# Outline

1. Abstract

2. Intro

3. Datasets & Algorithm Intro

1. Data

- Abalone

- Madelon

2. Algorithms

- Decision Tree

- Boosting

- ANN

- KNN

- SVM

4. Methodology

\* Brief overview of scripts/pipeline

5. Discussion: Algorithms Analysis

\* For each algorithm:

\* Train/Test Error Rates

\* At the end if there's enough time, might want to add

precision, recall, f1 score as metrics on best hyper param models

\* Training Time

\* Learning Rate

\* 'Overfitting' Curves (Expressiveness)

\* Hyperparameter Analysis

\* Why did these come out the best?

Discuss what each parameter does and reasoning for why that performed best.

Look at Grid search results if possible to see if distribution of params change

the outcome much (i.e. the effect of param on performance)

6. Conclusion

\* Key Takeaways

**Abstract**

The world is full of problems, which means there's learning to do. If you've got answers, I've got supervised learning techniques. Specifically, if the problem happens to be identifying the age range of abalones or identifying a non-linear classification with vast amounts of noise added then the supervised learning algorithms for Decision Trees, Boosting, Artificial Neural Networks, K Nearest Neighbors and Support Vector Machines may be of help. For each of these datasets, and in turn each algorithm, the accuracy of classification was tested under cross validation over a variety of hyperparameters (learning rate, regularization, etc.) using sci-kit learn's GridSearchCV. The resulting hyperparameters, model performance, learning curve's and 'overfit-ability' are examined. This analysis gives way to furth insight into both the algorithms and problems.

Intro

Datasets

Interesting analysis requires interesting problems, and in order illuminate the strengths, weaknesses and quirks of the examined supervised learning algorithms two well known datasets from the UCI machine learning repository data are examined.

***Cars***

*Instances*: 1728

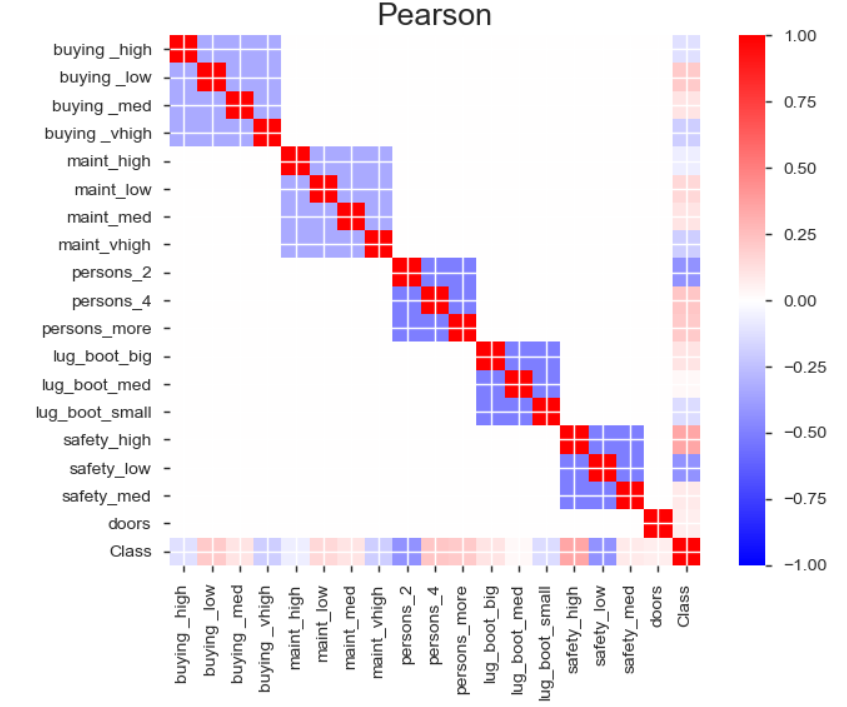
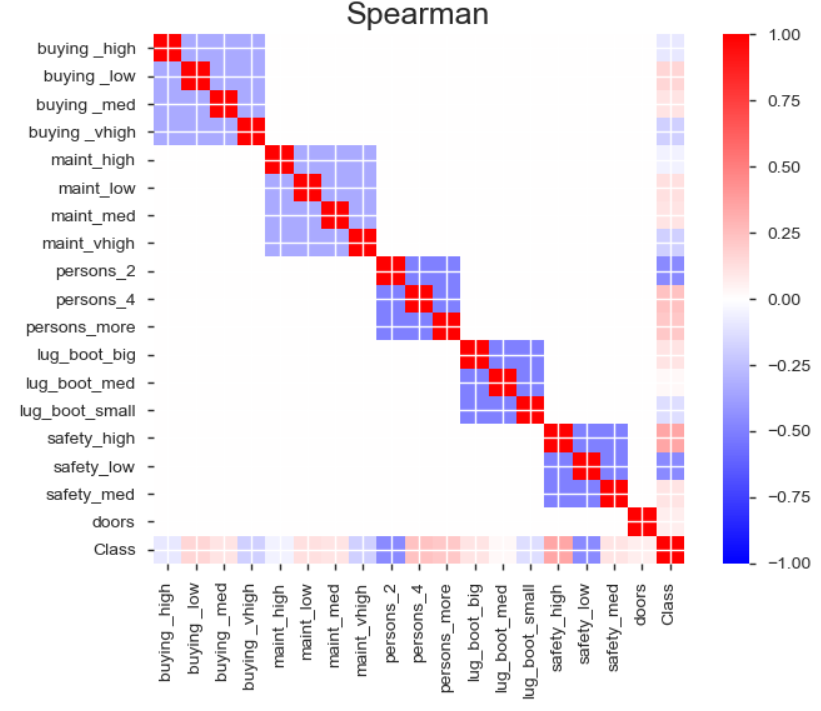
*Attributes*: 6 (19 when one hot encoded)

*Data Types*: Categorical (6)

*Classes*: 0, 1, 2 for Cars with ratings of Unacceptable (0), Acceptable (1), Good/Very Good(2)

