

**PROJECT REPORT**

on

**Design & Implementation of Neural network models**

Submitted by

|  |  |  |
| --- | --- | --- |
| **Manjeet Maan** | **11702825** | **K17SM** |
| **Vikas Sharma** | **11702757** | **K17SM** |
| **Chanakya Nallapati** | **11702917** | **K17SM** |
| **Abhinav** | **11702986** | **K17SM** |

Submittedto

**Dr. Aditya khamparia**

Program

**B.tech (Computer Science & Engineering)**

School of Computer Science & Engineering

Lovely Professional University, Phagwara

(Sept-Nov,2019)

**DECLARATION**

We hereby declare that the project work entitled “Design and implementation of neural networks” is an authentic record of our own work carried out as requirement of the project for the award of Btech degree in computer science and engineering from Lovely Professional University, Phagwara, Under the guidance of Dr. Aditya khamparia during Sept – Nov 2019. All the information furnished In this project is based on our own intensive work and genuine.

**Project Group**

Manjeet Singh Maan- 11702825

Vikas Sharma- 11702757

Chanakya Nallapati-11702917

Abhinav Patel - 11702986

Date: 10/11/2019

**CERTIFICATE**

This is to certify that the declaration statement made by this group of student is correct to the best of my knowledge and belief. They have completed this project under my guidance and supervision. The present work is the result of their original investigation, effort and study. No part of work has been ever submitted for any other degree at any university. This project is fit for the submission in Soft Computing subject, Computer Science ang Engineering from Lovely Professional University.

**Dr. Aditya Khamparia**

**ACKNOWLEDGEMENT**

We have taken efforts in learning this course. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We express our sincere gratitude to Lovely Professional University for giving us such an opportunity.

I would like to express my gratitude towards my parents for their kind co-operation and encouragement which help me in completion of this course.

I would like to express my special gratitude and thanks to **Dr. Aditya Khamparia** sir that he had told us about what our future in Machine Learning can be. He is the main reason that I started developing interest in Machine Learning.

My thanks and appreciations also go to my colleague who cleared my doubt anytime I have asked.

**Project Group**

Manjeet Singh Maan

Reg. No 11702825

Vikas Sharma

Reg. No. 11702757

Chanakya Nallapati

Reg. No.11702917

Abhinav Patel -11702986

**TABLE OF CONTENT**

Main Pge…………………………………………………………………………………...1

Declaration…………………………………………………………………………...............2

Acknowledgement……………………………………………………………………………3

Table of Contents…………………………………………………………………………….4

1. INTRODUCTION………………………………………………………………………….7
2. TYPES OF NEURAL NETWORK……………….………………………………………...8
3. a) Perceptron

b) SVM

c) LVQ

d) SOM

1. ScreenShots of Projects

INTRODUCTION

TOMACHINE LEARNING

1. **INTRODUCTION**

**1.1 WHAT IS MACHINE LEARNING**

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people. Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range.

* 1. **NEED OF MACHINE LEARNING**

Machine learning is needed for tasks that are too complex for humans to code directly. Some tasks are so complex that it is impractical, if not impossible, for humans to work out all of the nuances and code for them explicitly. So instead, we provide a large amount of data to a machine learning algorithm and let the algorithm work it out by exploring that data and searching for a model that will achieve what the programmers have set it out to achieve.

* 1. **TYPES OF MACHINE LEARNING**

There some variations of how to define the types of Machine Learning Algorithms but commonly they can be divided into categories according to their purpose and the main categories are the following:

* Supervised learning
* Unsupervised Learning
* Semi-supervised Learning
* Reinforcement Learning

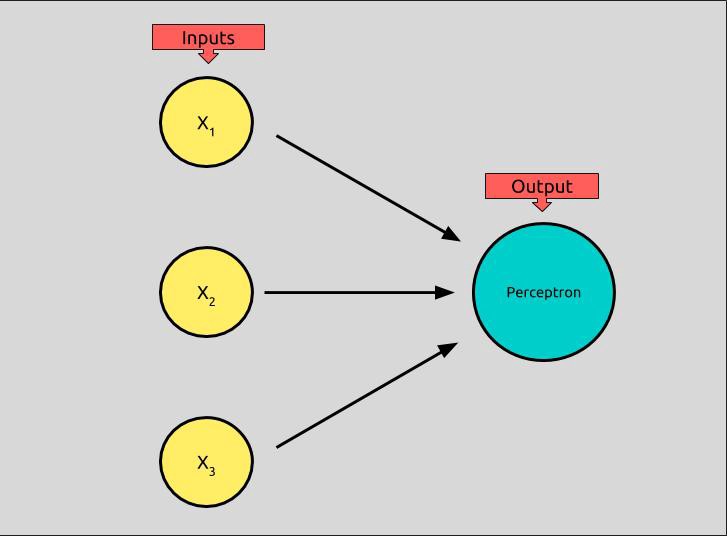
1. **SOME OF THE MOST IMPORTANT TYPE OF NEURAL NETWORK MODELS ARE :-**
2. **Perceptron**

**b) SVM**

**c) LVQ**

**d) SOM**

(a). PERCEPTRON

Perceptrons are a type of artificial neuron that predates the sigmoid neuron. It appear that they were invented in 1957 by [Frank Rosenblatt](https://en.wikipedia.org/wiki/Frank_Rosenblatt) at the Cornell Aeronautical Laboratory.The initial difference between sigmoids and perceptrons, as I understand it, is that perceptrons deal with binary inputs and outputs exclusively.Taken from [Michael Nielsen](http://michaelnielsen.org/)’s [Neural Networks and Deep Learning](http://neuralnetworksanddeeplearning.com/index.html) we can model a perceptron that has 3 inputs like this:

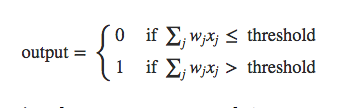
A perceptron can have any number of inputs, but this one has three binary inputs x¹, x², and x³, and produces a binary output, which is called its activation.

How can we take three binary inputs and produce one binary output? First, we assign each input a weight, loosely meaning the amount of influence the input has over the output.

In the picture above, weights are illustrated by black arrows. We’ll call each weight w. Each input, x above has an associated weight: x¹ has a weight w¹, x² a weight of w², and x³, a weight of w³.

To determine the perceptron’s activation, we take the weighted sum of each of the inputs and then determine if it is above or below a certain threshold, or bias, represented by b.

The formula for perceptron neurons can can be expressed like this:



<http://neuralnetworksanddeeplearning.com/index.html>

Let’s break this down.

* output is the output of our formula, which is called the *activation* of our perceptron.
* Both if branches start with the same ∑ formula which takes each input, x, multiplies it by its weight, w, and then add them all together. This is the *weighted sum,*in our case, x¹w¹ + x²w² + x³w³, which can also be, (and usually is), represented using dot product notation.
* If the *weighted sum* is less than or equal to our *threshold*, or *bias*, b, then our output will be 0
* If the *weighted sum* is greater than our *threshold*, or *bias*, b, then our output will be 1

It’s more common to represent the perceptron math like this:

https://miro.medium.com/max/239/1*T_mQVKH0PKS97waJ-RkDYg.png

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

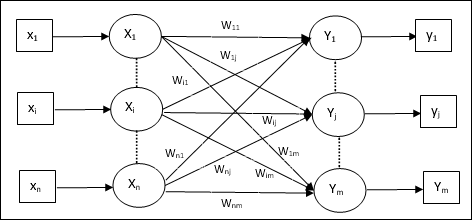
* The summation is represented using dot product notation.
* The “threshold” is moved to the other side of the equality and labeled b for “bias.”
* The summation and bias are added together and compared to to 0.

This new way of comparing to 0, offers us a new way of thinking about these artificial neurons. We can think of the bias, now, like a predictor of how easily our neuron will activate, or produce 1 as an output. A neuron with a large biases will indicate that it will “fire” more easily than the same neuron with a smaller bias.

**(b) LVQ (LINEAR VECTOR QUANTIZATION)**

Learning Vector Quantization LVQLVQ, different from Vector quantization VQVQ and Kohonen Self-Organizing Maps KSOMKSOM, basically is a competitive network which uses supervised learning. We may define it as a process of classifying the patterns where each output unit represents a class. As it uses supervised learning, the network will be given a set of training patterns with known classification along with an initial distribution of the output class. After completing the training process, LVQ will classify an input vector by assigning it to the same class as that of the output unit.

Following figure shows the architecture of LVQ which is quite similar to the architecture of KSOM. As we can see, there are **“n”** number of input units and **“m”** number of output units. The layers are fully interconnected with having weights on them.



Parameters Used

Following are the parameters used in LVQ training process as well as in the flowchart

* **x** = training vector (x1,...,xi,...,xn)
* **T** = class for training vector **x**
* **wj** = weight vector for **jth** output unit
* **Cj** = class associated with the **jth** output unit

Training Algorithm

**Step 1** − Initialize reference vectors, which can be done as follows −

* **Step 1**aa − From the given set of training vectors, take the first “**m**” numberofclustersnumberofclusters training vectors and use them as weight vectors. The remaining vectors can be used for training.
* **Step 1**bb − Assign the initial weight and classification randomly.
* **Step 1**cc − Apply K-means clustering method.

