relationship between the data and the response variable so that these relationships are preserved in our model. In other words, we are attempting to build a model taking a Bayesian approach to preserve previous insight on the data. Currently, Support Vector Regression is regarded as a highly successful regression technique, however incorporating previously known relations in data into models is not so easy. The example given in the paper is, if two identical houses on the market are being appraised, then all else held the same, the house closer to the city should cost more. Of course it is rare that all other variables will be held equal and as such it is possible that our model will miss this fact if there if the only houses we have in our data set which are closest to the city also happen to be located next to the loud, busy, and dangerous highway leading into the city. For an example with real data, they showed that on the AutoMPG dataset from the UCI Machine Learning Repository, training a monotonicity constrained SVR model led to better predictions on the true MPG of a car using the following relations: 

than an unconstrianed SVR model with the following results: 

Where FMR is a measure of monotonicity violations as follows.

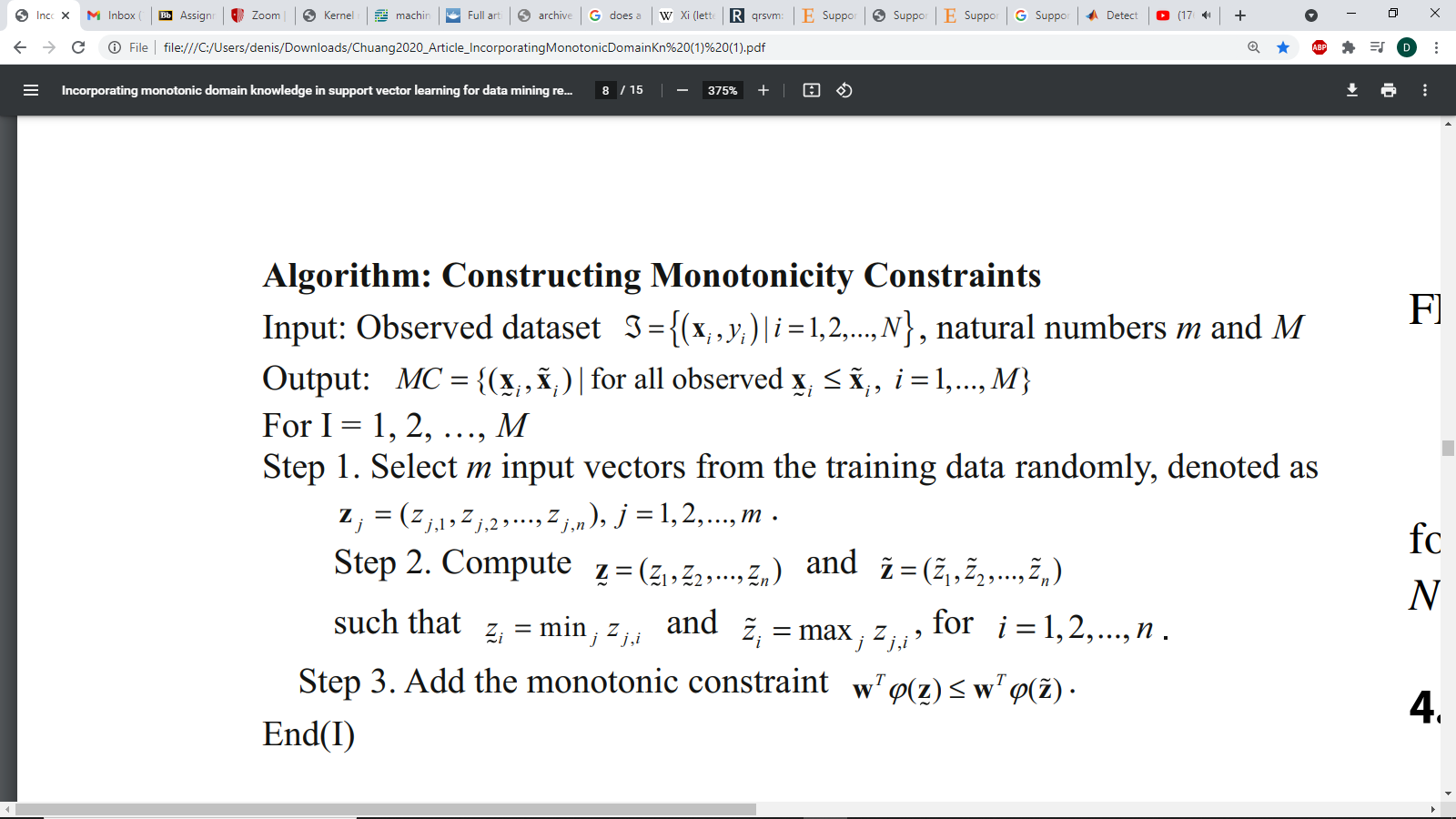
For a dataset a pair of instances and are said to not violate monotonicity, if and only if Then, FMR is defined according to all possible combinations of data which have monotonicity, and for an increasing or a decreasing function, it is defined, respectively, as: 

Based on these results and the generalization of this problem to allowing more prior knowledge, besides just monotonicity constraints, to help strengthen a regression model developed via SVR makes a solution to this problem useful. Currently, Monotonicity constraints increase time complexity for a solution as put in the paper “Another future work concerns the scalability issue of the RMC-SVR. The temporal and spatial complexity of the RMC-SVR is determined by the quadratic programming, which can be solved in polynomial time with respect to N + M; the sum of the numbers of data and constraints. Similar to traditional SVRs, when the data scale increases, quadratic programming becomes intractable” thus applying a multiple monotonicity constraints is not feasible in most cases as the problem is not parallelized and thus the solution will take up extensive amounts of memory and time to compute.

**Monotonicity Constraints Applied to Stock Data:**

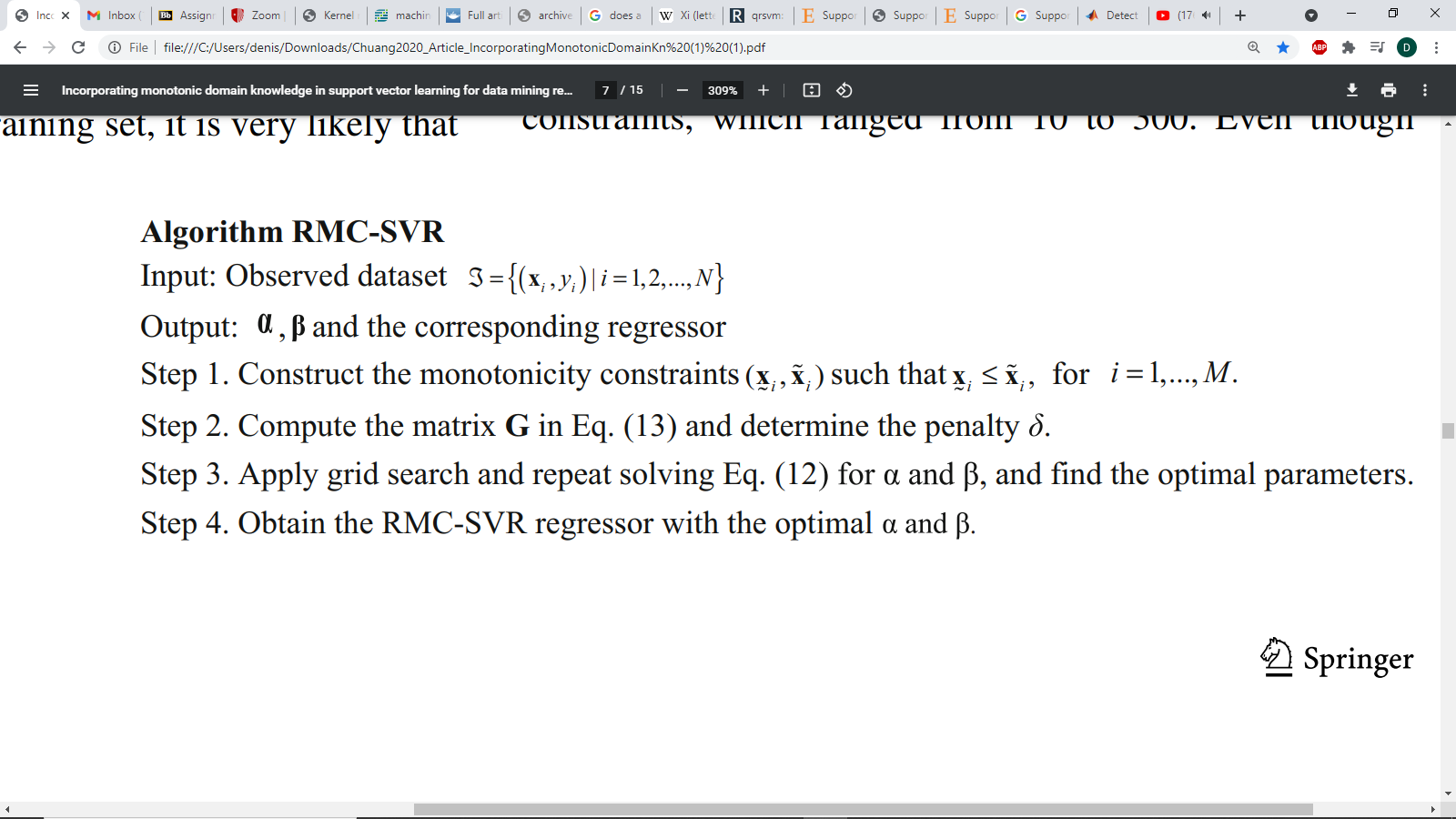
To leave off I will apply SVR to the log-retruns of stock data to compute a model for the return of a stock based on the other returns in the set using the best kernel choice followed by the monotonicity costrained SVR and I will compare the models based on their MSE.

To do this we must understand how to develop the monotonicity constraints. The paper gives an algorithm to do so as follows:



Note N is the number of observations, while n is the number of features per observation.

Then we can follow the following method to compute the MC-SVR



After spending many hours coding both a normal SVR model and enhancing the model to run RMC-SVR as described above, I am unable to find a solution to the RMC-SVR problem using quadratic programing in R on the log-return dataset, possibly due to an, error, but worse, using Matlab as described in the paper, I am unable to achieve the results published in the paper, using the regression modeling statistical app on the mpg dataset using the best known optimization techniques I was unable to get an MSE less than 84 while they report an MSE of 12 meaning they have made some mistake in their unpublished code.