This dataset measures the hiearchical characteristics of cars to predict the overall acceptability rating of the car. There are two overarching characteristics measure: price (buying and maintenance), technical (number of doors, person capacity, trunk size and safety rating) . Within those six attributes measured, each has multiple ordinal levels. The dataset was originally designed to showcase a simple hierarchical model for decision making. It serves as a test for a learner’s ability to recognize structure within the six dimensions. Post processing the categorical values to a single dimension for each (e.g. Person Capacity processed from one dimension with three levels into three dimensions, each designating a separate condition (capacity-2, capacity-4, capacity5+), the dataset expands to 1728 instances with 19 dimensions. A learner will have to distinguish the which of the dimensions are related (e.g. capacity-2, capacity-4, capacity5 are not independent) and model them accordingly. Additionally, the classification problem is imbalanced with the vast majority of cars classified as unacceptable (70%), while 22% are acceptable and the final 8% classified as good. The distribution of the categorical data are balanced between levels (e.g. capacity-2, capacity-4, capacity5 each have 1/3 of the total instances in each level).

There are no dead giveaways relating any of the categoric levels to the classes, as evidence by the correlation matrix. Since the labels are roughly ordinal in that a higher level indicates a higher quality, the Pearson and Spearman correlation should indicate a relationship for a categoric level being associated with an increase or decrease in quality:

The bottom row of the figures highlight the relationship between the categoric levels and quality class. Most have a weak relationship (+- .25) at best, which will require the learner to recognize the combinations of the weak relationships in patterns related to Class.

***Madelon***

*Instances*: 5000

*Attributes*: 440

*Data Types*: Continuous (440)

*Classes*: 0, 1

The MADELON dataset is an artificial dataset created in 2003 for the NIPs conference as part of a feature selection challenge. The target class comes from a group of 32 clusters on the vertices of a five dimensional hypercube. Those points were randomly assigned a class (either 1 or -1). Additionally the five dimensions were transformed by linear combinations to form fifteen more features. To complicated the problem 480 features of random noise were added to the dataset.

Of particular interest here is that the Madelon dataset presents a highly non-linear problem where the signal-to-noise ratio is very low. 1% of the features are truly useful (the 5 dimensions) while 15 (3%) are superfluous albeit still informative. This leaves 96% as completely useless to learn from. To alleviate some of the imbalance in signal-to-noise ratio, sklearn's feature selection method SelectFromModel in tandem with a RandomForestClassifier was implemented. The feature selection was repeated four times with a threshold set to 'median', i.e. any feature deemed to be in the lower half of feature importance is dropped. In other words, the more important half of the features were kept with this repeated four times leaving 31 features for the algorithm to learn from. In the best case scenario, this would leave the 20 informative features and 11 noise features.

In addition the noise issues, the non-linearity of the problem presents an interesting challenge to the learning algorithms. Algorithms without the expressiveness to describe non-linear patterns, e.g. a linear SVM, may struggle on the dataset while others, e.g. an SVM with RBF kernel, may have improved performance.

***Algorithms & Methodology***

All algorithms were implemented via the python machine learning package sci-kit learn.

For each algorithm, the learner was five fold cross validation trained using balanced accuracy as the performance metric across a variety of hyperparameters. Balanced accuracy was chosen in light of the Cars dataset class imbalance, as without the cost balancing most learners would likely not focus on correctly classifying ‘good’ cars as they represent only 8% of the sample. The best parameters were stored, with the best performing classifier then trained on varying amounts of the data with its performance and wall clock time recorded to illustrate its learning curve and computation cost. The variance or 'overfit-ablility' of the iterative learners (ANN, Boosting, SVMs) was tested by measuring the train and test accuracy across an increasing number of iterations using hyper parameters with high expressiveness (i.e. regularization parameters set to very low values).

***Artificial Neural Network***

*Hyperparameters Searched: Activation Function, Learning Rate, Hidden Layer Size*

***Decision Tree***

*Hyperparameters Searched: Splitting Criteria, Learning Rate*

***Boosting***

*Hyperparameters Searched: Number of Estimators, Learning Rate (of base estimator Decision Tree)*

***K Nearest Neighbors***

*Hyperparameters Searched: Distance Metric, Number of Neighbors, Weighting of Neighbors Method*

***SVM***

*Hyper Parameters Searched: Kernel (Linear, RBF), Learning Rate, Number of Iterations*

## Algorithm Analysis

5. Discussion: Algorithms Analysis

\* For each algorithm:

\* Train/Test Error Rates

\* At the end if there's enough time, might want to add

precision, recall, f1 score as metrics on best hyper param models

\* Training Time

\* Learning Rate

\* 'Overfitting' Curves (Expressiveness)

\* Hyperparameter Analysis

\* Why did these come out the best?

Discuss what each parameter does and reasoning for why that performed best.