**Step 2** − Initialize reference vector αα

**Step 3** − Continue with steps 4-9, if the condition for stopping this algorithm is not met.

**Step 4** − Follow steps 5-6 for every training input vector **x**.

**Step 5** − Calculate Square of Euclidean Distance for **j = 1 to m** and **i = 1 to n**

D(j)=∑i=1n∑j=1m(xi−wij)2D(j)=∑i=1n∑j=1m(xi−wij)2

**Step 6** − Obtain the winning unit **J** where **D**jj is minimum.

**Step 7** − Calculate the new weight of the winning unit by the following relation −

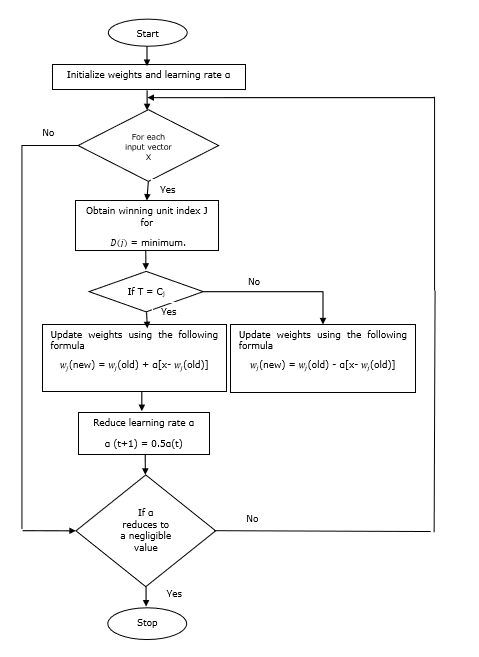
if **T = Cj** then wj(new)=wj(old)+α[x−wj(old)]wj(new)=wj(old)+α[x−wj(old)]

if **T ≠ Cj** then wj(new)=wj(old)−α[x−wj(old)]wj(new)=wj(old)−α[x−wj(old)]

**Step 8** − Reduce the learning rate αα.

**Step 9** − Test for the stopping condition. It may be as follows −

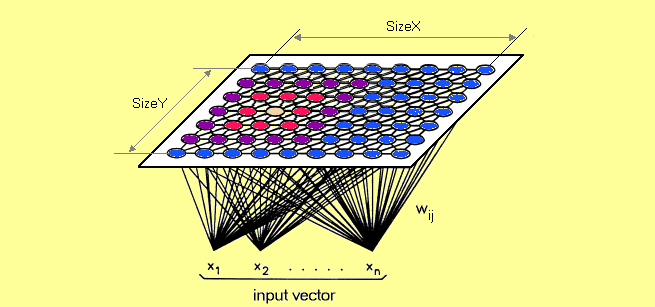
* Maximum number of epochs reached.
* Learning rate reduced to a negligible value

.

**(c) SOM (SELF ORGANIZING MAP)**

So far we have looked at networks with supervised training techniques, in which there is a target output for each input pattern, and the network learns to produce the required outputs. We now turn to unsupervised training, in which the networks learn to form their own classifications of the training data without external help. To do this we have to assume that class membership is broadly defined by the input patterns sharing common features, and that the network will be able to identify those features across the range of input patterns. One particularly interesting class of unsupervised system is based on competitive learning, in which the output neurons compete amongst themselves to be activated, with the result that only one is activated at any one time. This activated neuron is called a winner-takesall neuron or simply the winning neuron. Such competition can be induced/implemented by having lateral inhibition connections (negative feedback paths) between the neurons. The result is that the neurons are forced to organise themselves. For obvious reasons, such a network is called a Self Organizing Map (SOM)

The principal goal of an SOM is to transform an incoming signal pattern of arbitrary dimension into a one or two dimensional discrete map, and to perform this transformation adaptively in a topologically ordered fashion. We therefore set up our SOM by placing neurons at the nodes of a one or two dimensional lattice. Higher dimensional maps are also possible, but not so common. The neurons become selectively tuned to various input patterns (stimuli) or classes of input patterns during the course of the competitive learning. The locations of the neurons so tuned (i.e. the winning neurons) become ordered and a meaningful coordinate system for the input features is created on the lattice. The SOM thus forms the required topographic map of the input patterns. We can view this as a non-linear generalization of principal component analysis (PCA).



**Overview of the SOM Algorithm**

We have a spatially continuous input space, in which our input vectors live. The aim is to map from this to a low dimensional spatially discrete output space, the topology of which is formed by arranging a set of neurons in a grid. Our SOM provides such a nonlinear transformation called a feature map. The stages of the SOM algorithm can be summarised as follows:

1. Initialization – Choose random values for the initial weight vectors wj .

2. Sampling – Draw a sample training input vector x from the input space.

3. Matching – Find the winning neuron I(x) with weight vector closest to input vector.

4. Updating – Apply the weight update equation ∆wji = η t Tj I t xi − wji ( ) ( ) ( ) , (x) . 5. Continuation – keep returning to step 2 until the feature map stops changing. Next lecture we shall explore the properties of the resulting feature map and look at some simple examples of its application.

**(d) SVM (SUPPORT VECTOR MACHINE)**

The NSVM (see Figure 1a) consists of: (1) an input layer consisting of D nodes; (2) a central feature layer z consisting of d nodes; (3) a total of d two-layer neural networks (MLPs) N, which each take the entire input layer as their input and produce one of the feature values as their output, and (4) a main support vector regression model M that takes the entire feature layer as its input and determines the value of the output node. When a pattern x of dimension D is presented to the NSVM, it is propagated through the neural networks, determining the values of the feature layer. We use Φ(x|θ) to denote the mapping performed by the neural networks, i.ez = Φ(x|θ). Here, Φ : R D → R d and θ is a vector containing all the weights of the neural networks. The representation in the feature layer is used as input for the support vector machine M that determines the value of the output node. The regression NSVM computes its output using:

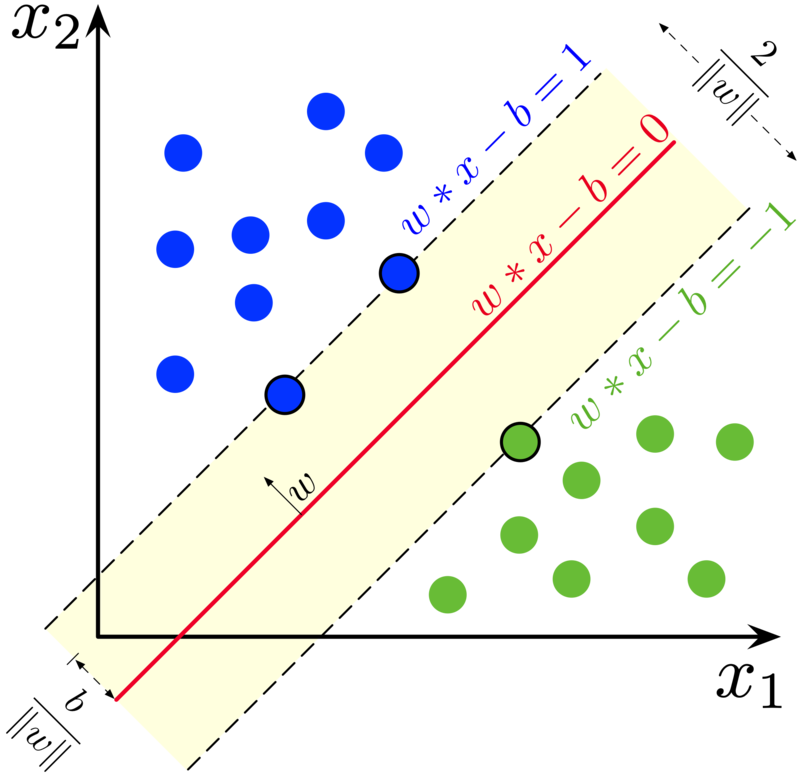
f(x) = X ` i=1 (α ∗ i − αi)K(Φ(xi |θ), Φ(x|θ)) + b

Where K(·, ·) is the kernel function of the main SVM.