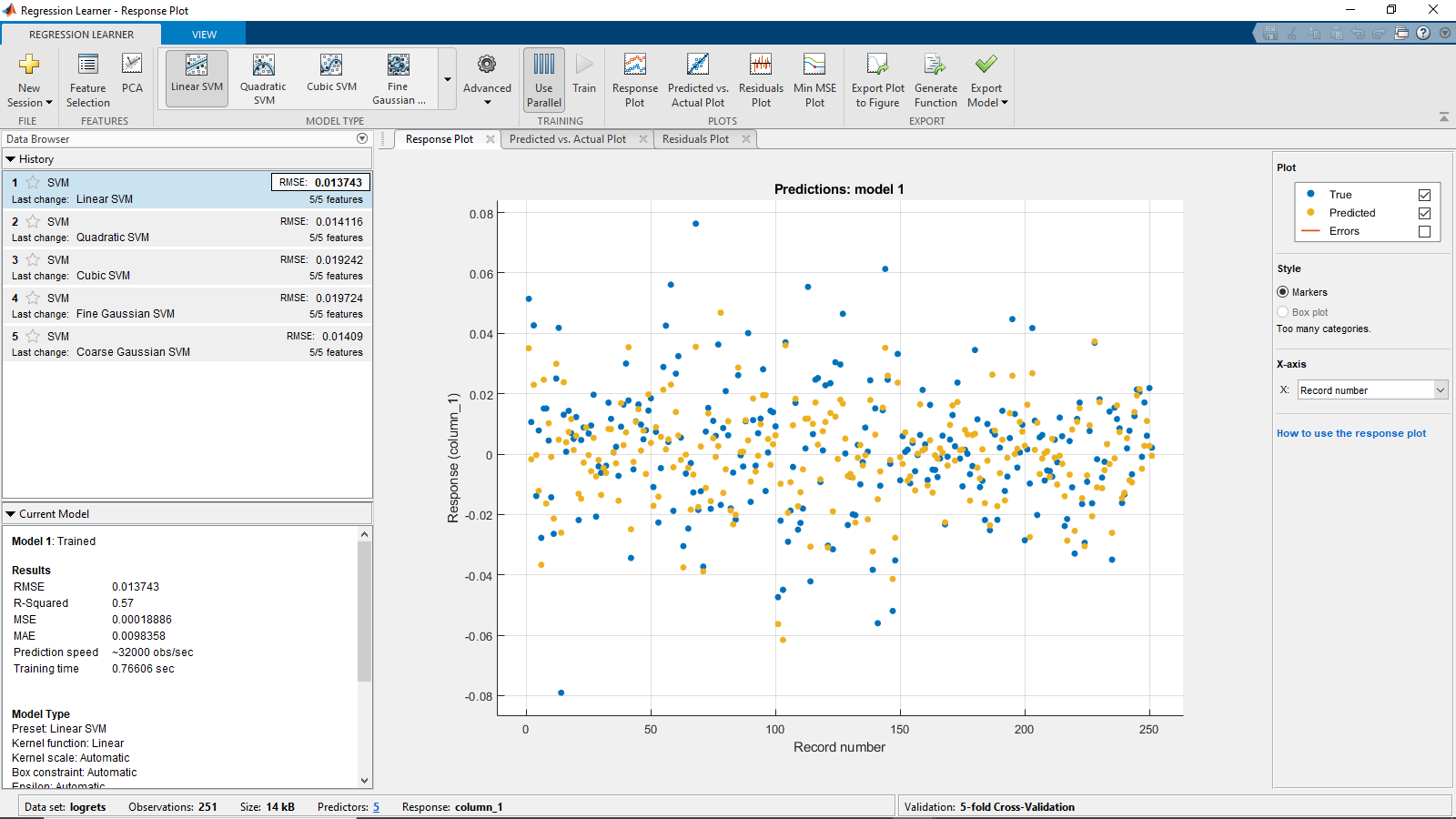
See my output from R in a pdf file to see the formulation of the MCSVR problem and the normal SVR problem in reference [16] on page 44.

In R coding the SVR problem from scratch I was able to achieve an RMSE of 0.025 on the log-return data to predict the price of the Amazon Stock using the linear Kernel which produced the best results using my code and setting the hyper-parameters C and by free choice. In MatLab however the model could be fitted using Grid Search to find the optimal linear hyperparameters C & as , 0.002 & 0.017 respectively, when fitted to minimize cross validation error on 5 out of sample observations with an RMSE of 0.013743. My R code and results are included in a separate pdf but here we will discuss the Matlab work because it is easier to digest visually. In this optimizer the RMSE for multiple models is given below with optimal hyper parameters, along with plots of the predicted vs actual observed values.

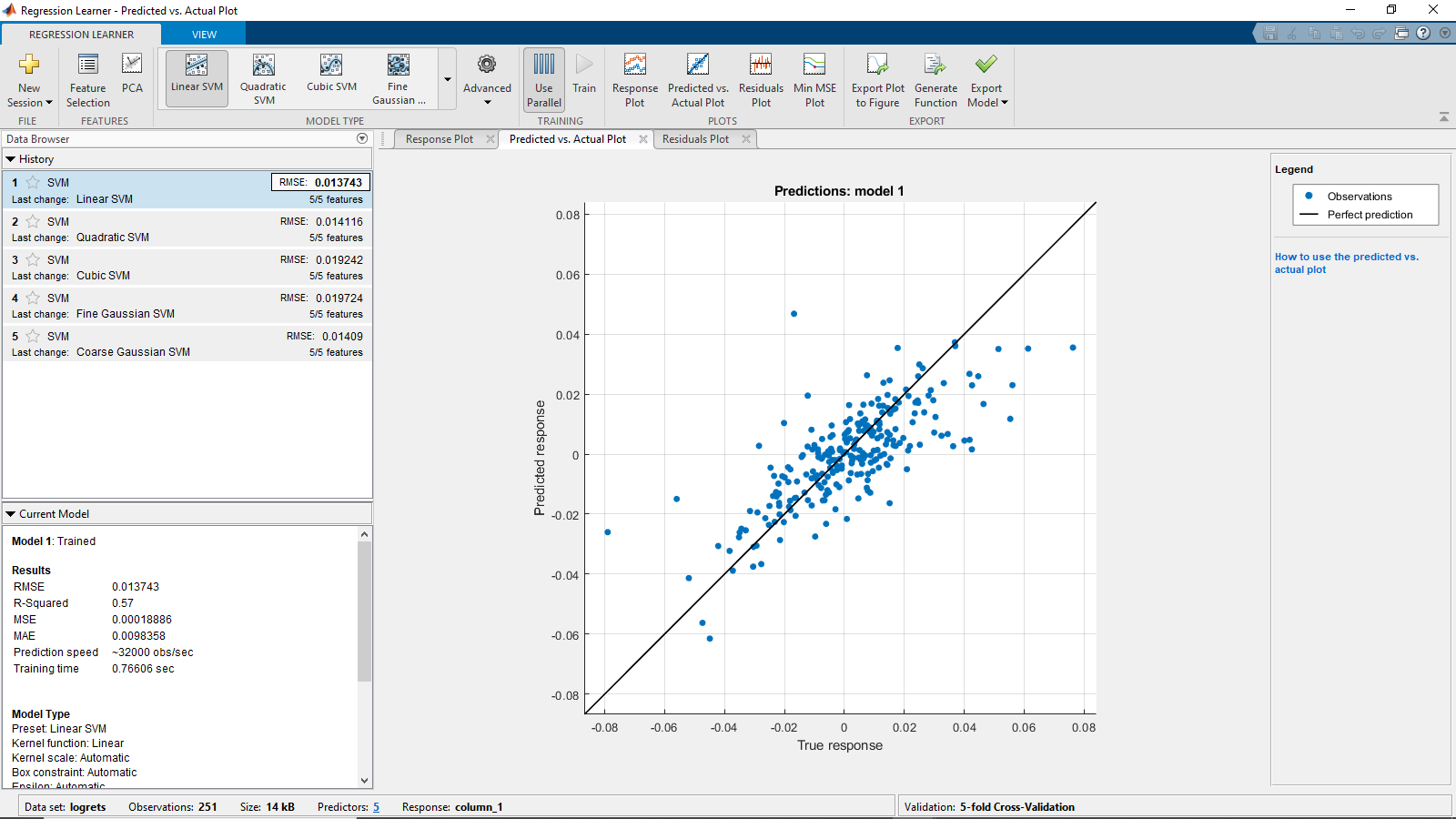
**Log-return data [reference 18]:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kernel |  | C |  | RMSE |
| Linear | 0.017 | 0.002 | NA | 0.013743 |
| Poly 2 | 0.017 | 0.002 | NA | 0.014116 |
| Poly 3 | 0.017 | 0.002 | NA | 0.019242 |
| Gaussian (Fine) | 0.017 | 0.002 | 0.56 | 0.019724 |
| Gaussian (Coarse) | 0.017 | 0.002 | 8.9 | 0.01409 |

Below are plots of the best model, the Linear SVR model for the log-return data



In this first plot we can see that the predictions of this model are generally close to the actual log-return.



Looking at the Predicted vs Actual plot we see the model generally performs well on all quantiles except extreme tail events. We did not discuss this in this paper but, SVR has been adapted to run VaR and C-VaR regression using the Pinball Loss as discussed in the paper “Support vector machine quantile regression approach for functional data: Simulation and application studies” linked in my references.