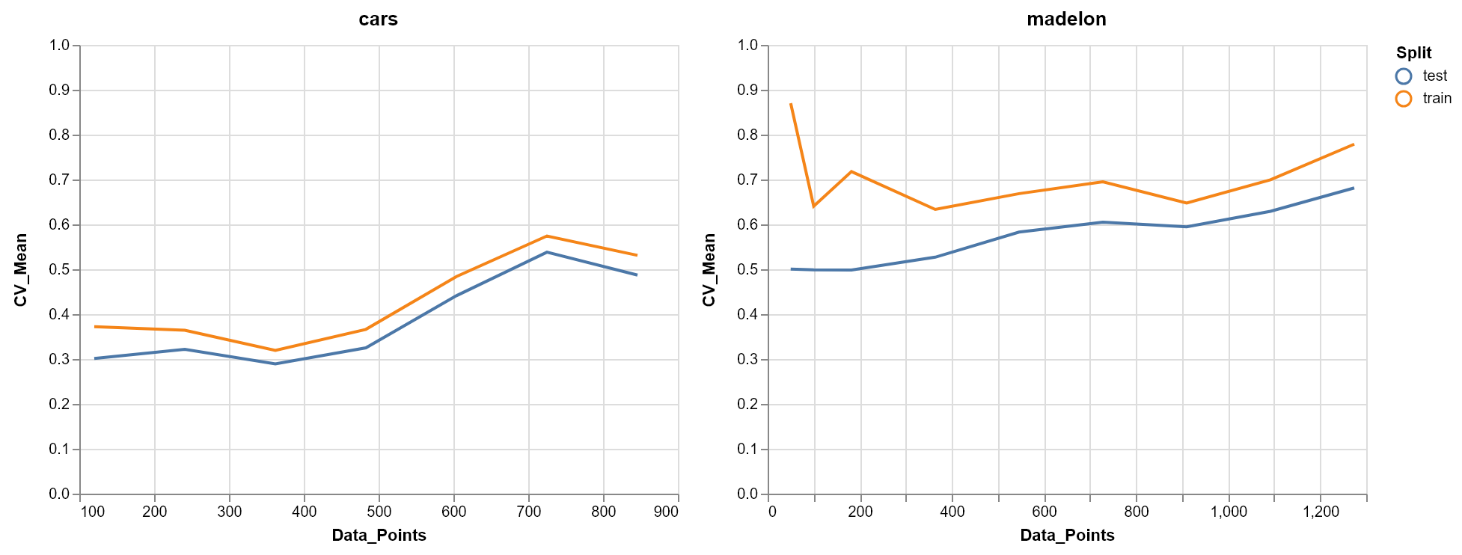
Look at Grid search results if possible to see if distribution of params change

the outcome much (i.e. the effect of param on performance)

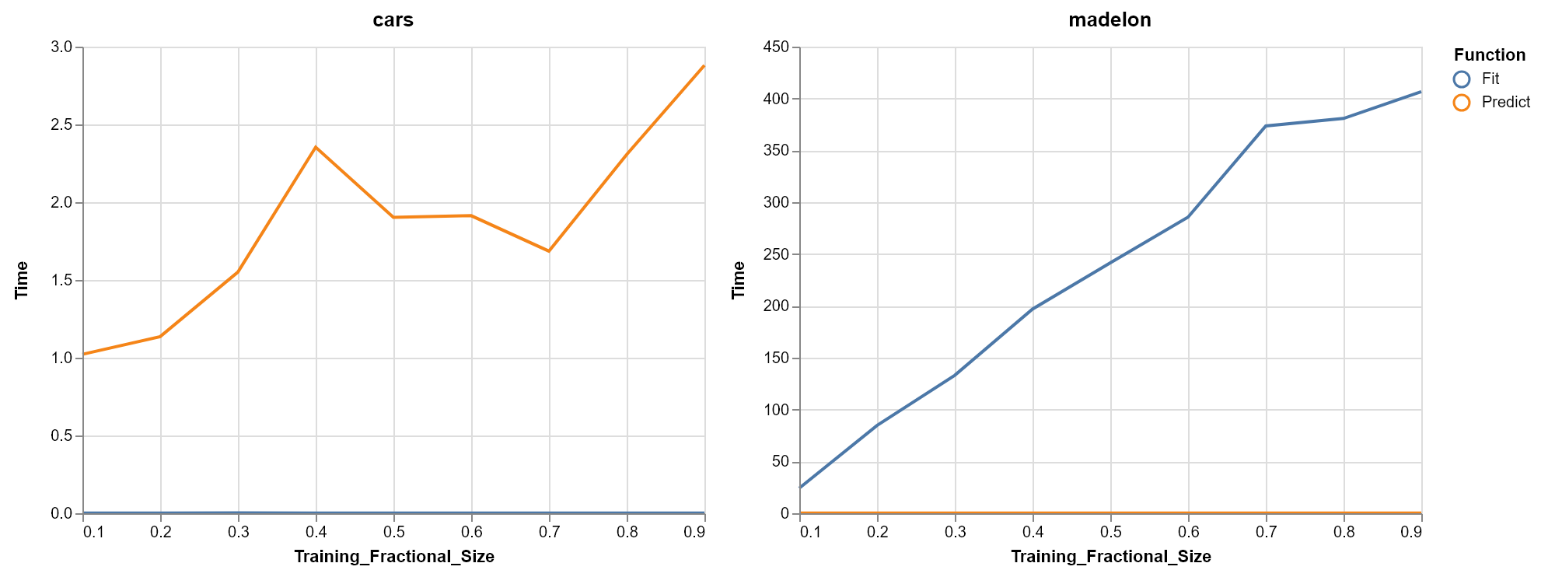
***Artificial Neural Network***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **5 Fold CV Score** | **Activation Function** | **Learning Rate** | **Hidden Layer Size** |
| Cars | .5159 | Relu | .01 | (18, 18, 18) |
| Madelon | .7628 | Relu | .0001 | (62, 62, 62) |