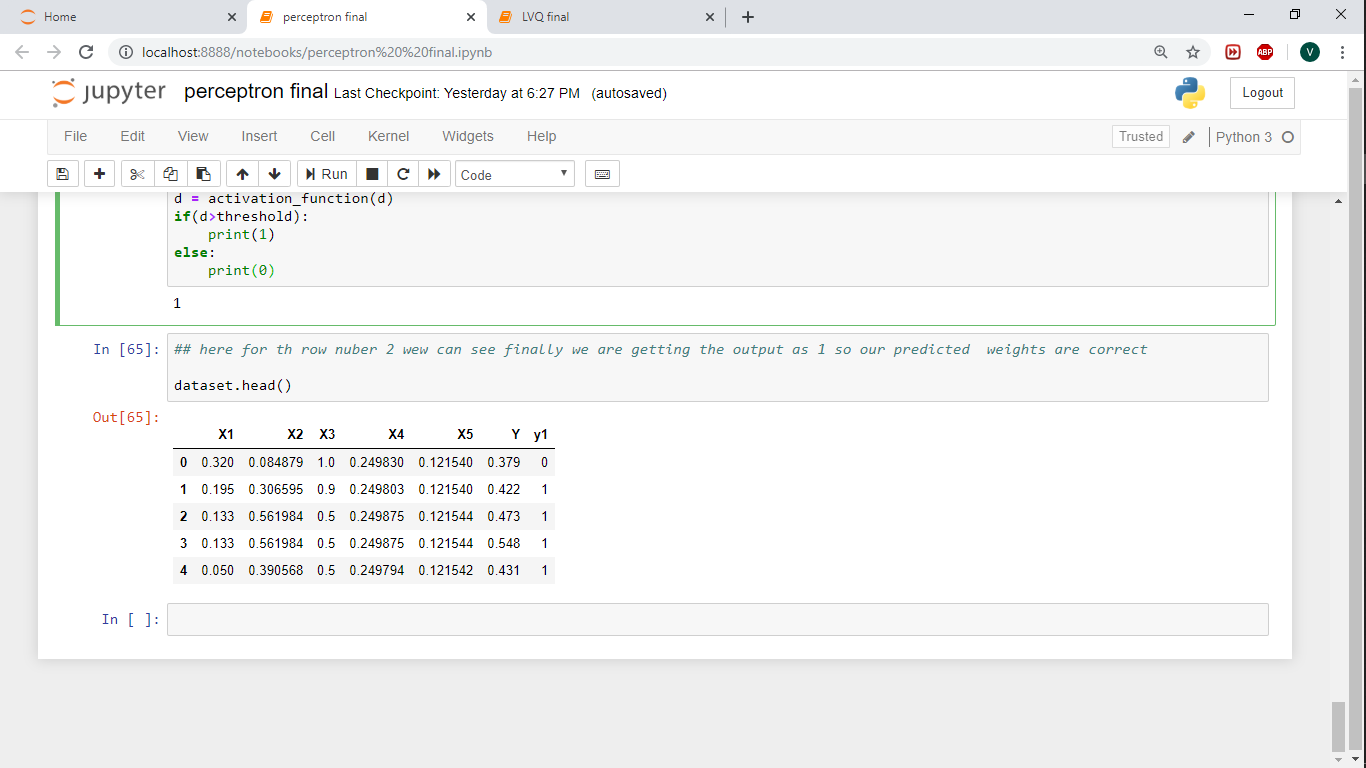
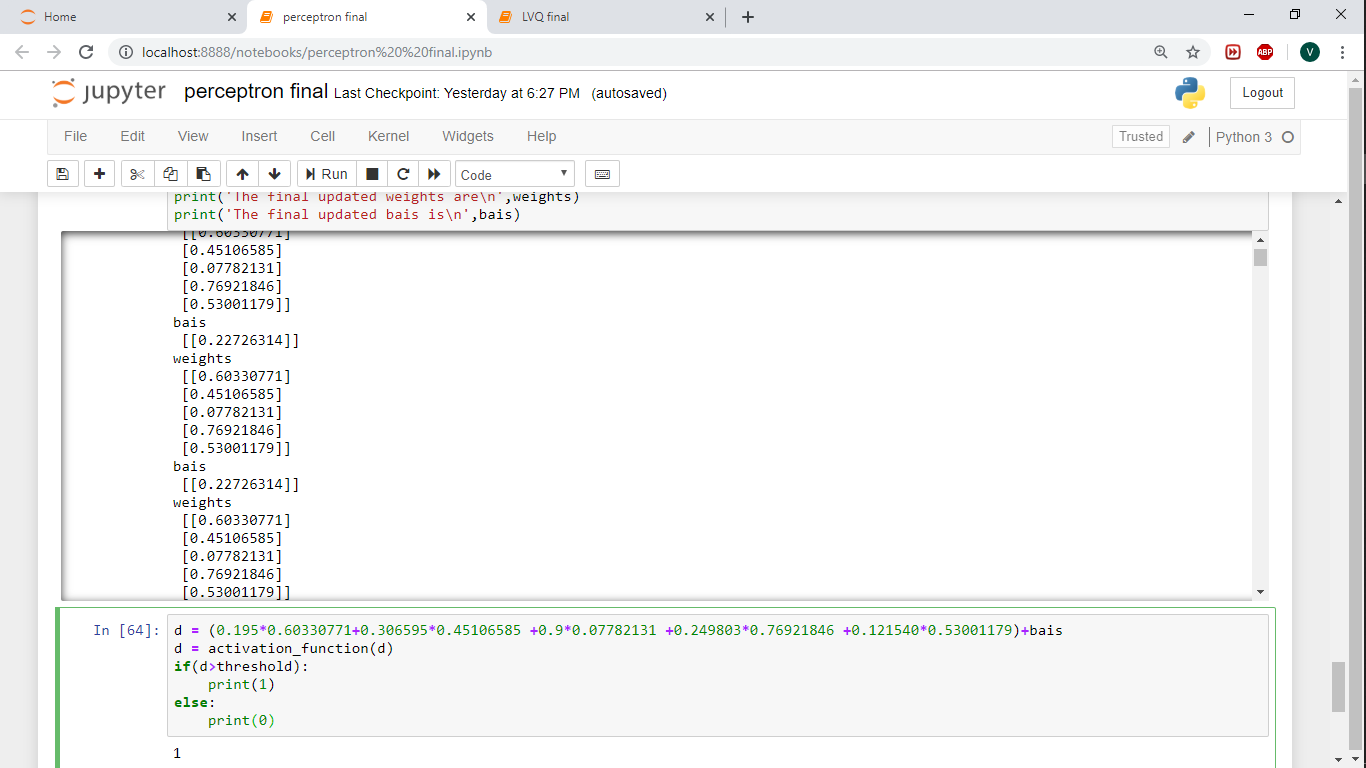
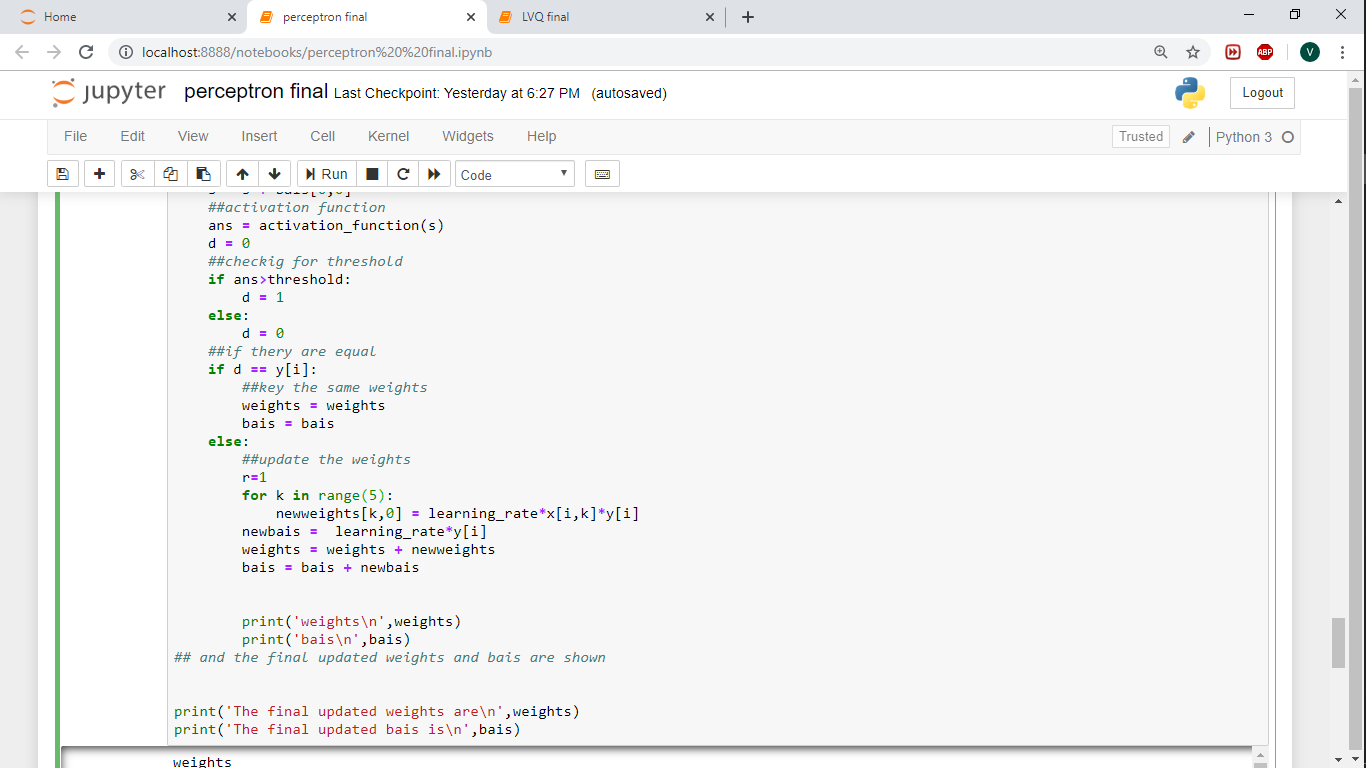