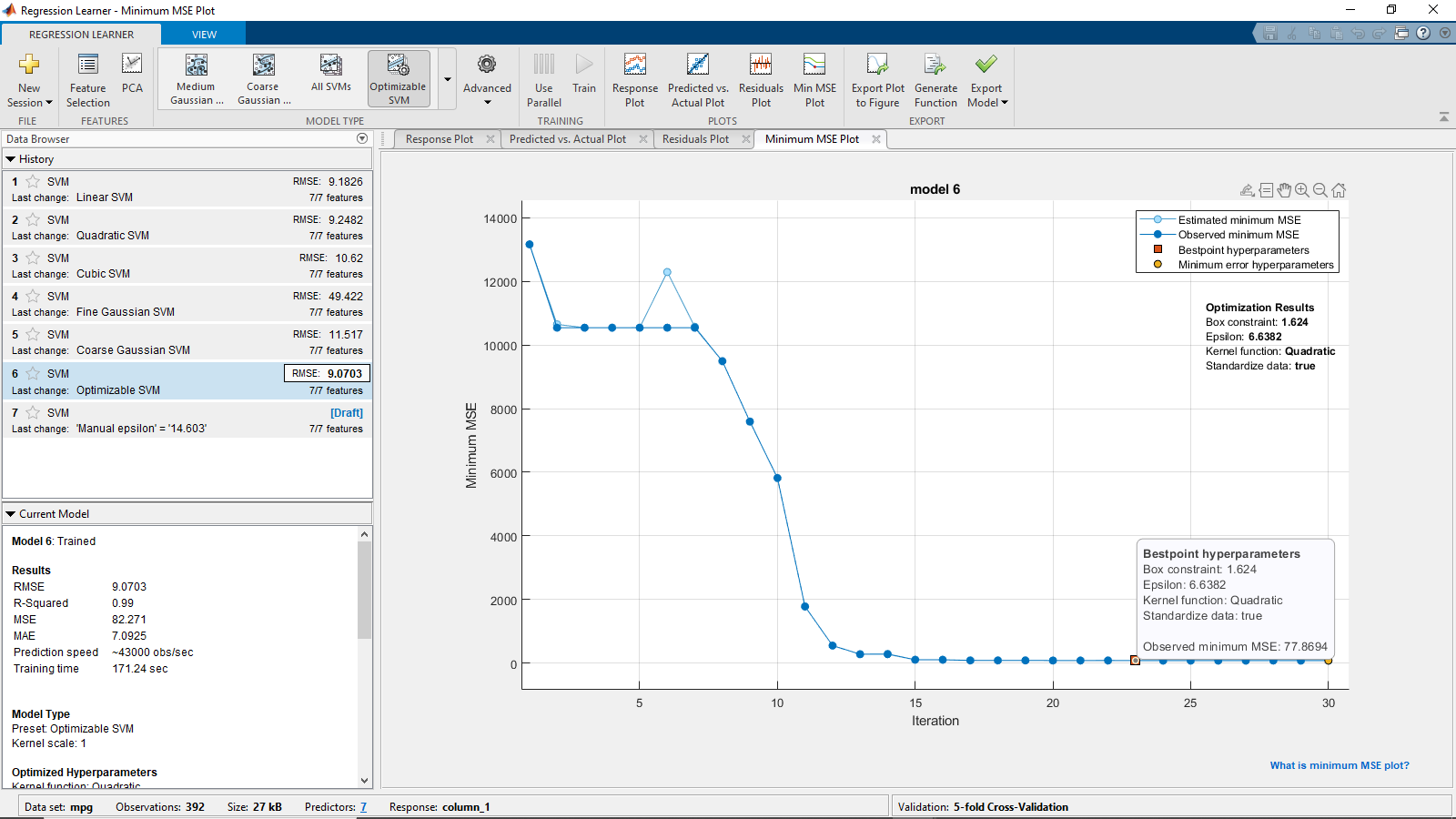
Plots for the other models in Appendix. Reproducible code found in References [17]

Turning our attention to the mpg dataset, I have produced the same table below with models fitted via minimizing Cross-validation error.

**Auto-MPG data [reference 9]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kernel |  | C |  | RMSE |
| Linear | 14.603 | 146.034 | NA | 9.1826 |
| Poly 2 | 14.603 | 146.034 | NA | 9.2482 |
| Poly 3 | 14.603 | 146.034 | NA | 10.62 |
| Gaussian (Fine) | 14.603 | 146.034 | 0.66 | 49.422 |
| Gaussian (Coarse) | 14.603 | 146.034 | 11 | 11.517 |
| Minimum MSE Optimization Model:  Poly 2 | 6.638 | 1.624 | 0.336  (scale is a multiplier on the dot product inside the polynomial ) | 9.0703 |

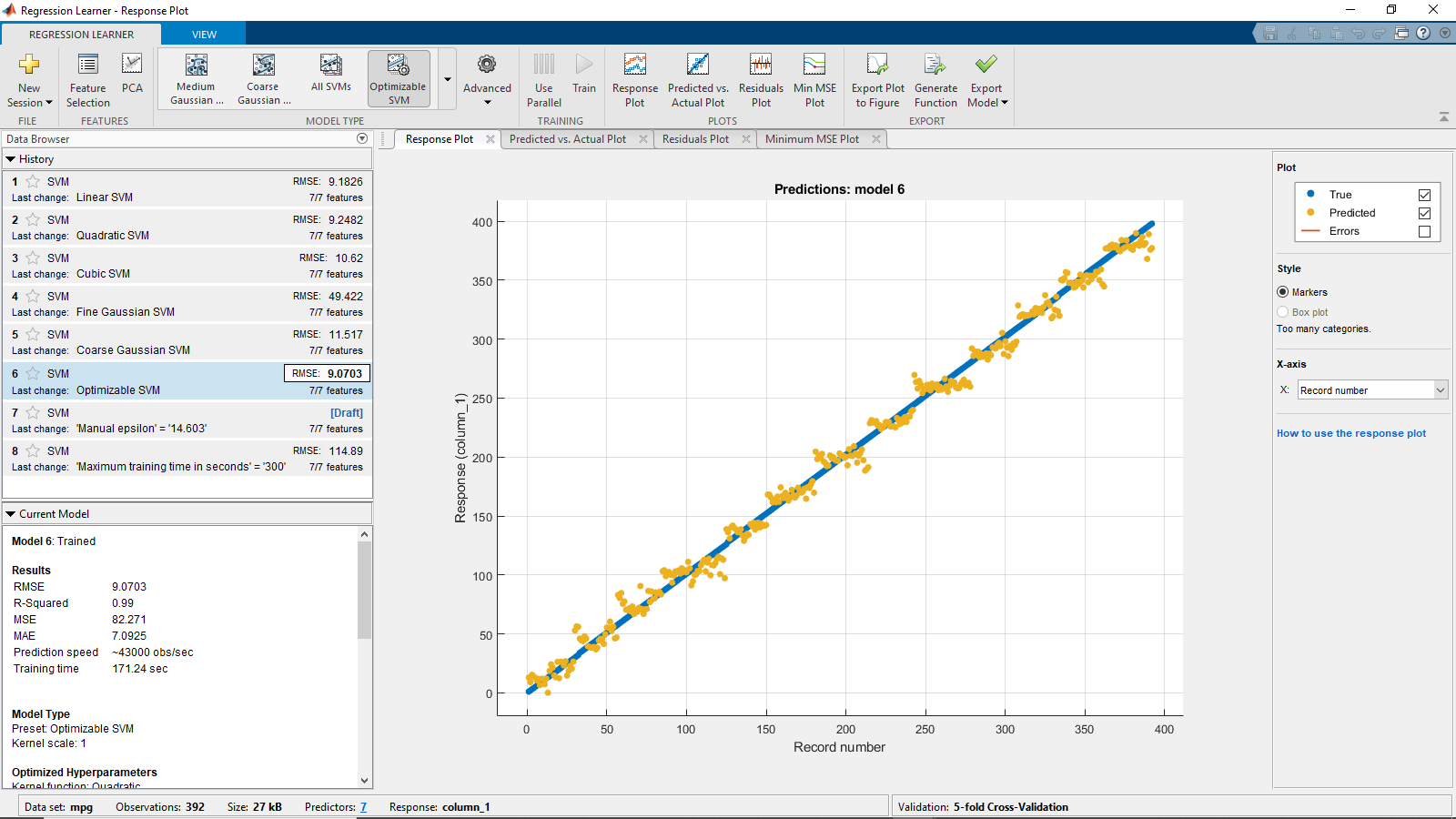
On this data I had hoped to let the optimizer look for the best possible model in hopes it might find the hyperparameters used in the paper, however the results were unable to come close to the performance of the model reported in the paper. Below is a plot of the MSE of the predicted model as a function of the optimizations made to the hyper-parameters via grid-search methods:



This plot actually gives us very valuable insight for a few things. As can be seen in my coded SVR in R, making a poor choice of hyperparameters, wildy destroyed the MSE of my model which in R I had found an RMSE of close to 11,000. This shows that although SVR can be a useful tool, it is necessary to use insightful hyper-parameters.

More importantly, this plot proves devastating for the original problem we set out to solve with this paper. If we take the results of the MC-SVR paper on this same data to be truthful (I state this in such a way because it cannot be reproduced) then it would seem unlikely that if there were a solution to the Problem, ie that there exists some kernel for which the results are equivalent to the MC-SVR problem then it should be possible to find the kernel, or at least a kernel with parameters which produce results with close to the same output but we can see that using the unconstrained SVR approach we were not able to get to an MSE below 84 while the paper reports an astounding MSE of 6 on this same dataset using the MC-SVR approach while I had let my computer search for optimal parameters for at least 10 minutes with a rate of about 4,000 kernels checked per second. For this reason, I would say that it seems unlikely that the problem has a solution (assuming the results reported are true). I would compare to the R model, but this again found no solution, meaning I may have a mistake in the code somewhere.

The last thing I would like to point out in my research is that due to the nature of SVMs and their original intention being for classification, we can see in the data for the highly optimized model, data points are usually clumped into clusters. This just happened to be an unusual feature I had noticed for the Gaussian models and the hyper-parameter optimized model seen below:



In this model we can see that the true data in blue was continuous on the interval while the predicted values were clustered into groups and we have a model similar to a clustering classifier.