*Learning Curve*



*Timing Curve*



*'Expressiveness’ Curve*

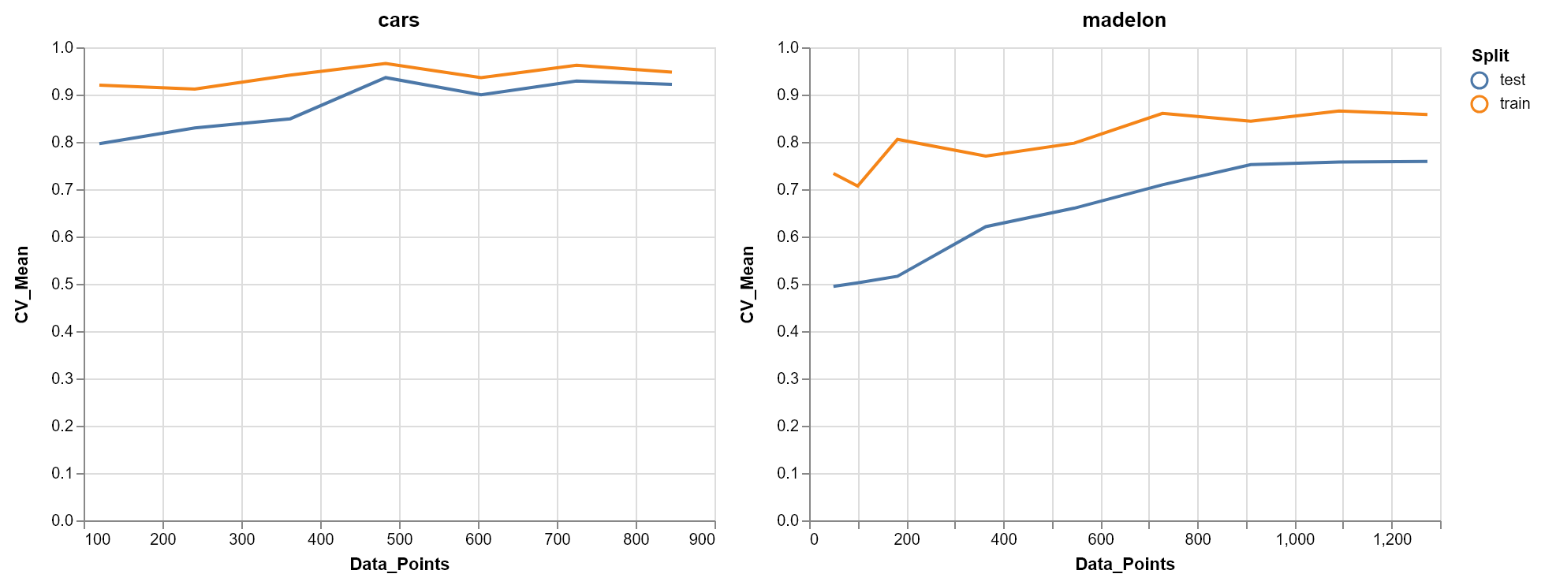
*Cars*

*Madelon*

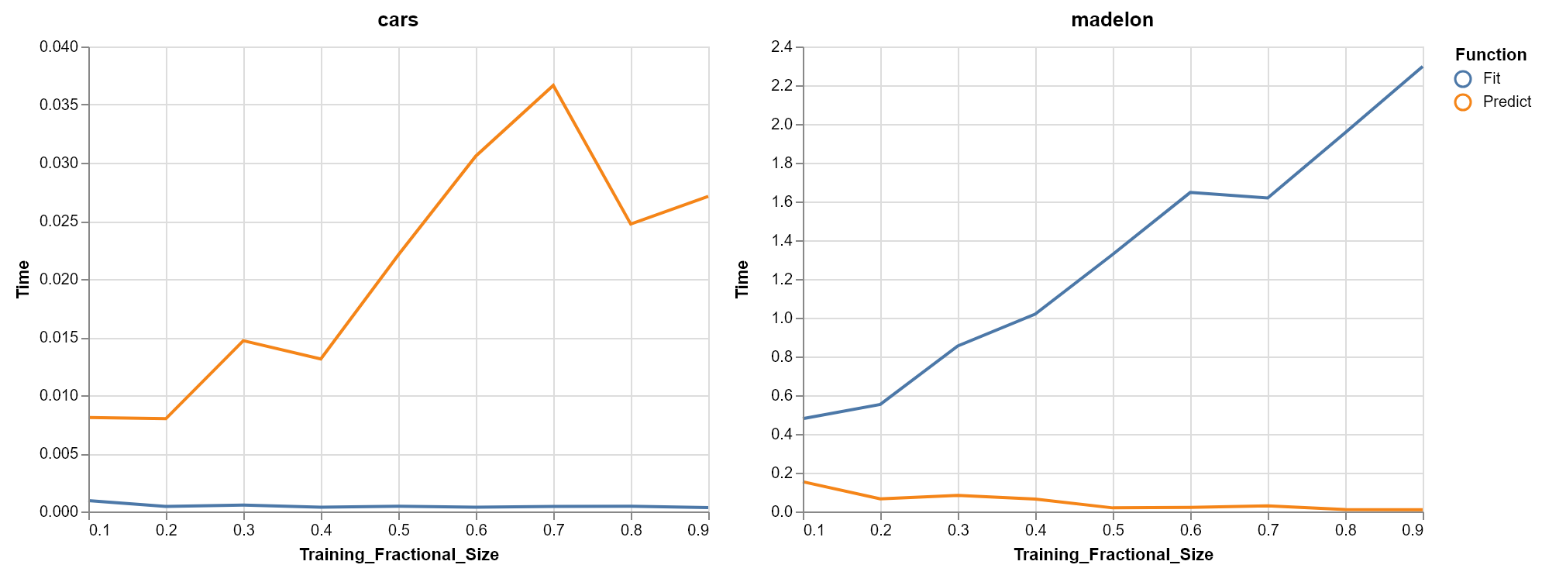
***Decision Tree***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **5 Fold CV Score** | **Splitting Criteria** | **Learning Rate** | **Class Weight** |
| Cars | .9555 | Gini | 0 | Balanced |
| Madelon | .8102 | Gini | .01 | Balanced |