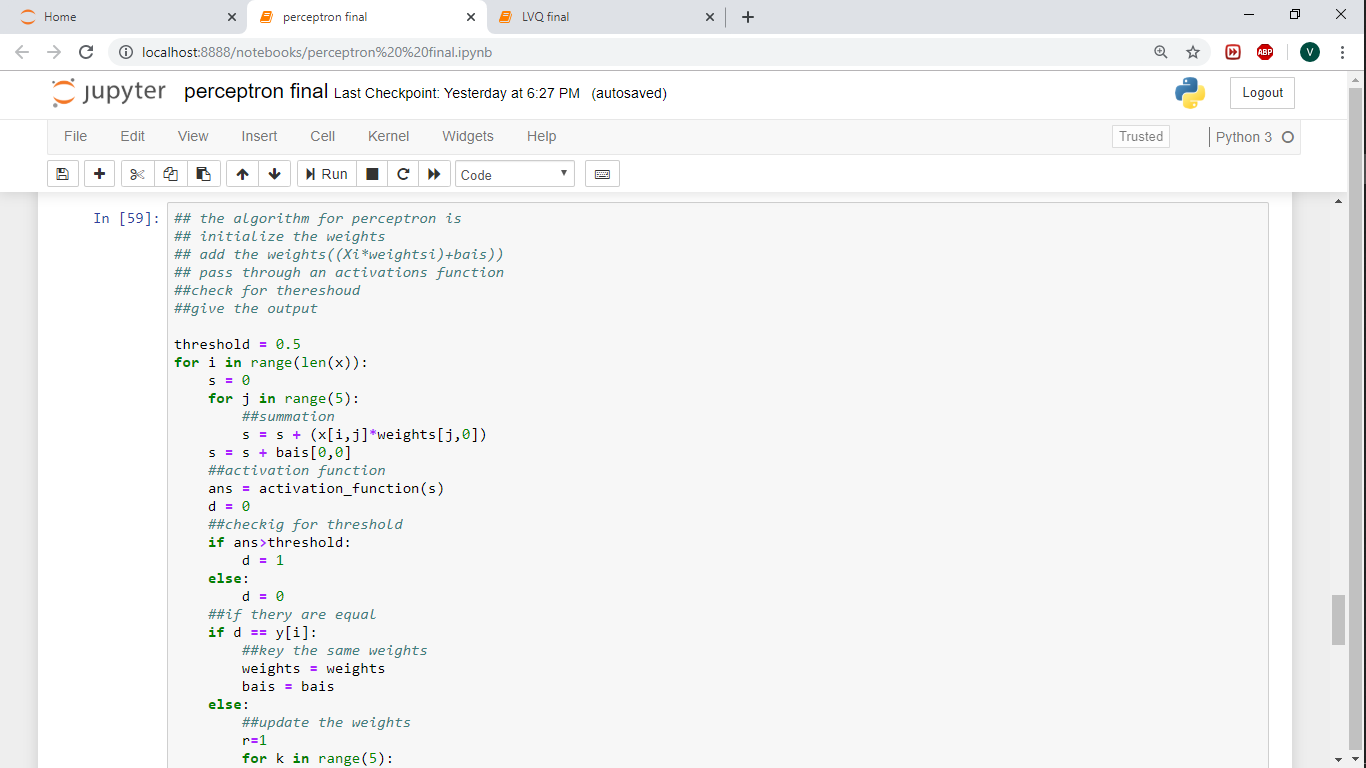
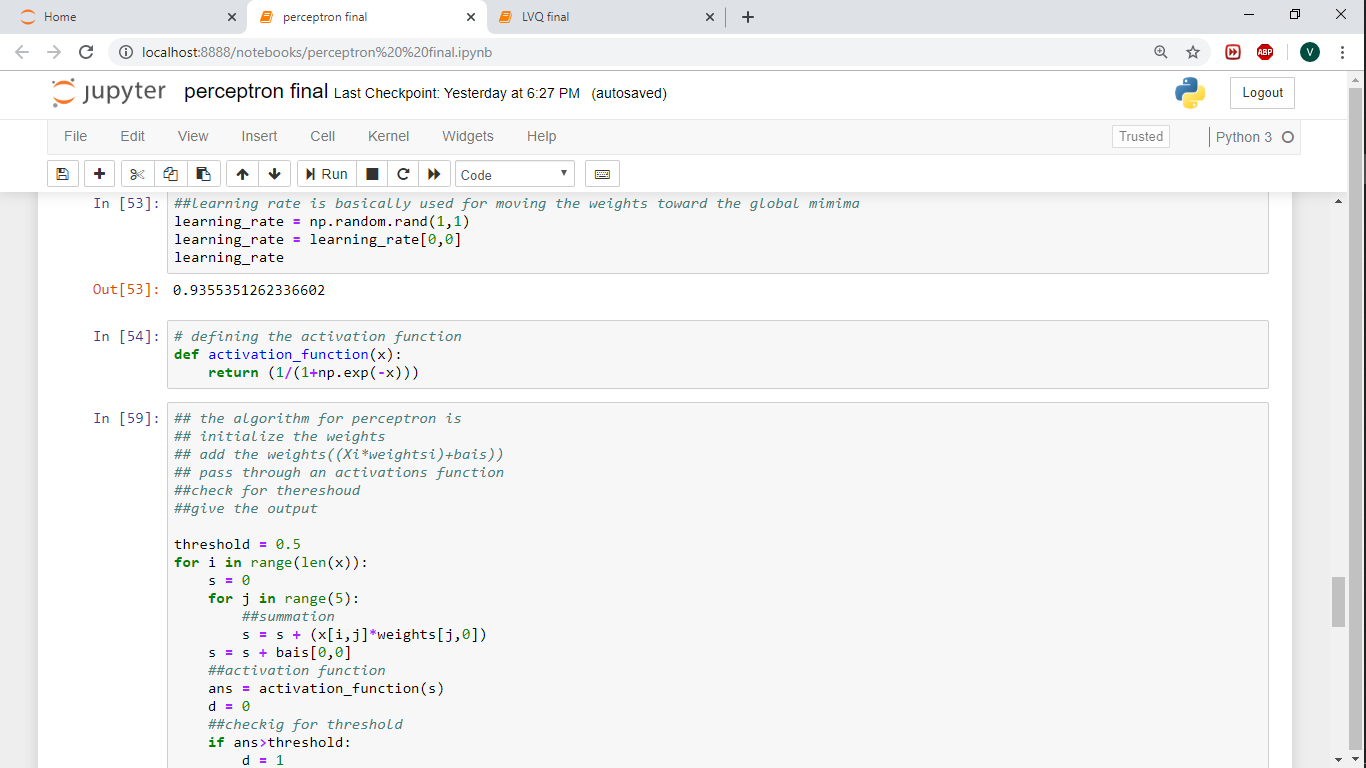
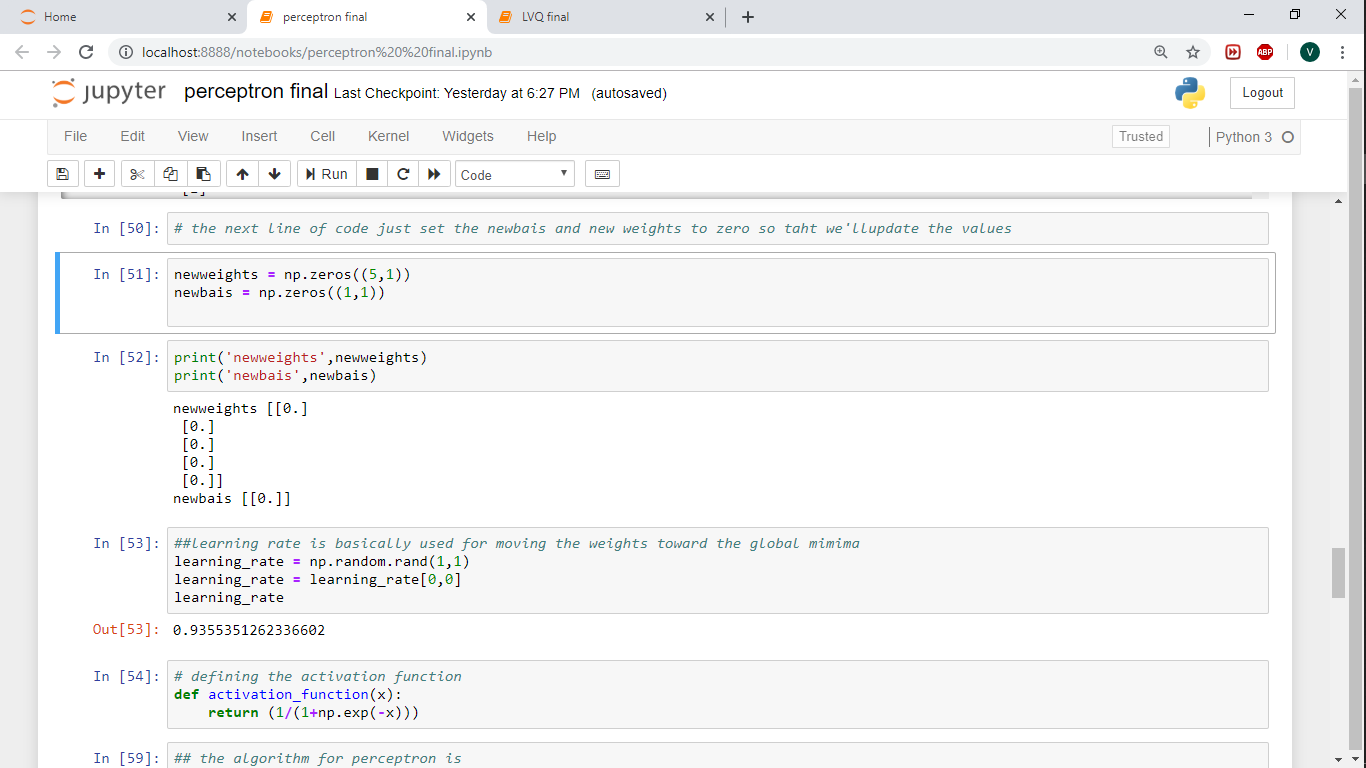
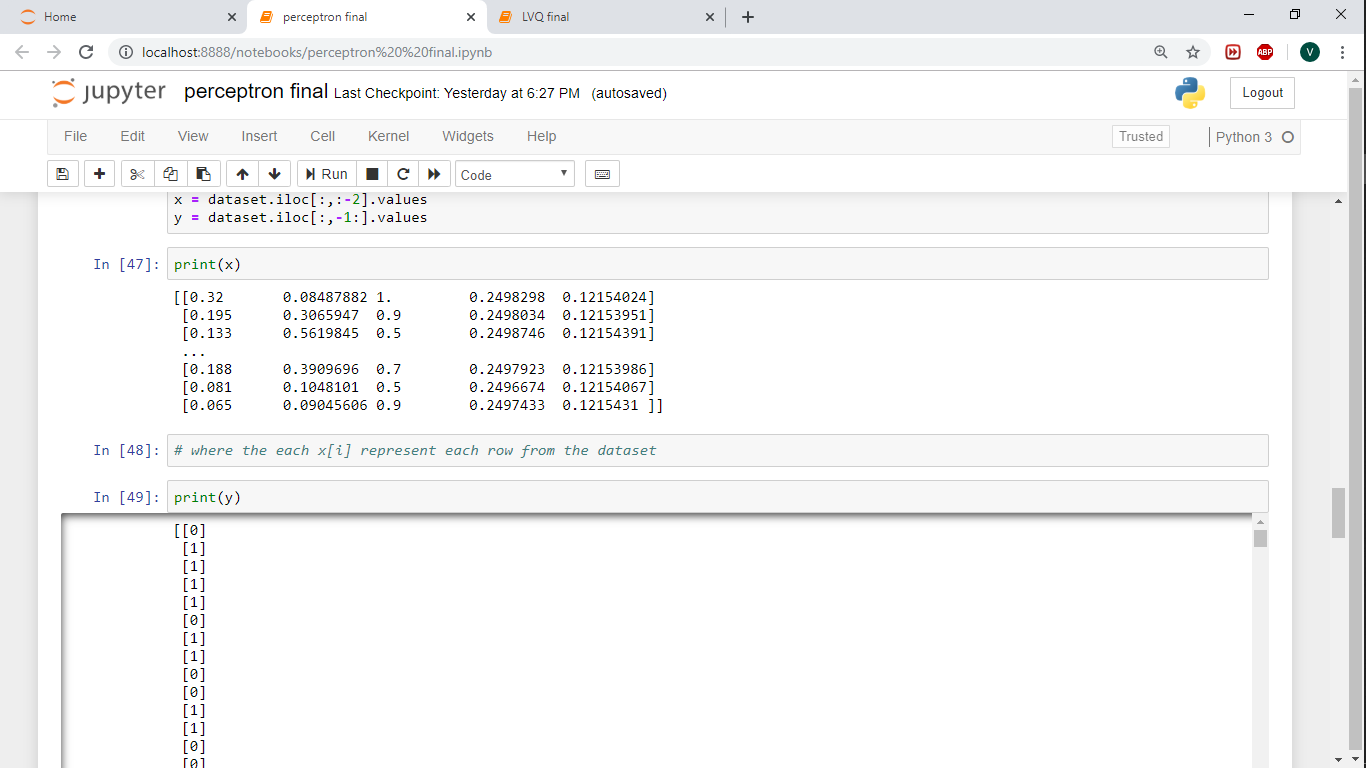