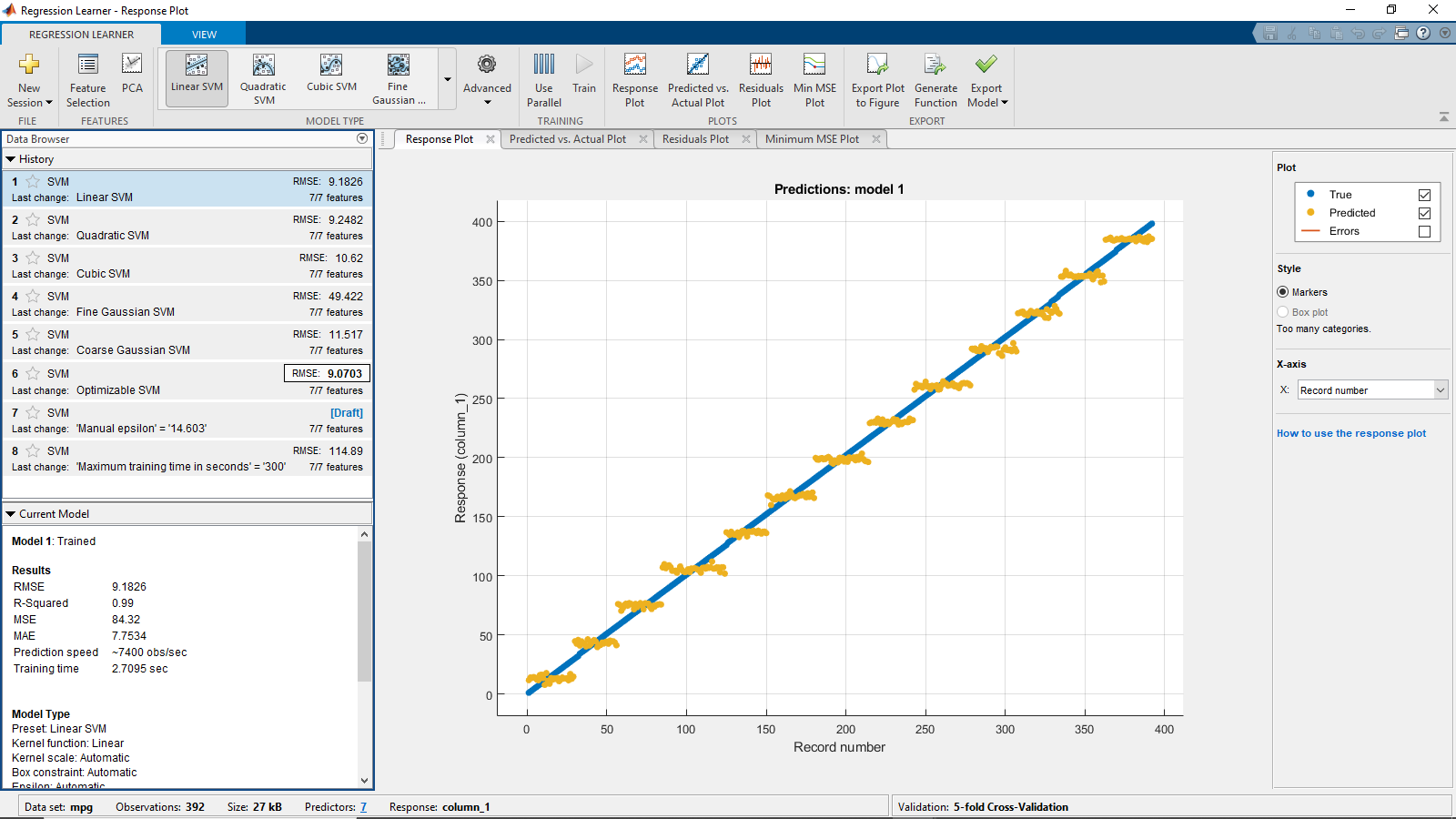
**Conclusion:**

We have seen that Support Vector Regression can be an extremely efficient method for developing a model, however it is clear none of this work could be possible if it had not been for the strengths of the Kernel Trick to avoid having to consider hyper dimensional transformation functions . From this augmented view of the inner product on these hyper-spaces, we can see that it seems feasible that one should be able to use kernel functions to avoid having to work with large numbers of additional constraints, which can increase computation time and memory required to solve by the cube of the sum of the number of constraints and the number of data observations. With the work here I hope it is clear that if the problem does have a solution, it will require making a clever trick with the base kernel functions. If one were to solve this, then it would be possible to run SVR while incorporating prior knowledge on the data more easily by simply updating the kernel choice.

**Appendix:**

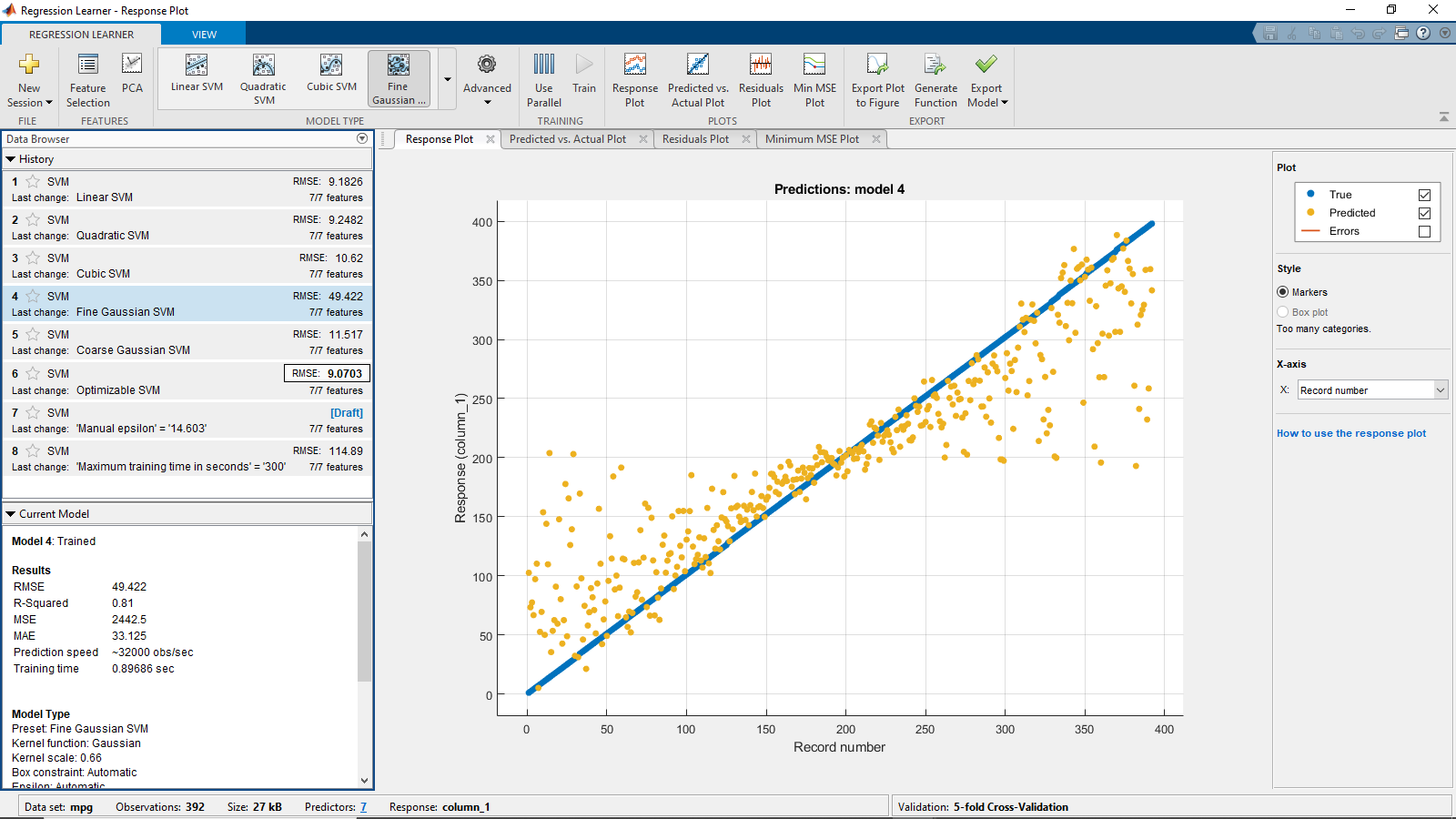
Plots of the MPG data by Kernel

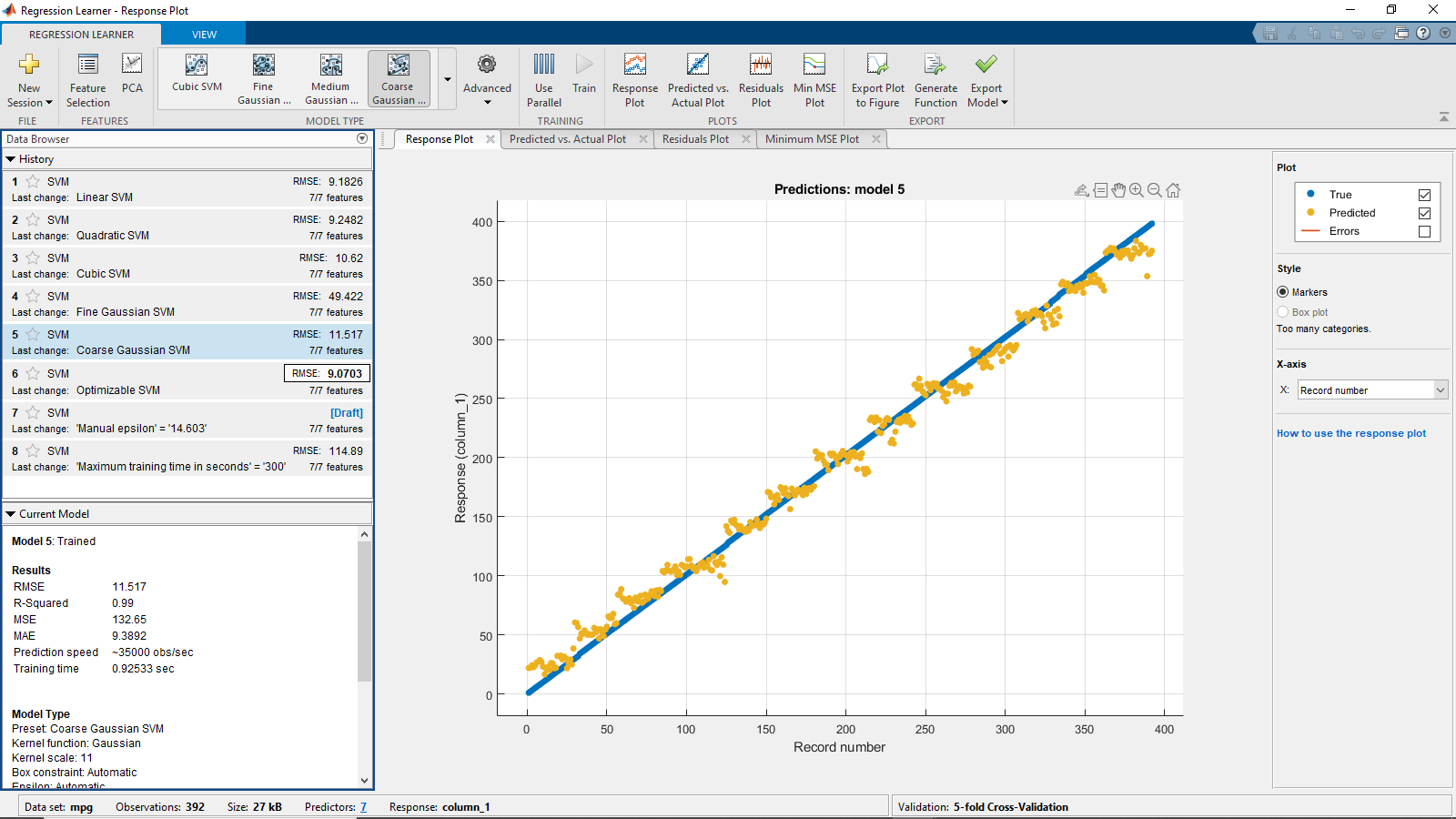
Orange= Predicted, Blue = True

Linear: 

Polynomial 2: 

Poly3: 

Fine Gaussian: 