*Learning Curve*



*Timing Curve*



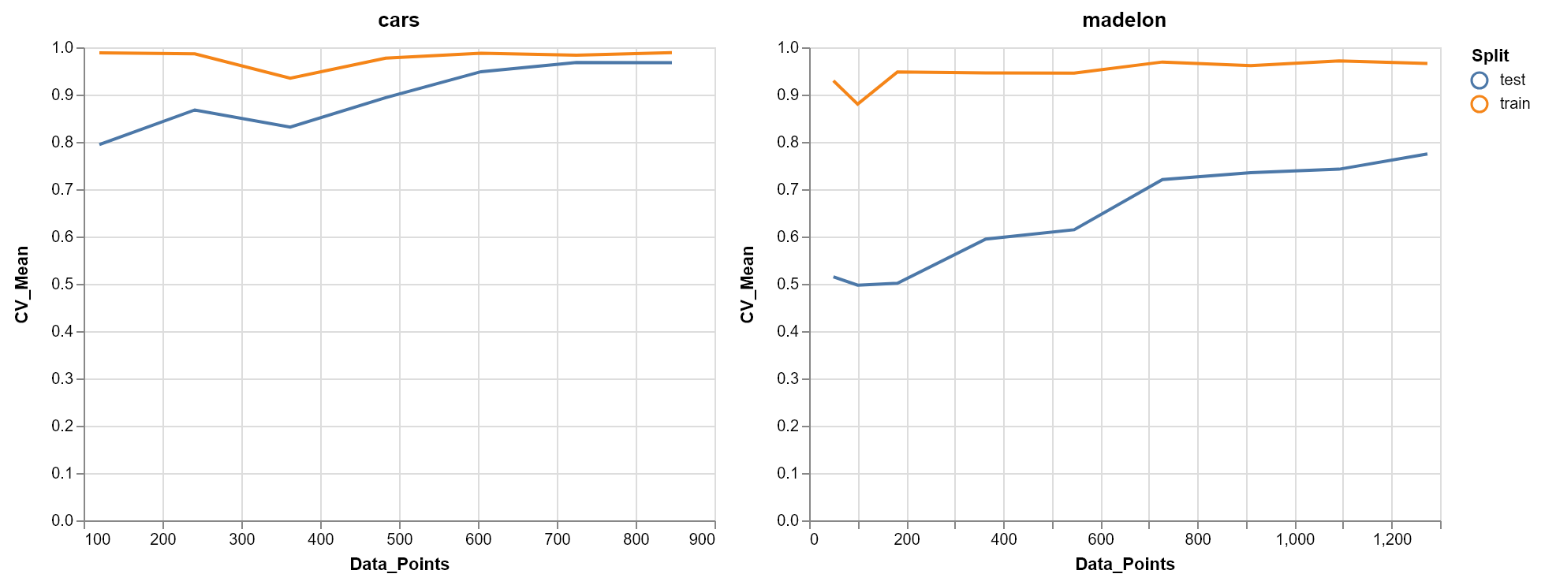
***Boosting***

*Cars*

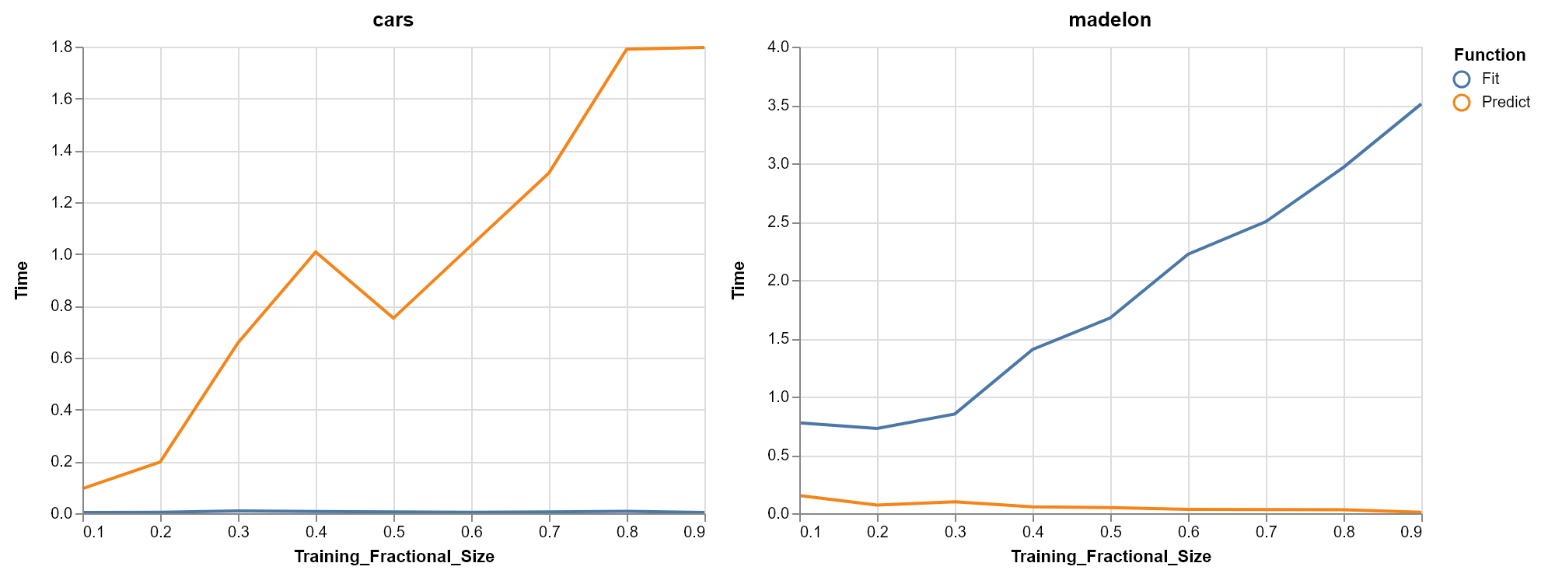
*Madelon*

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **5 Fold CV Score** | **Base Estimator (Decision Tree) Learning Rate** | **Number of Estimators** |
| Cars | .9649 | .0316 | 30 |
| Madelon | .8269 | -.001 | 20 |