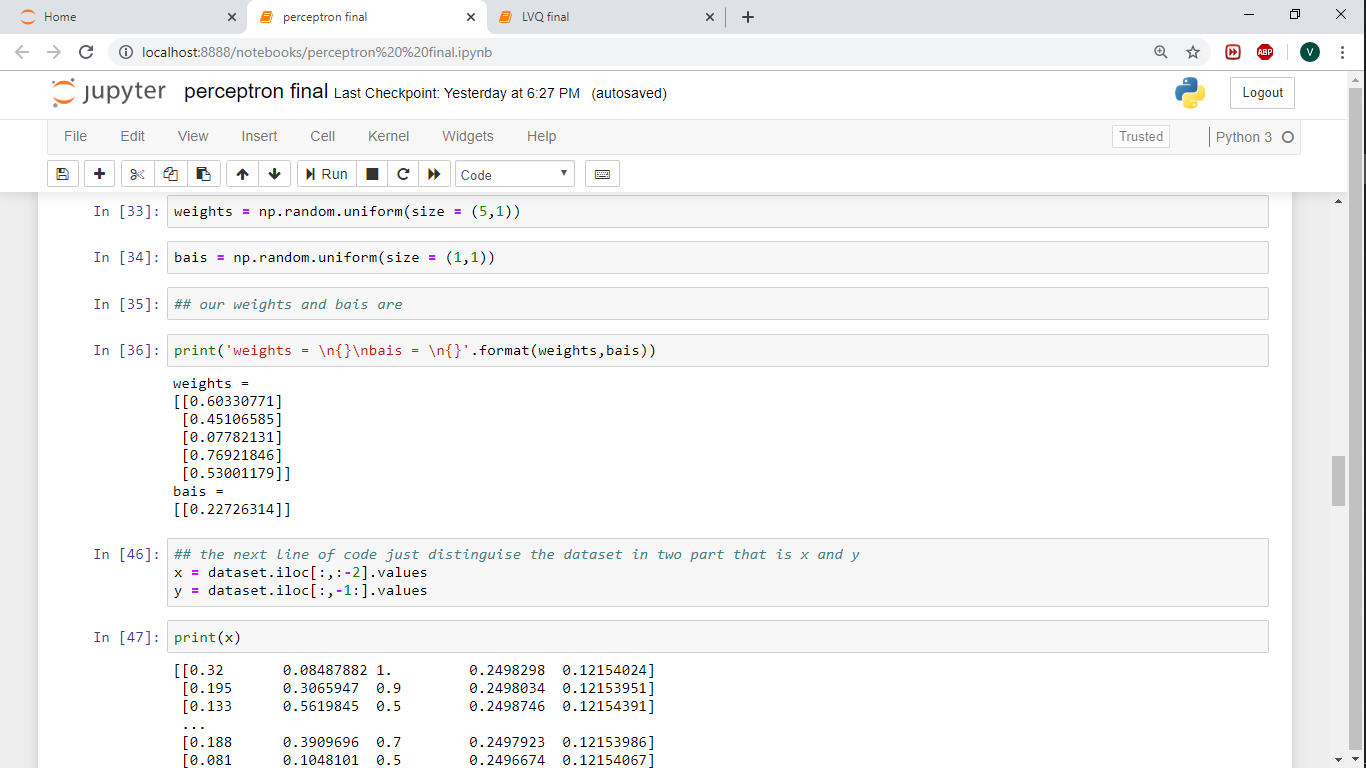
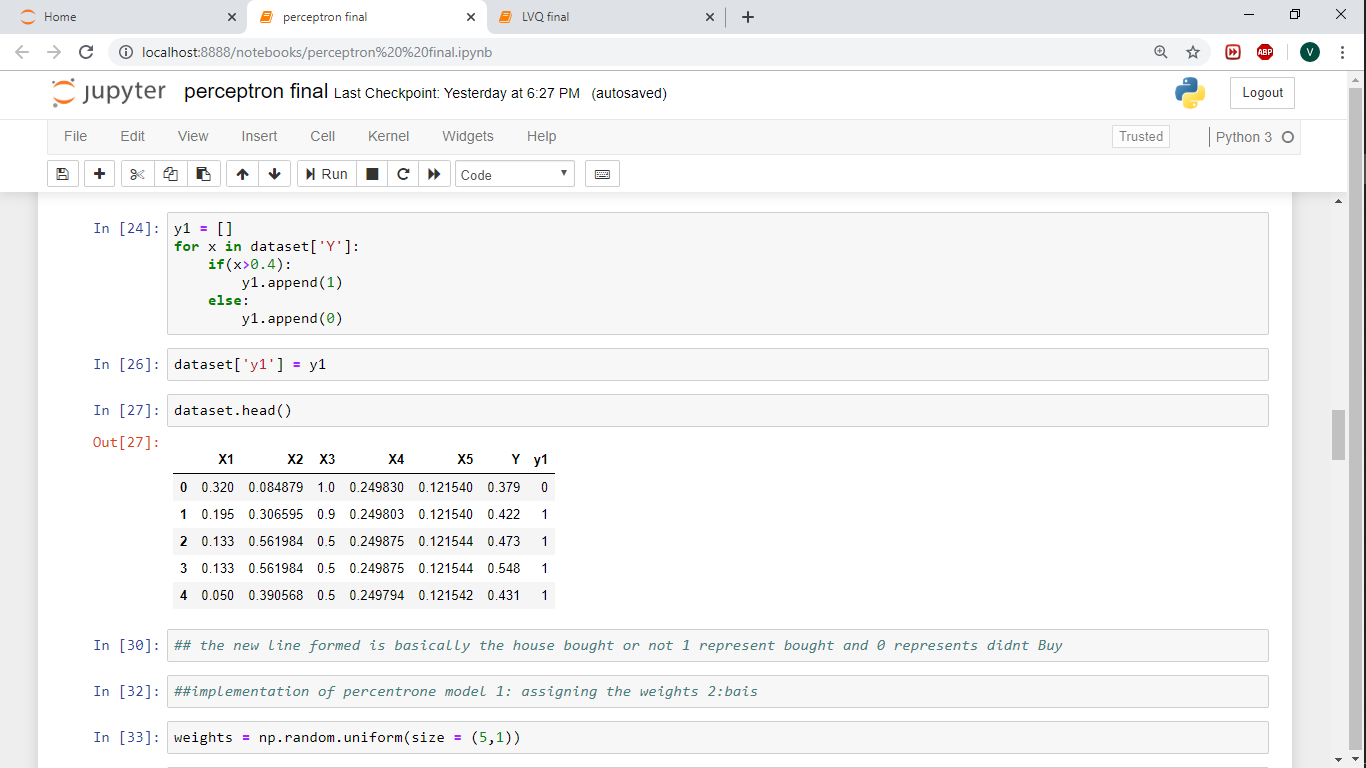
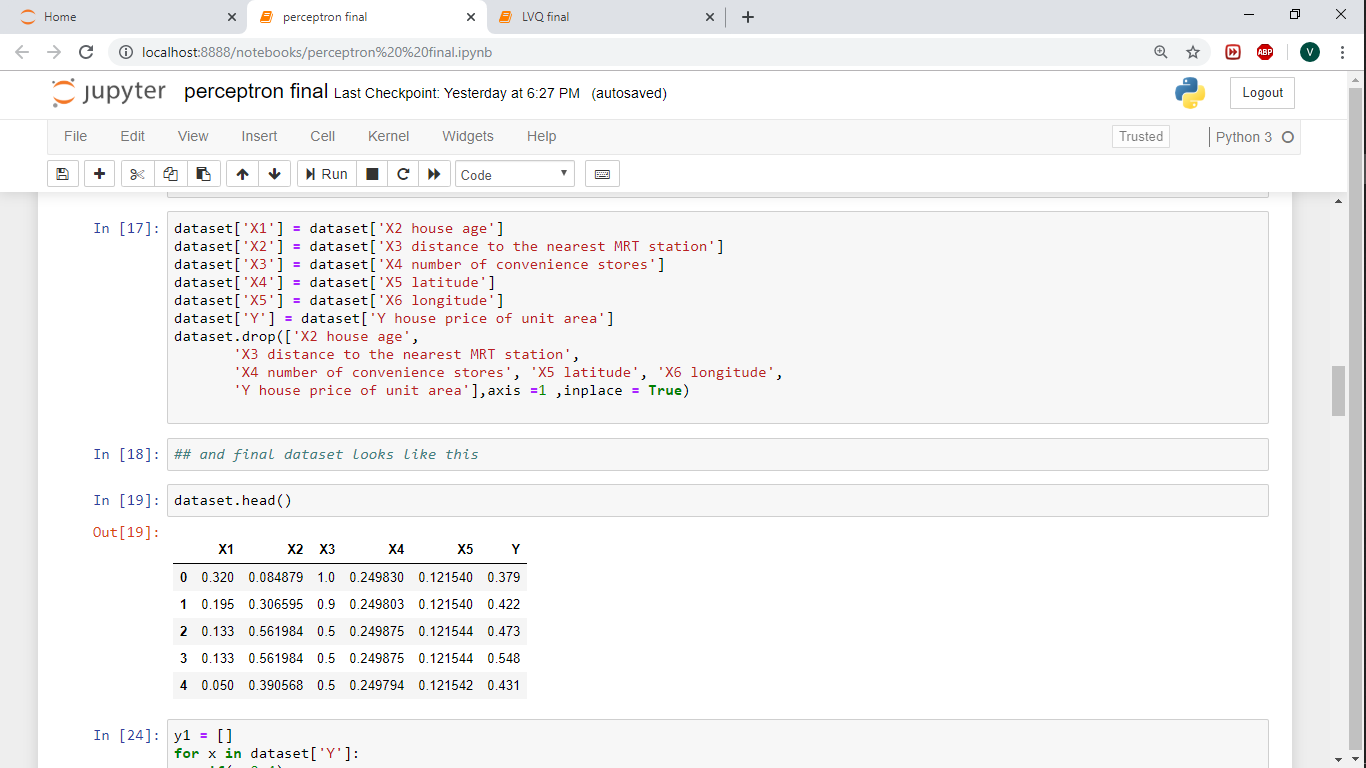