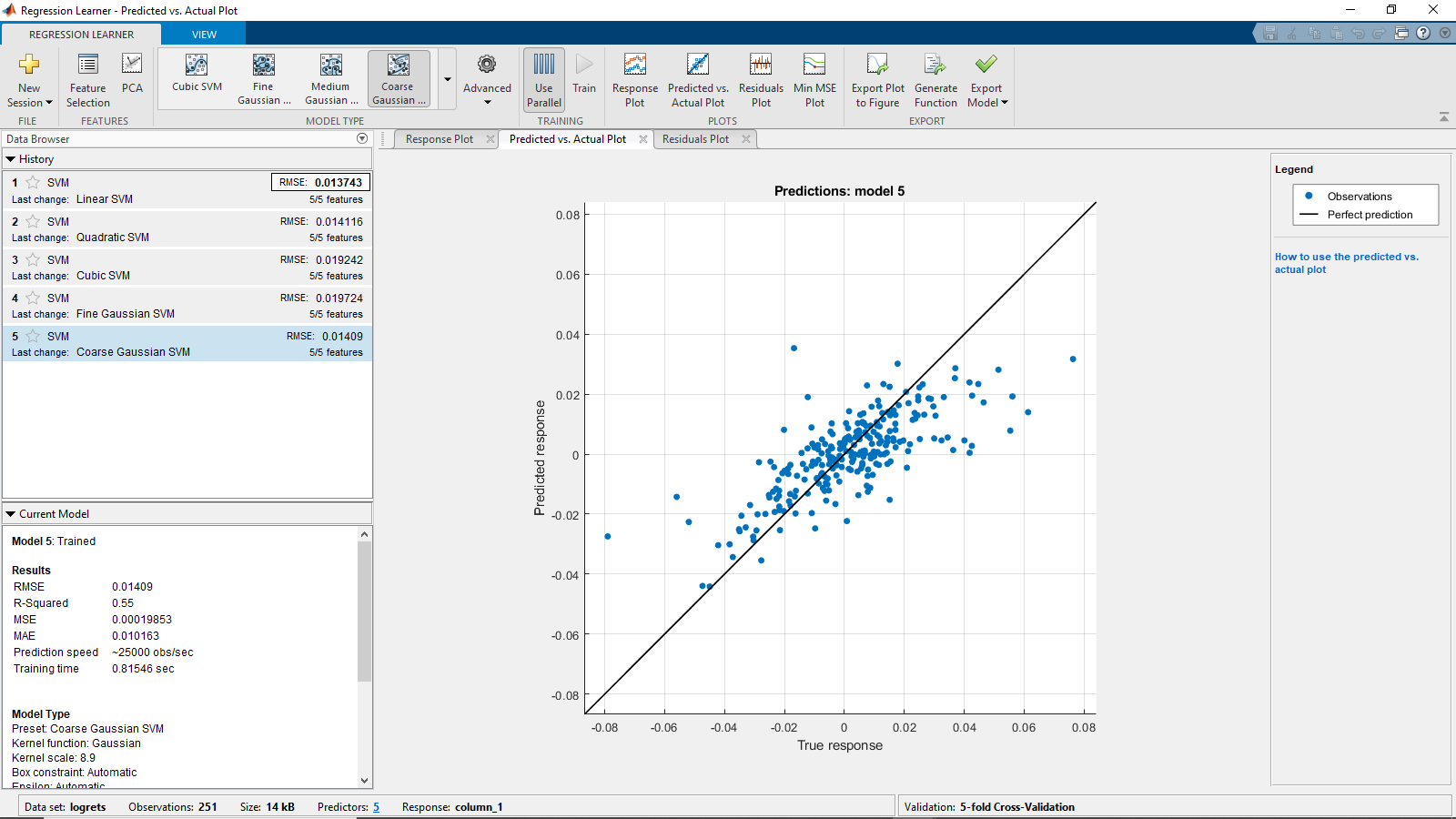
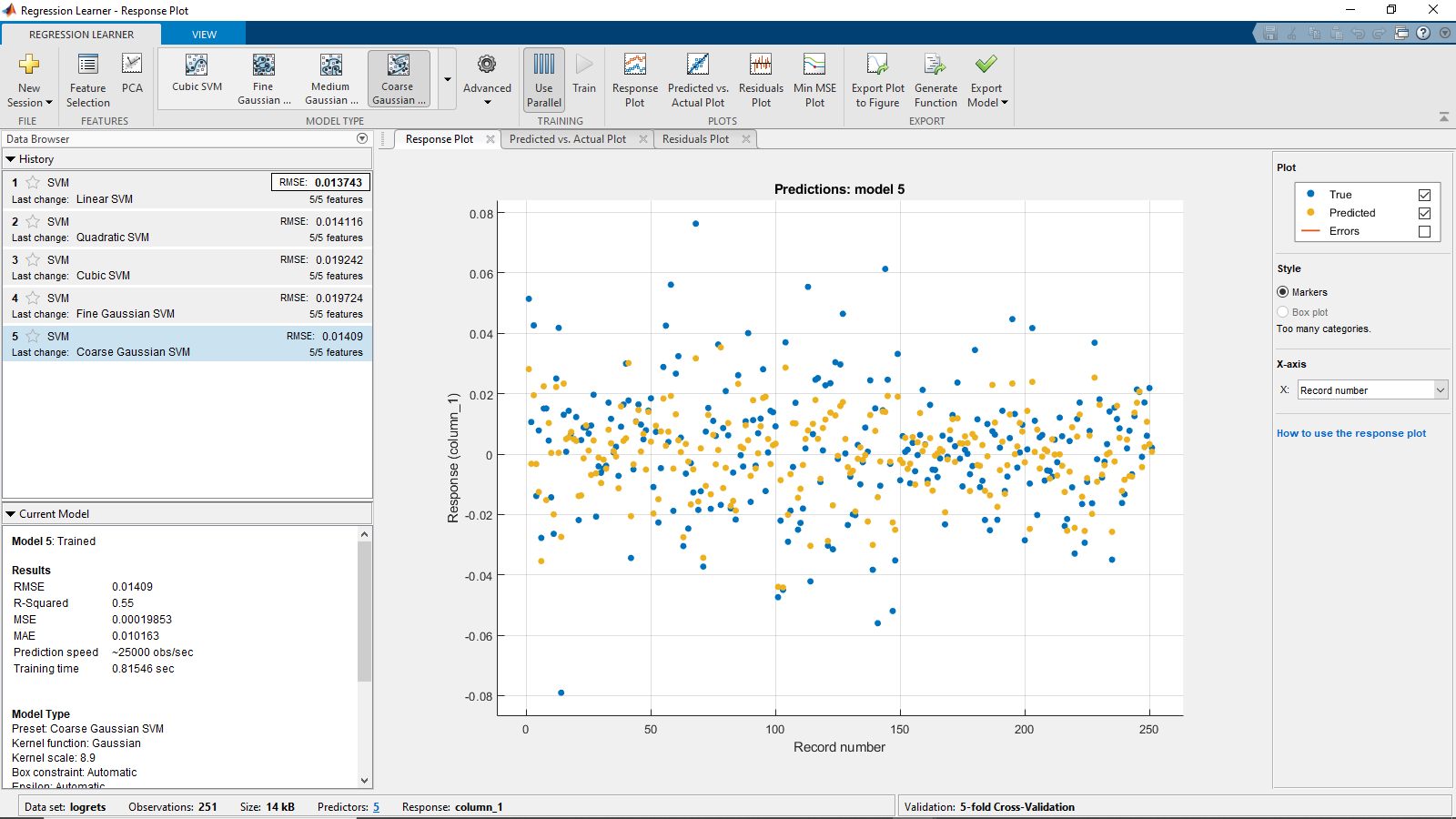
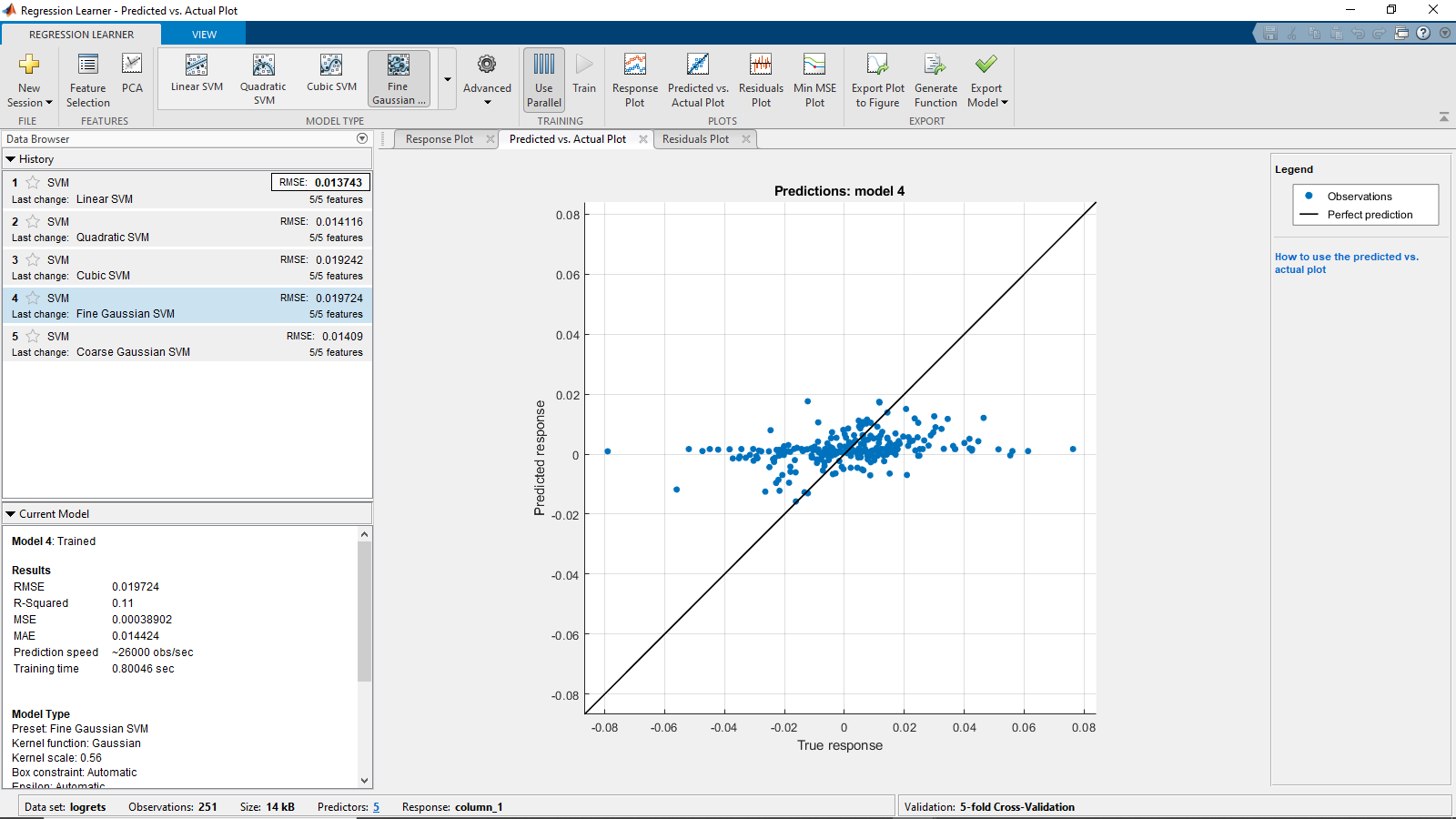