*Learning Curve*



*Timing Curve*

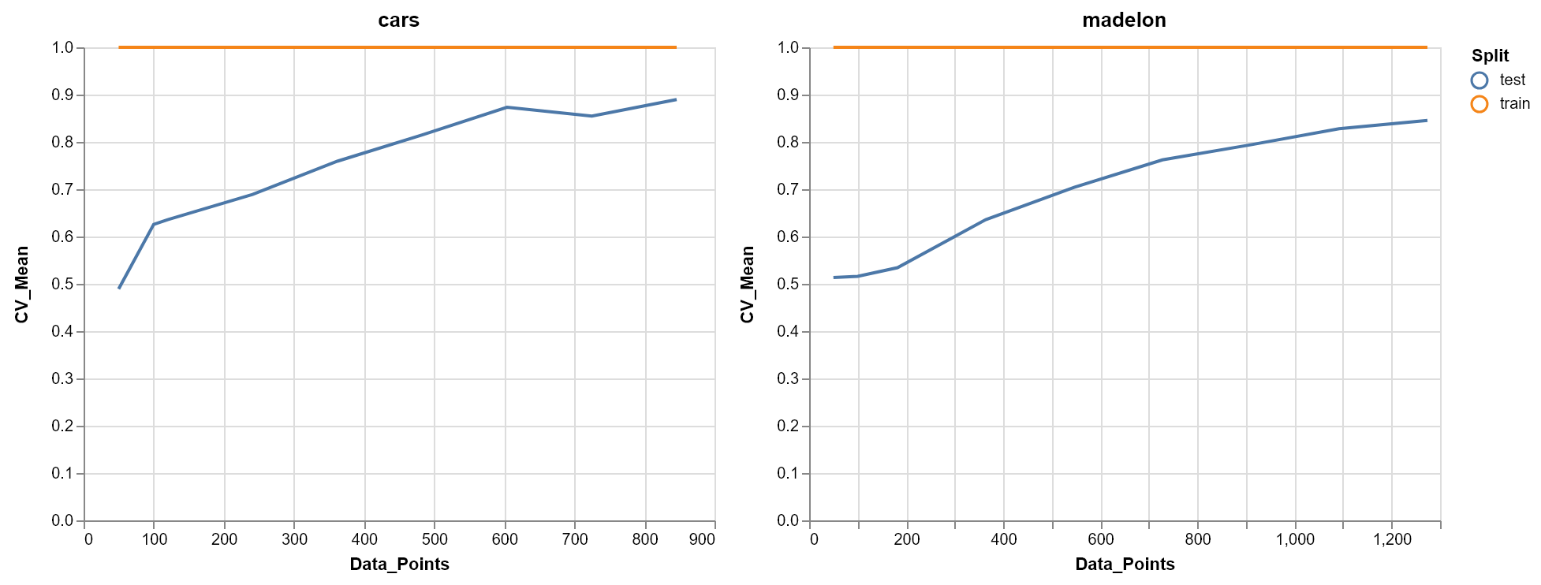


*'Expressiveness’ Curve*

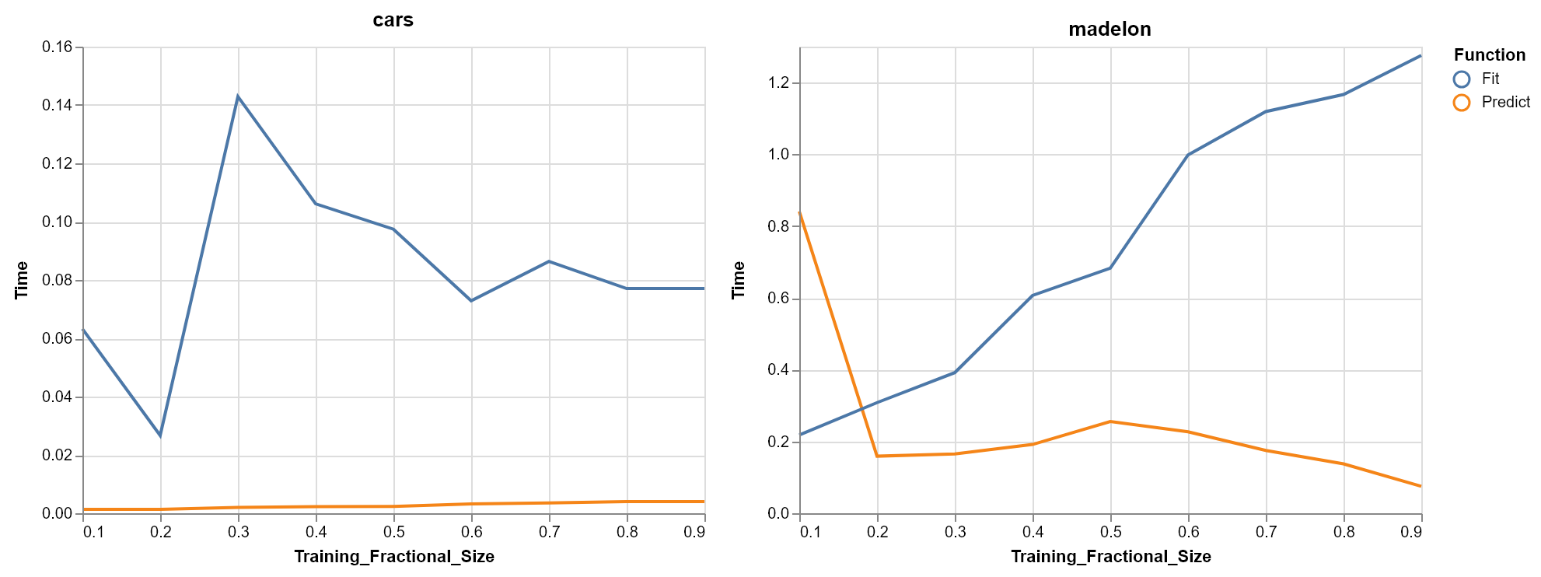
***KNN***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **5 Fold CV Score** | **Distance Metric** | **Number of Neighbours** | **Neighbours Weight** |
| Cars | .9540 | Manhattan | 10 | Distance |
| Madelon | .8641 | Manhattan | 10 | Distance |