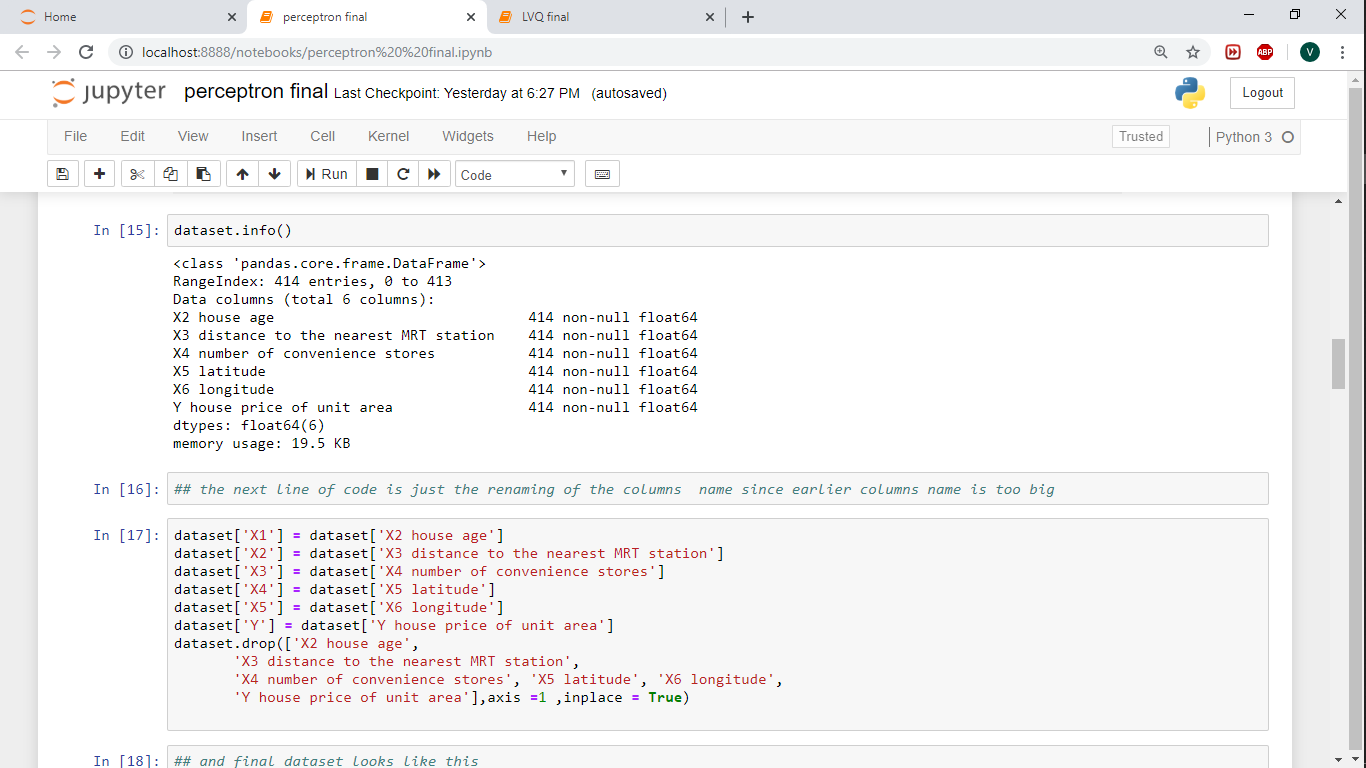
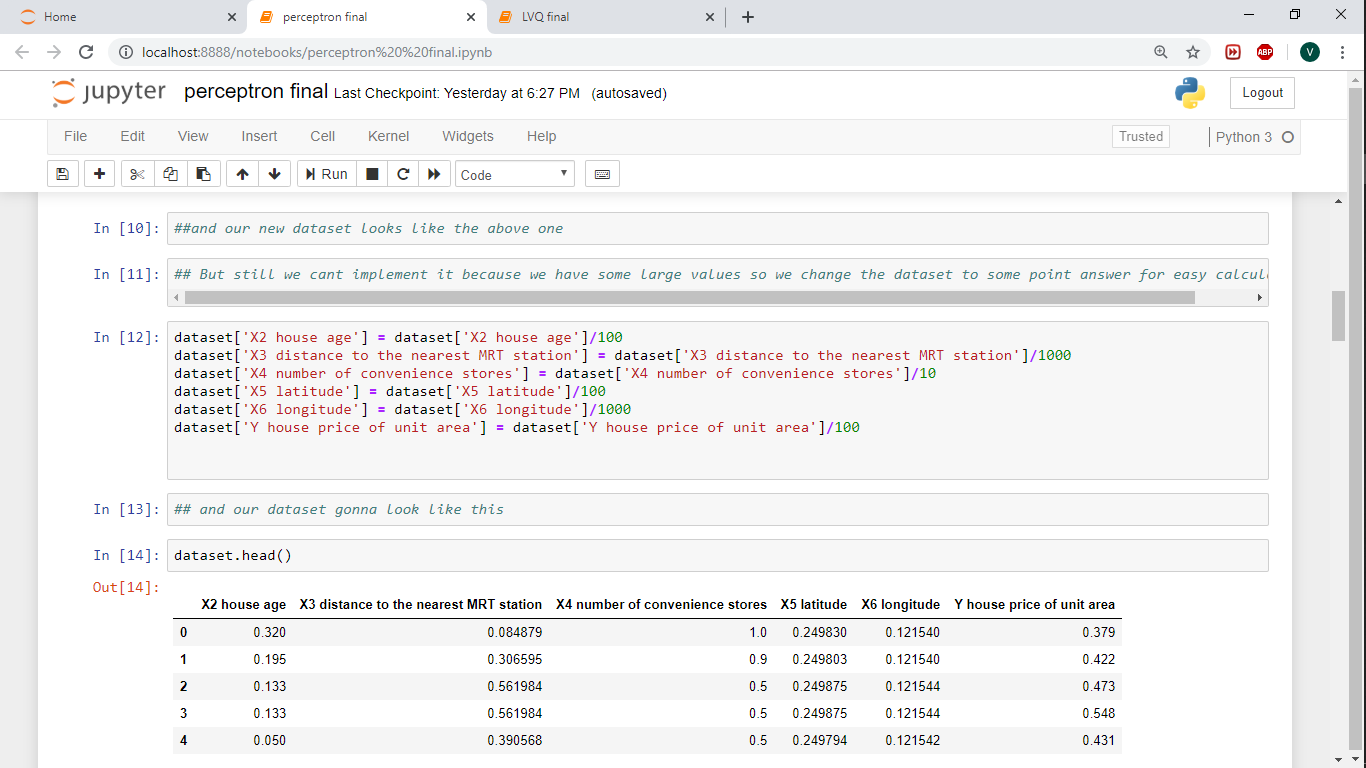
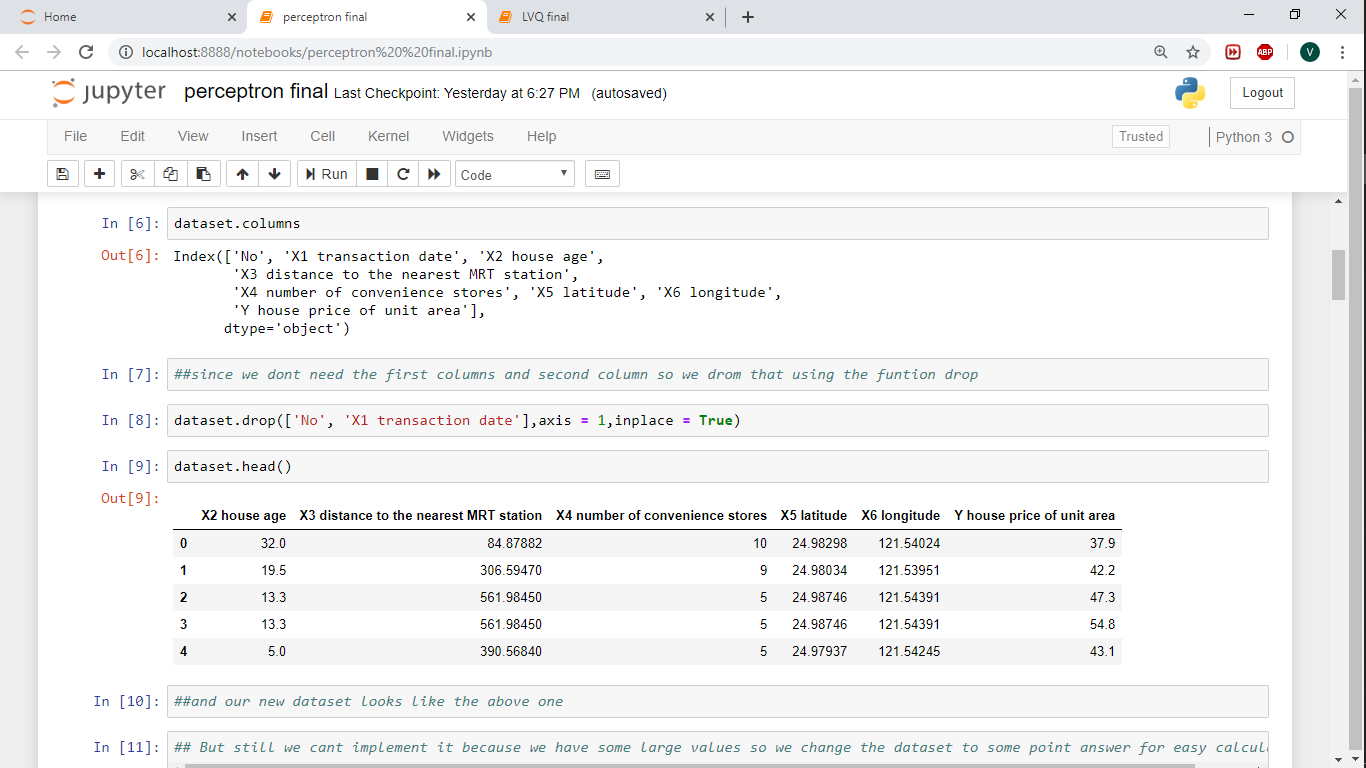
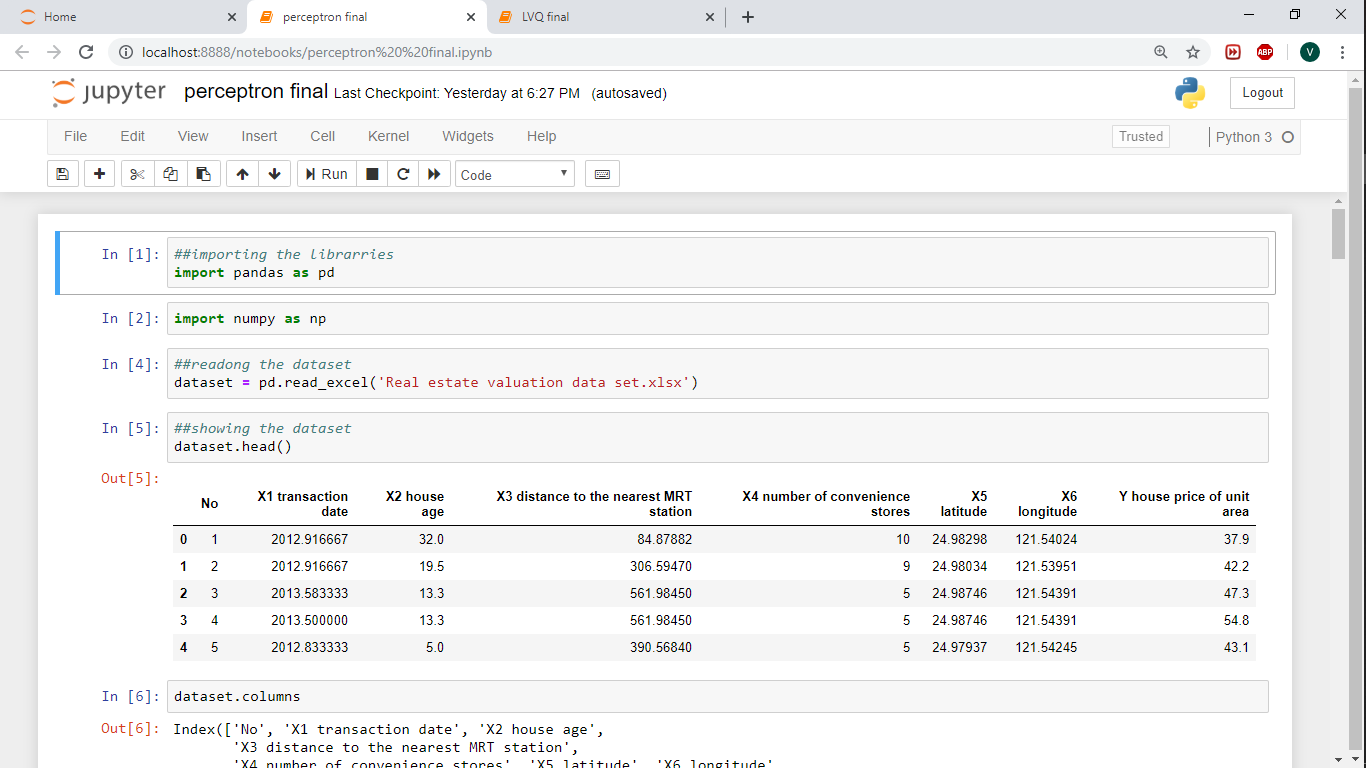
Overlapping classes: data points on the “wrong” side of the discriminant margin are weighted down to reduce their influence (“soft margin”);