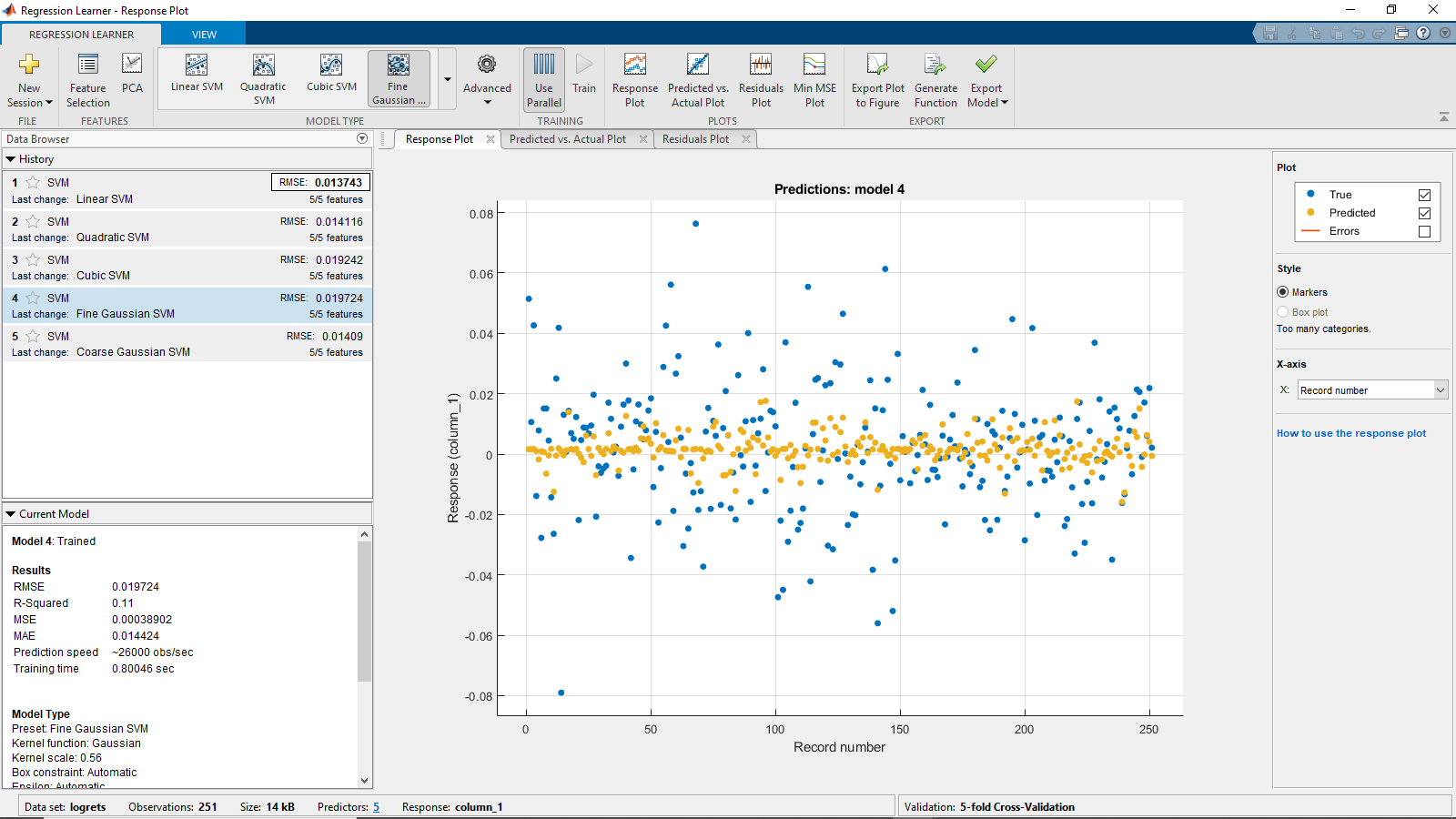
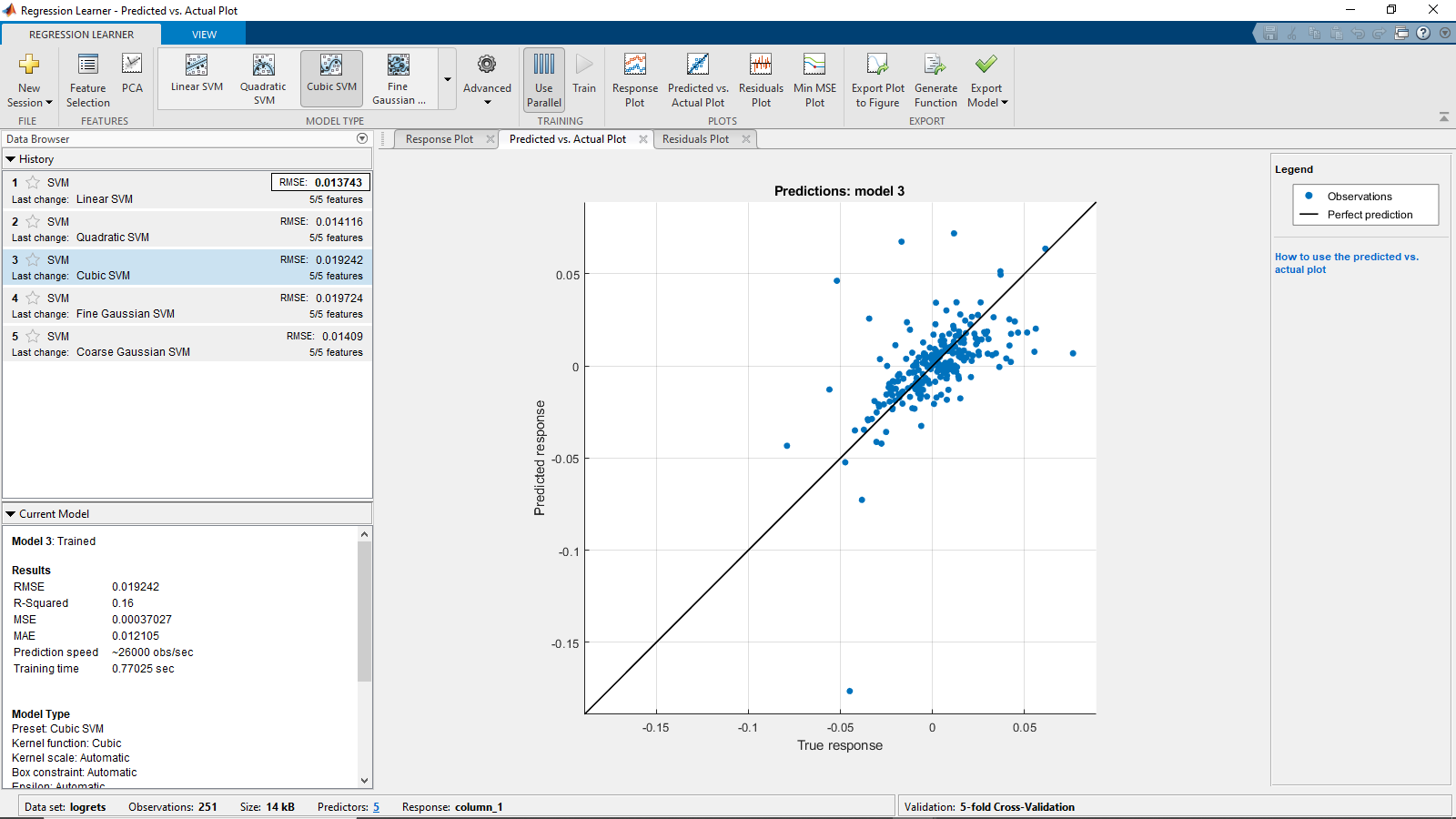
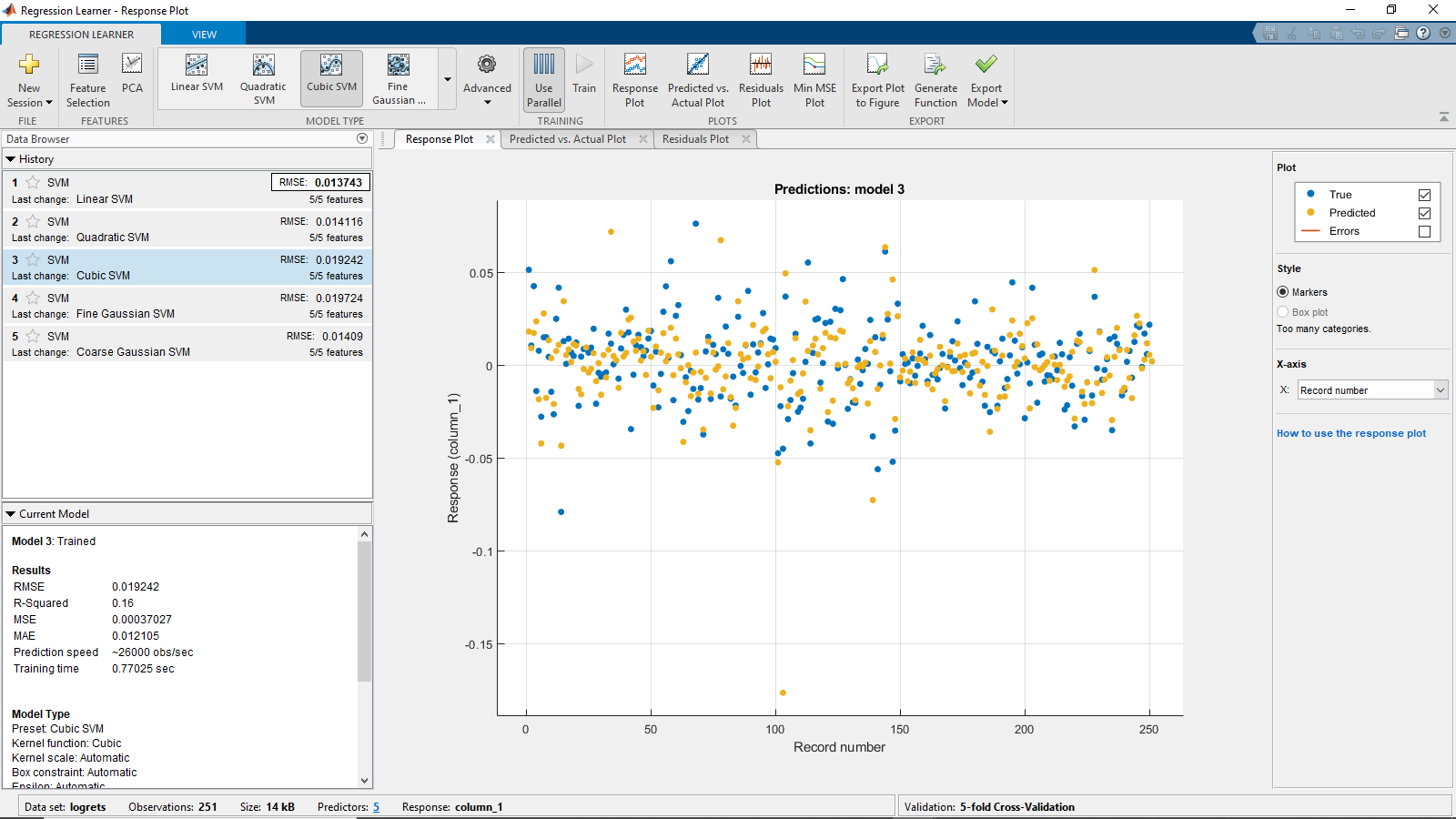
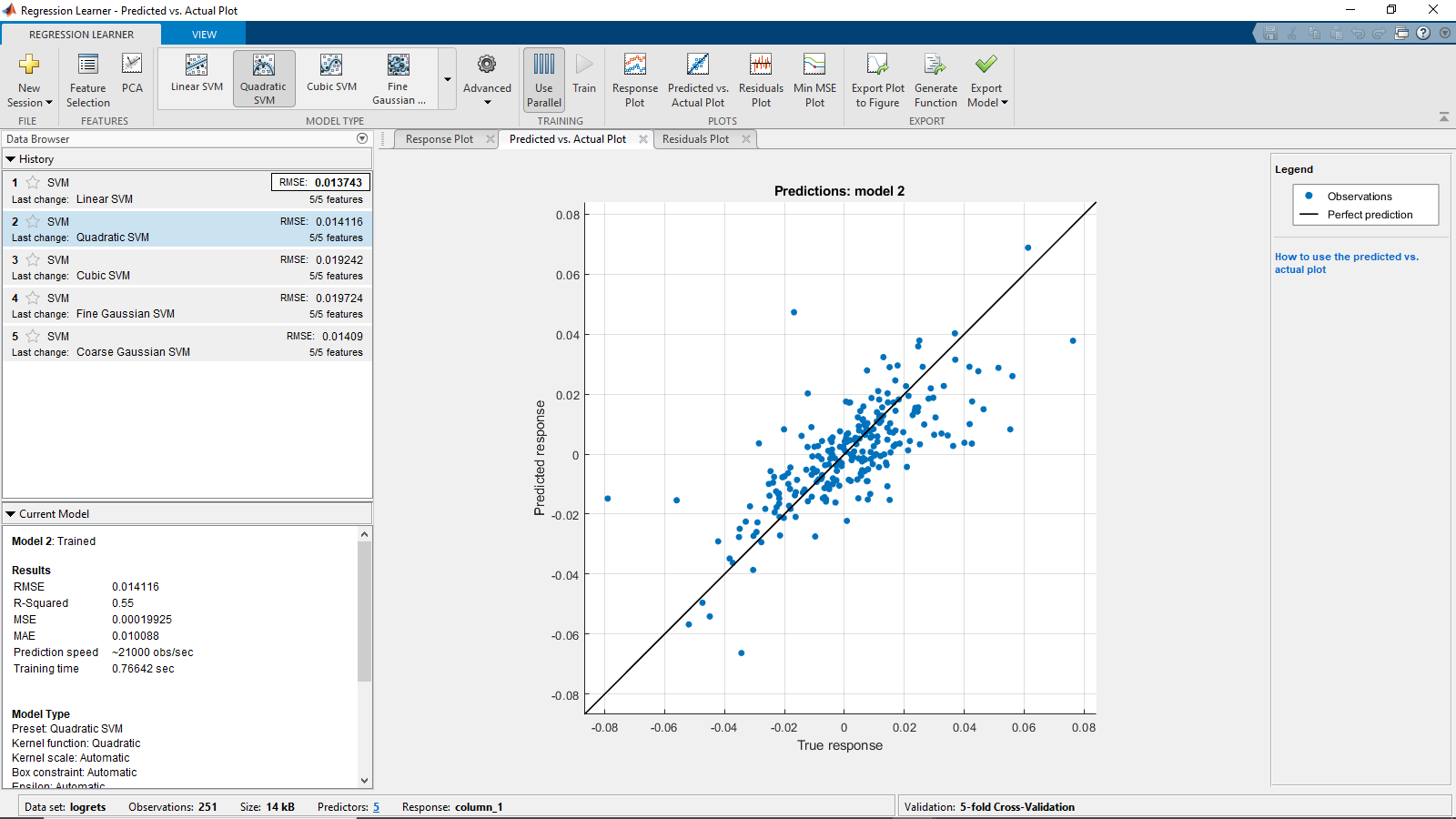