*Learning Curve*



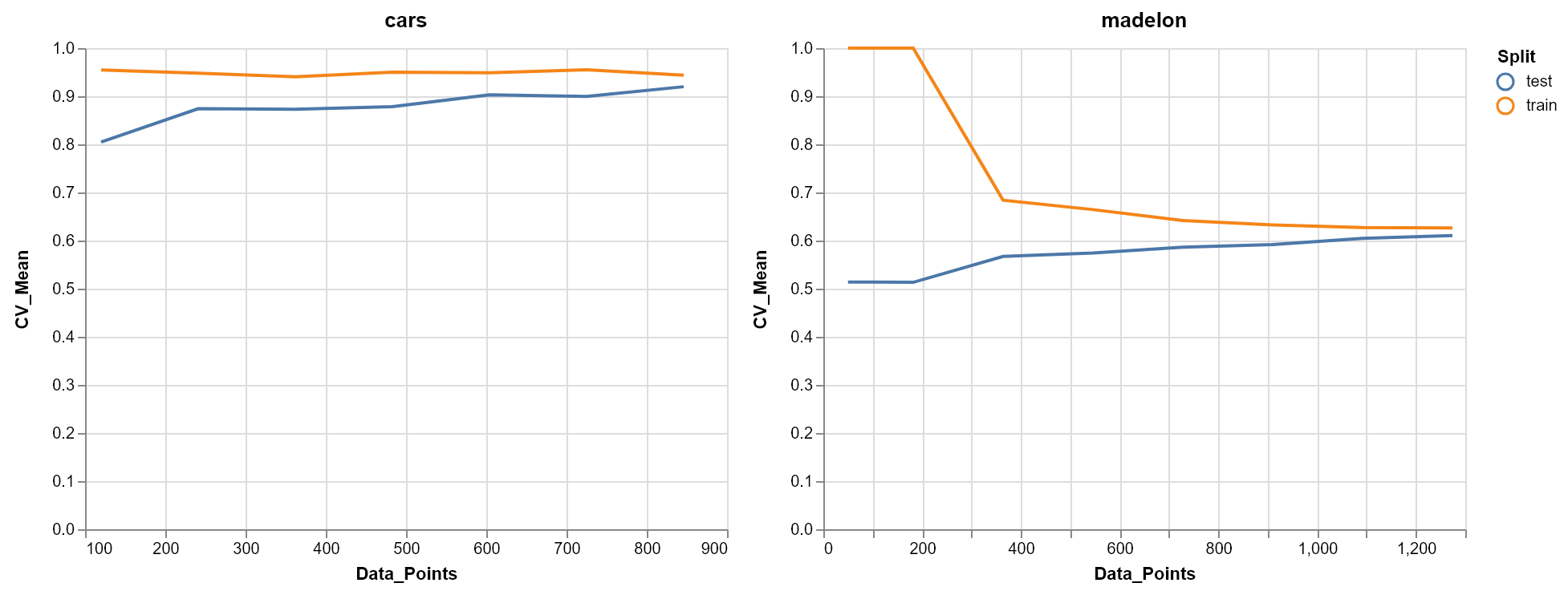
*Timing Curve*



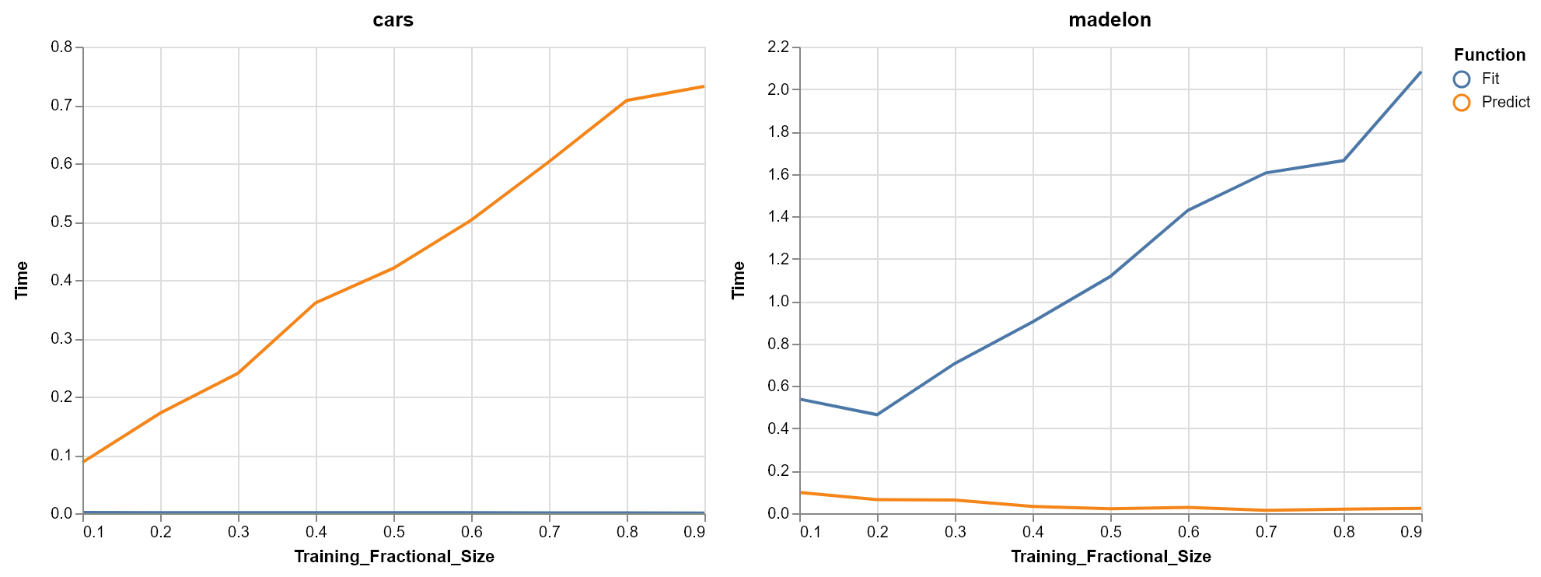
***SVM (Linear Kernel)***

|  |  |  |  |
| --- | --- | --- | --- |
| **Dataset** | **5 Fold CV Score** | **Learning Rate** | **Number of Iterations** |
| Cars | .9007 | .0001 | 1034 |
| Madelon | .6012 | .1 | 687 |