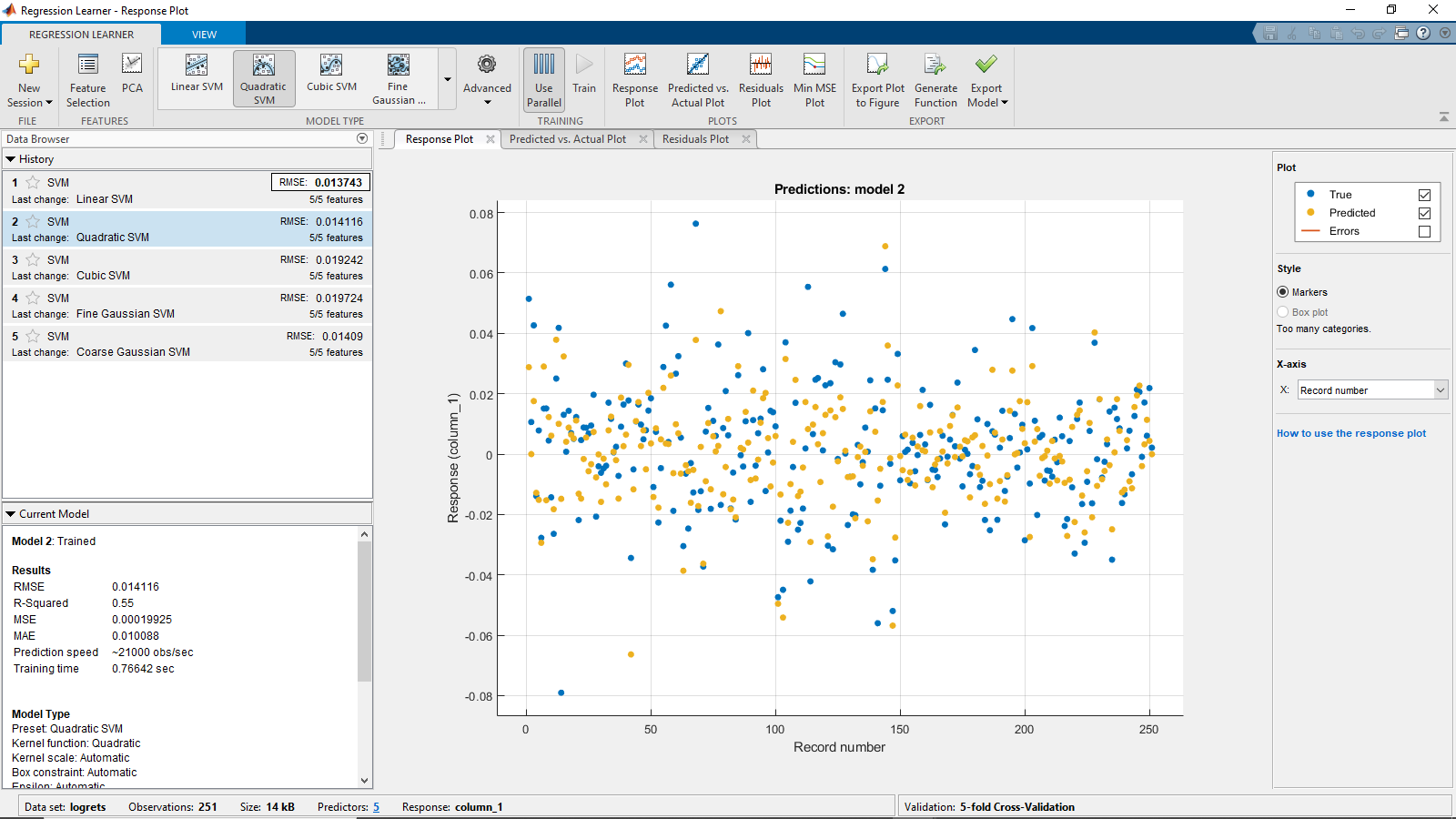
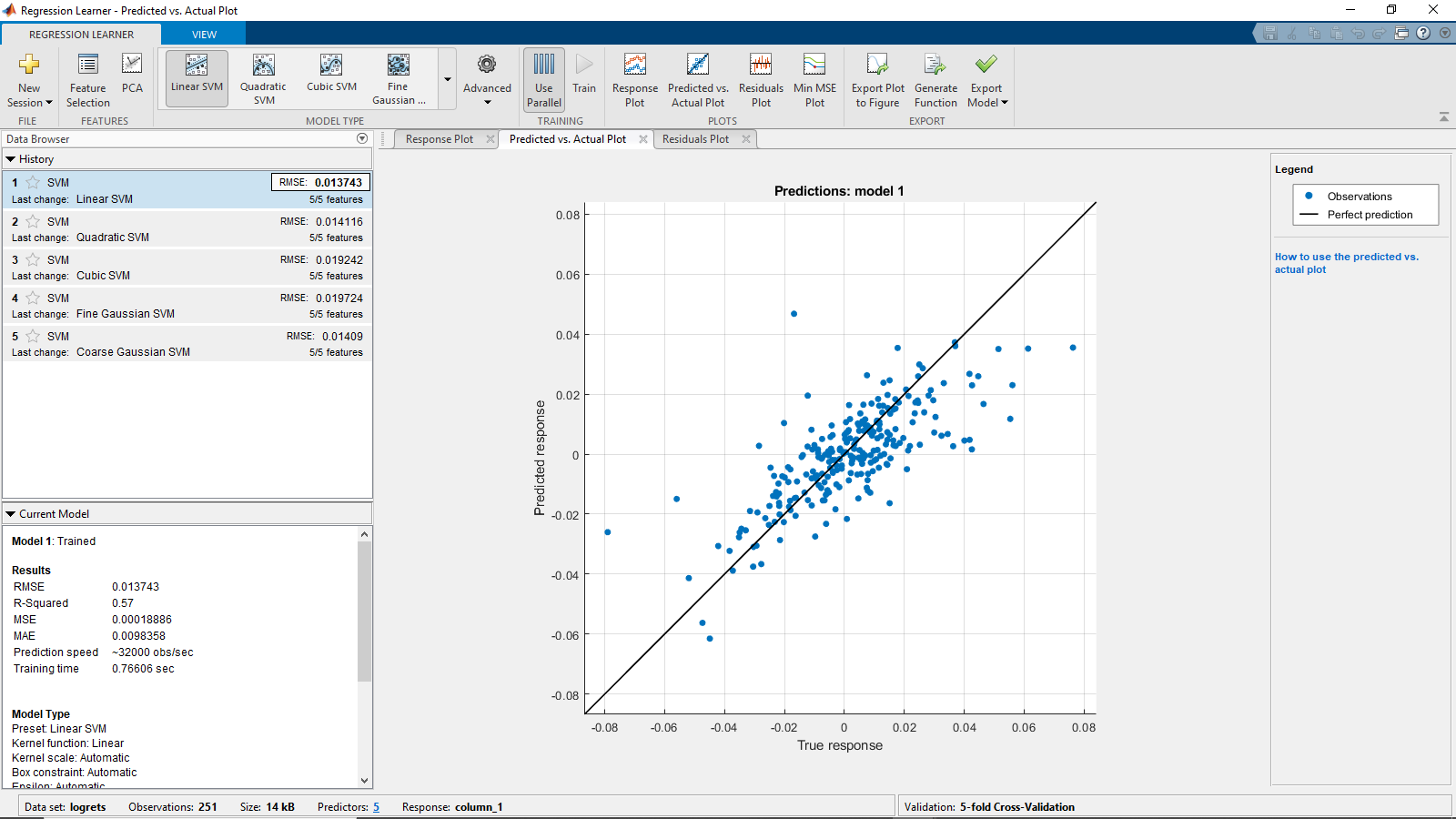
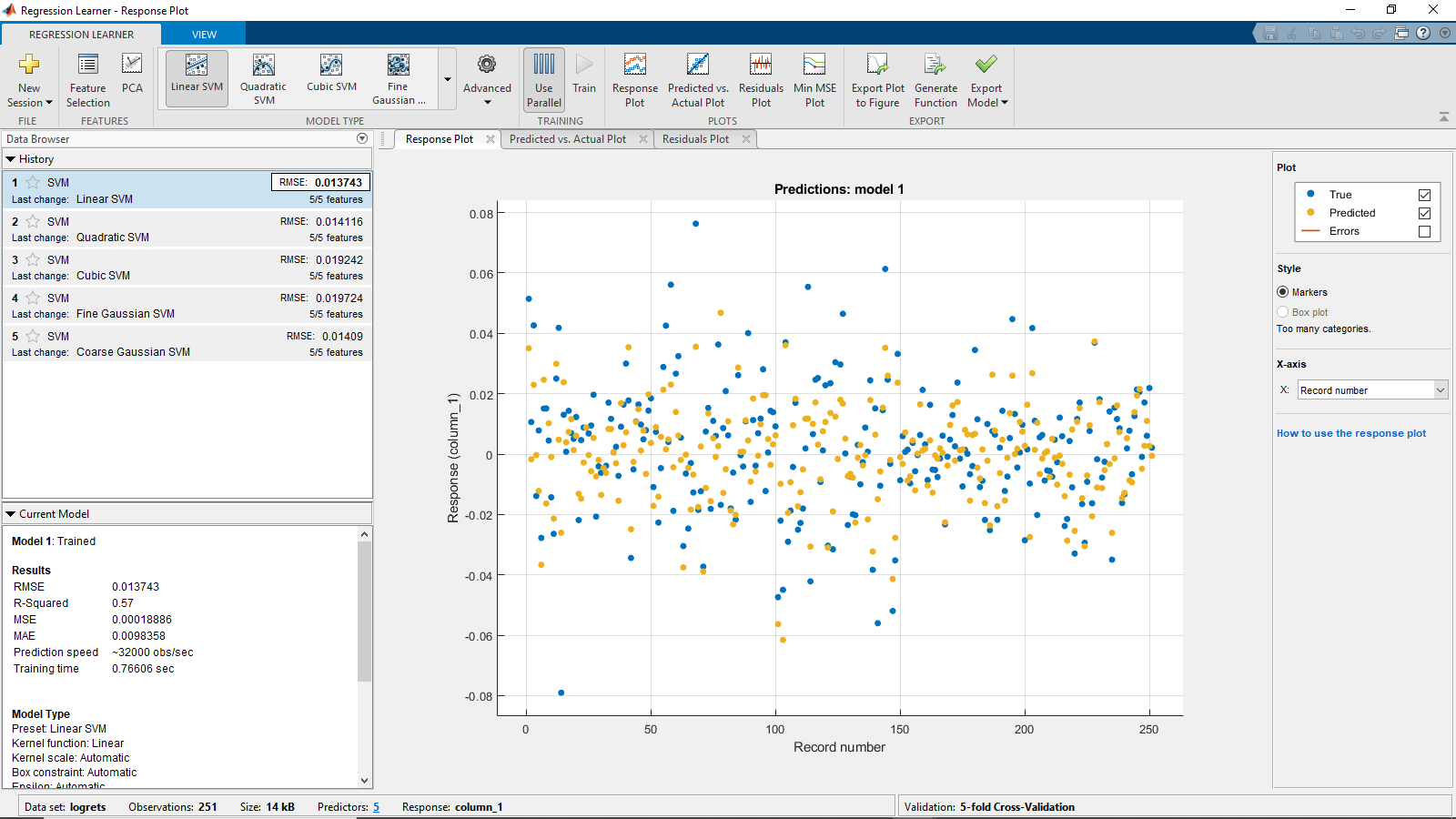
Coarse Gaussian: 

Plots of the log-return data by Kernel

Orange= Predicted, Blue = True

Linear (shown in paper as best model)

I will let the software label the rest in the screenshots.



**References:**

If any references are miss labeled in the paper I apologize I tried to move the most prominent references to the front and my code to the back for ease of access.

[1] Chuang, HC., Chen, CC. & Li, ST. Incorporating monotonic domain knowledge in support vector learning for data mining regression problems. *Neural Comput & Applic* **32,**11791–11805 (2020). <https://doi.org/10.1007/s00521-019-04661-4>

Paper suggested by Ph.D. Chris Bemis to use for studying Monotonic constrained Support Vector Regression

[2] Thomas Hofmann. Bernhard Schölkopf. Alexander J. Smola. "Kernel methods in machine learning." Ann. Statist. 36 (3) 1171 - 1220, June 2008. <https://doi.org/10.1214/009053607000000677>

Essential for understanding the Kernel trick and formulating the associated functions

This paper was the most helpful in understanding the actual problem posed by Bemis.

[3] bemis, chris, The Relationship Between Investor Views, Constraints, Expectation, and Covariance in Mean-Variance Optimization (May 24, 2020). Available at SSRN: <https://ssrn.com/abstract=3609510> or [http://dx.doi.org/10.2139/ssrn.3609510](https://dx.doi.org/10.2139/ssrn.3609510)

Paper By Ph.D Chris Bemis, source of inspiration for searching for a method to remove constraints on the optimization problem in this paper. I should also add that he was very helpful in the formulation of the problem for this paper and gave useful insight on the problem formulation for unconstrained SVR.

[4] Mathematical Formulation of SVM Regression

https://www.mathworks.com/help/stats/understanding-support-vector-machine-regression.html

Most of the derivation of the SVR in my paper comes from this page and my understanding of the topic is mostly due to the help of this page.

[5] Support-vector machine Wikipedia

https://en.wikipedia.org/wiki/Support-vector\_machine

Helped to understand the Concept of Soft vs Hard margins and the associated hyper-parameters C and

[6] MatLab Regression Learner Application

<https://www.mathworks.com/help/stats/regression-learner-app.html>

Used to run optimization of hyper-parameters

[7] Implementing a Soft-Margin Kernelized Support Vector Machine Binary Classifier with Quadratic Programming in R and Python

<https://sandipanweb.wordpress.com/2018/04/23/implementing-a-soft-margin-kernelized-support-vector-machine-binary-classifier-with-quadratic-programming-in-r-and-python/>

Used to develop code for running SVR in R and developing Monotonicity Constrained SVR code

[8]QuadProg Package in R

<https://cran.r-project.org/web/packages/quadprog/quadprog.pdf>

Used to solve Lagrangian maximization problem for SVR in my code in R

[9] Christophe Crambes, Ali Gannoun, Yousri Henchiri,

Support vector machine quantile regression approach for functional data: Simulation and application studies, Journal of Multivariate Analysis, Volume 121, 2013, Pages 50-68, ISSN 0047-259X, <https://doi.org/10.1016/j.jmva.2013.06.004>. (https://www.sciencedirect.com/science/article/pii/S0047259X13001164)

Used as a reference for some formulas. I would recommend this Paper as it goes into applying the Pinball Loss for the SVR to compute VaR regression

[10] Auto Mpg Dataset from UCI Machine Learning Repository

<https://archive.ics.uci.edu/ml/datasets/auto+mpg>

[11]Power Point By Professor Wei Zhu from Stony Brook University on Support Vector Machines

Part1:

<https://drive.google.com/file/d/1CtEBgyPuOnBQY-C0-vM4411KGeW96LvE/view?usp=sharing>

Part 2:

<https://drive.google.com/file/d/1fPZ3IjeU_7cJoUro2VRtg9P28oaCMb8P/view?usp=sharing>

Notes on Support Vector Machines from my AMS 580 Class this semester

[12] Radial basis function kernel

<https://en.wikipedia.org/wiki/Radial_basis_function_kernel>

Derivation of the fact there does not exist a linear transform for the RBFK

[13]Kernel method

<https://en.wikipedia.org/wiki/Kernel_method>

Used to help introduce the kernel trick

[14]Kernels and support vector machines

<https://web.stanford.edu/class/stats202/notes/Support-vector-machines/Kernels.html>

Notes on SVM from Stanford University

[15]Polynomial kernel

<https://en.wikipedia.org/wiki/Polynomial_kernel>

Derivation of degree 2 polynomial transformation function

[16]Stock data from Yahoo Finance Used to compute logrets dataset.

<https://drive.google.com/drive/folders/1Xszw_JKmou63495yOWGDReZGE45l-AHb?usp=sharing>

[17] My R code as an RMD file and PDF of Output:

<https://drive.google.com/drive/folders/1E5qhfHlLCpuGavJTyqLFrsT7XarmNd7Z?usp=sharing>

[18]My MatLab Code Folder for different Models:

<https://drive.google.com/drive/folders/1ZyWBtD7WBadLQj3xTP6RW-_O--IDnV9s?usp=sharing>

[19]My Log-Return Data

<https://drive.google.com/file/d/11rW-vQOCHglKiuHm_qcjSkbWjRHE_yum/view?usp=sharing>