*Learning Curve*



*Timing Curve*



*'Expressiveness’ Curve*

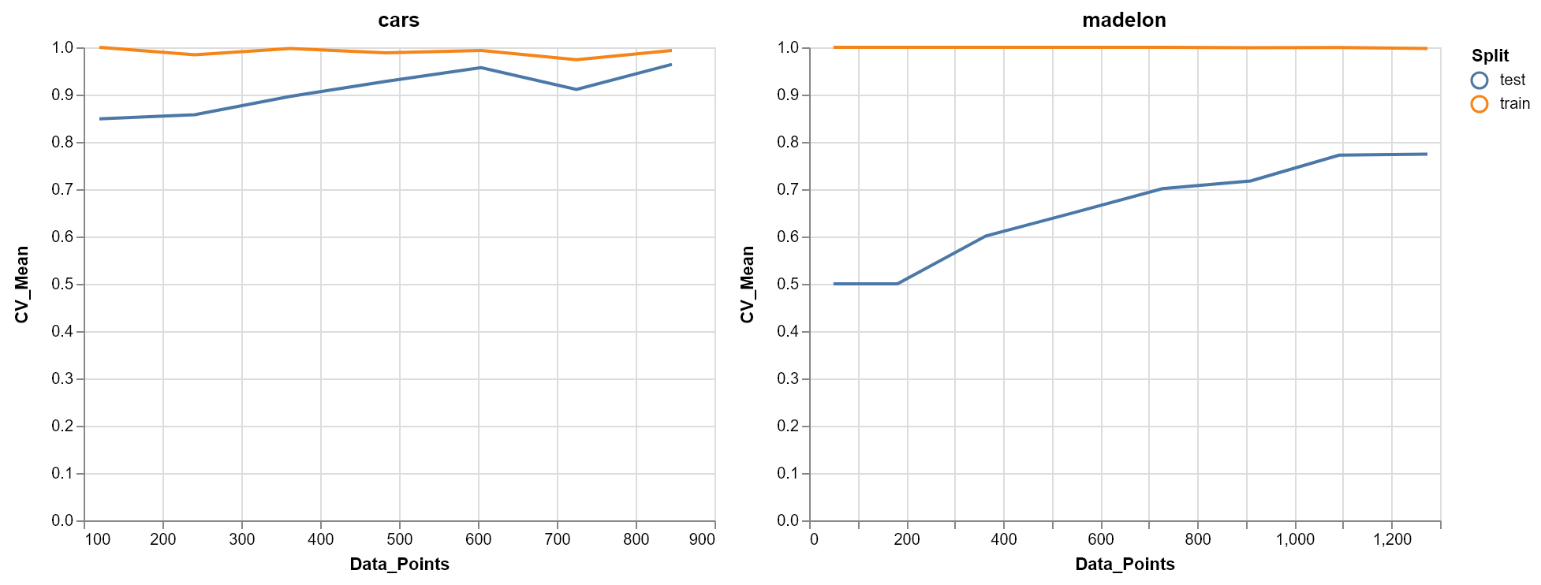
*Cars*

*Madelon*

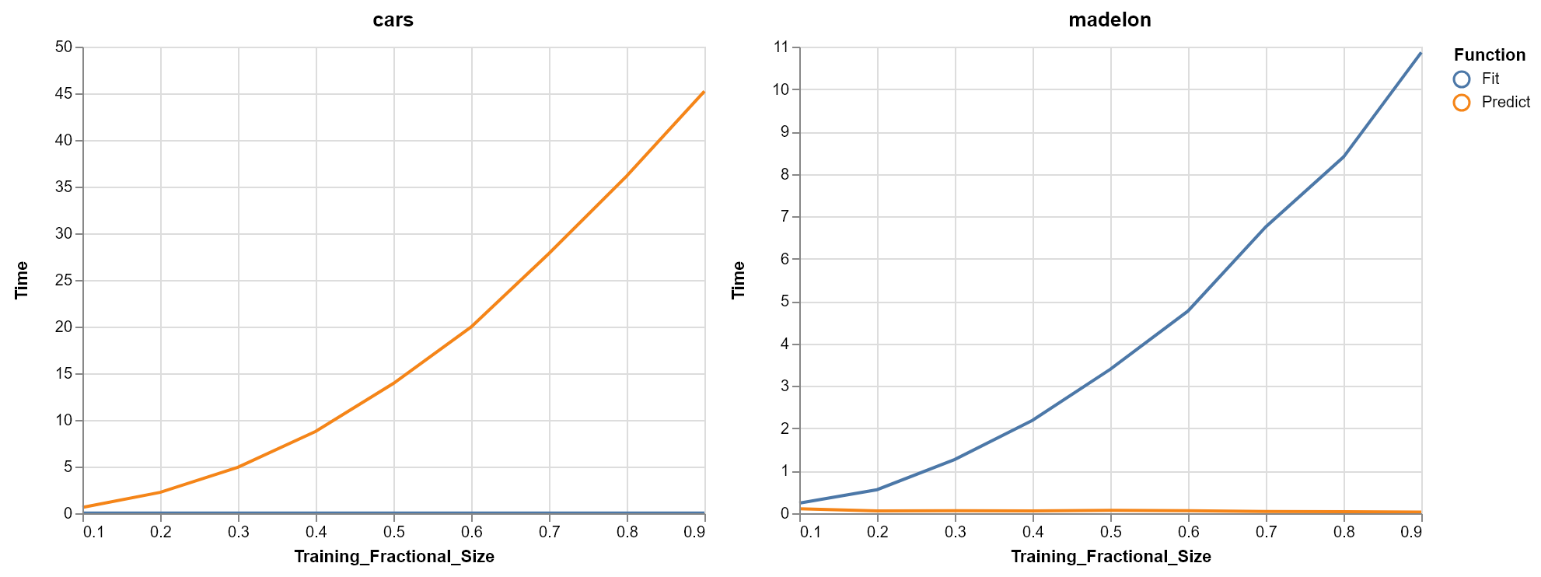
***SVM (RBF Kernel)***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset** | **5 Fold CV Score** | **Learning Rate** | **Number of Iterations** | **Gamma** |
| Cars | .9856 | .0001 | 1034 | .8 |
| Madelon | .8371 | 0.00031622776601683794 | 687 | .15 |

*Learning Curve*



*Timing Curve*



*'Expressiveness’ Curve*

*Cars*

*Madelon*