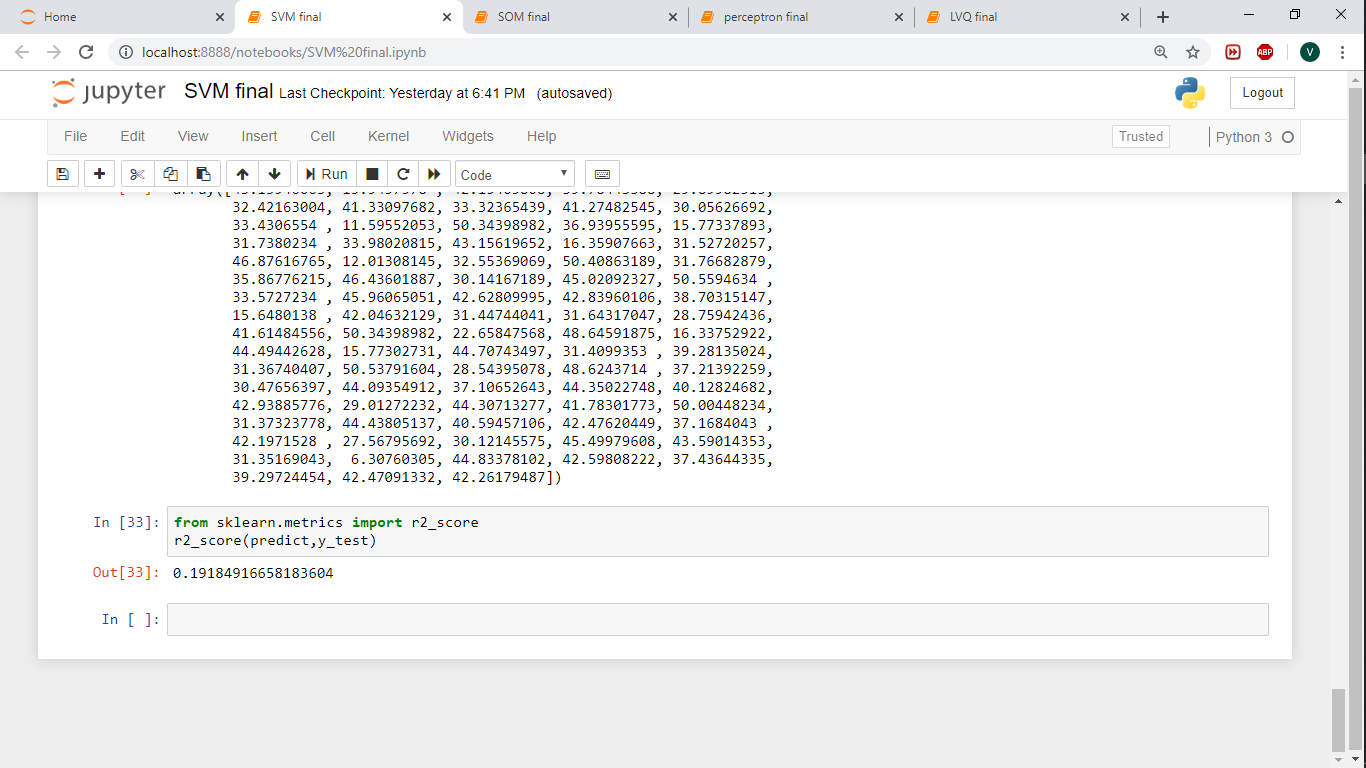
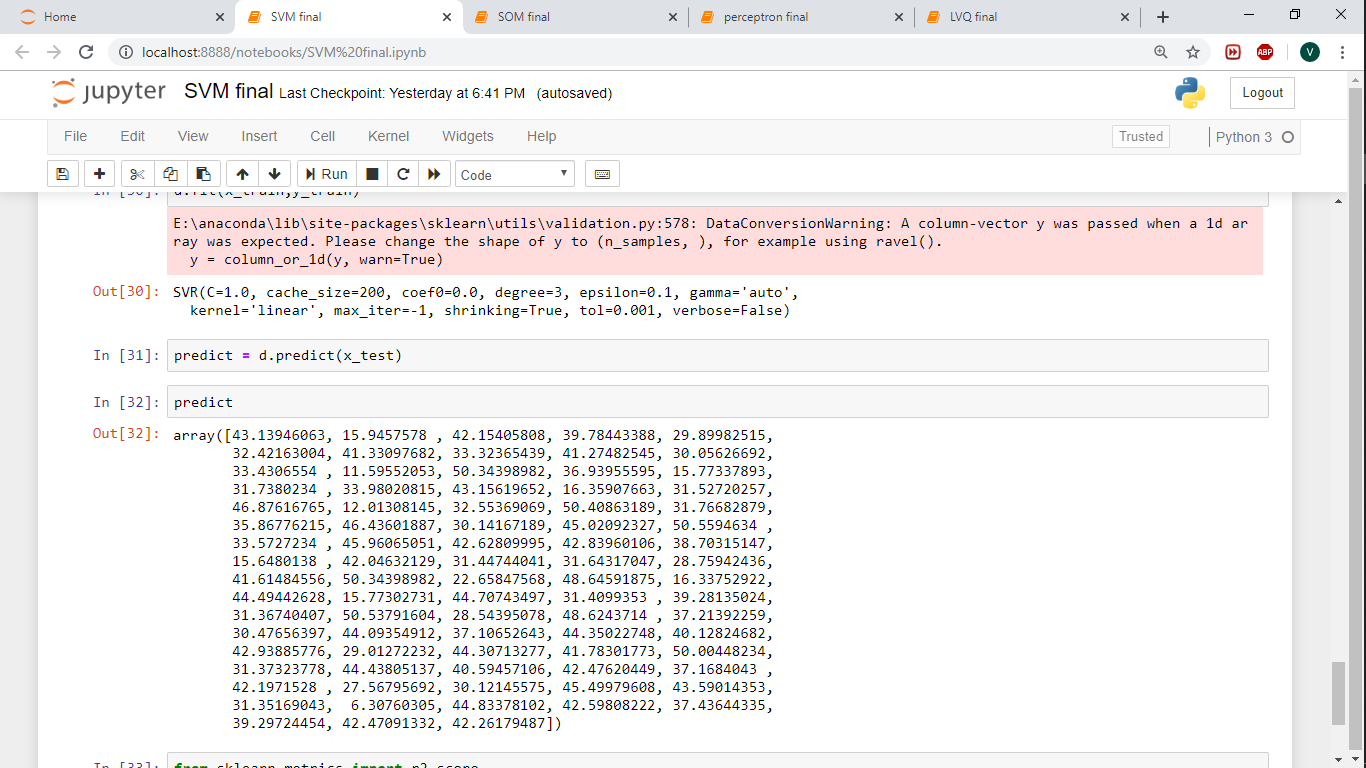
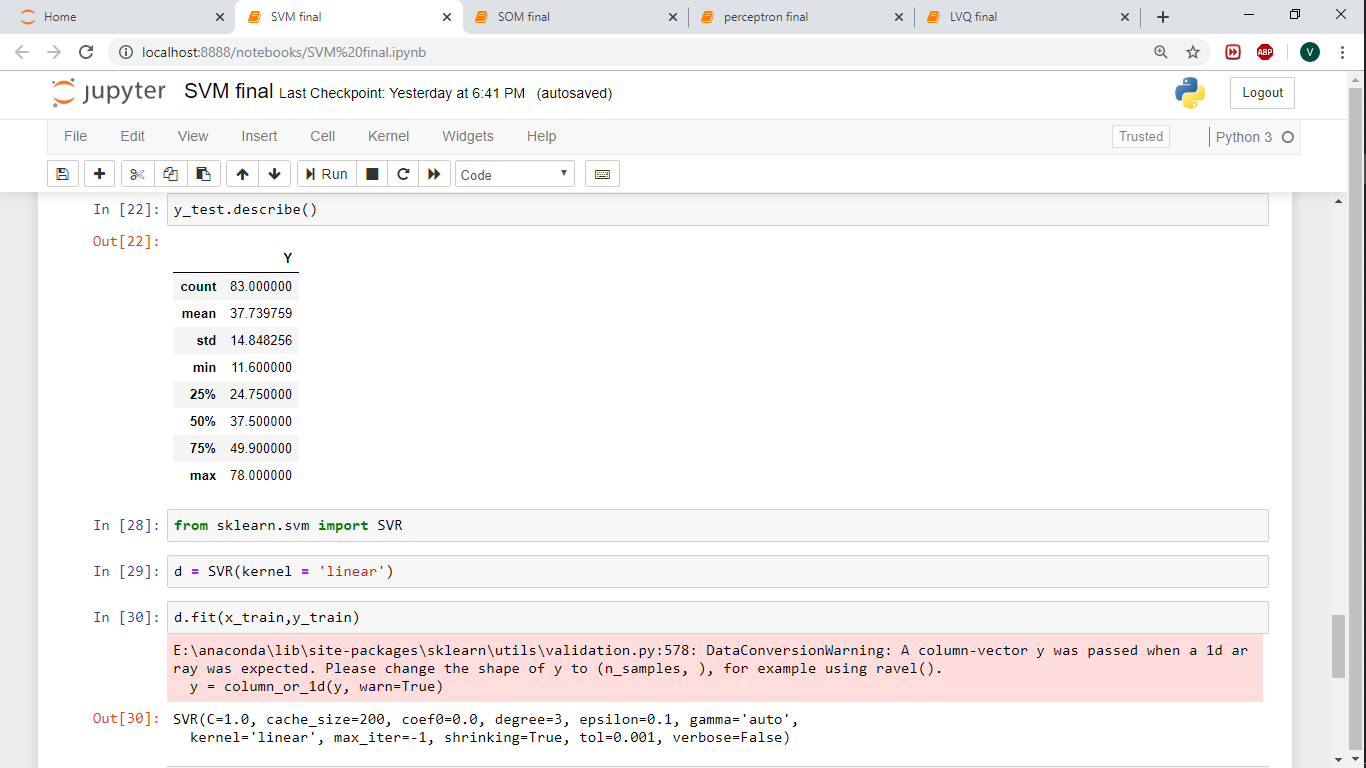
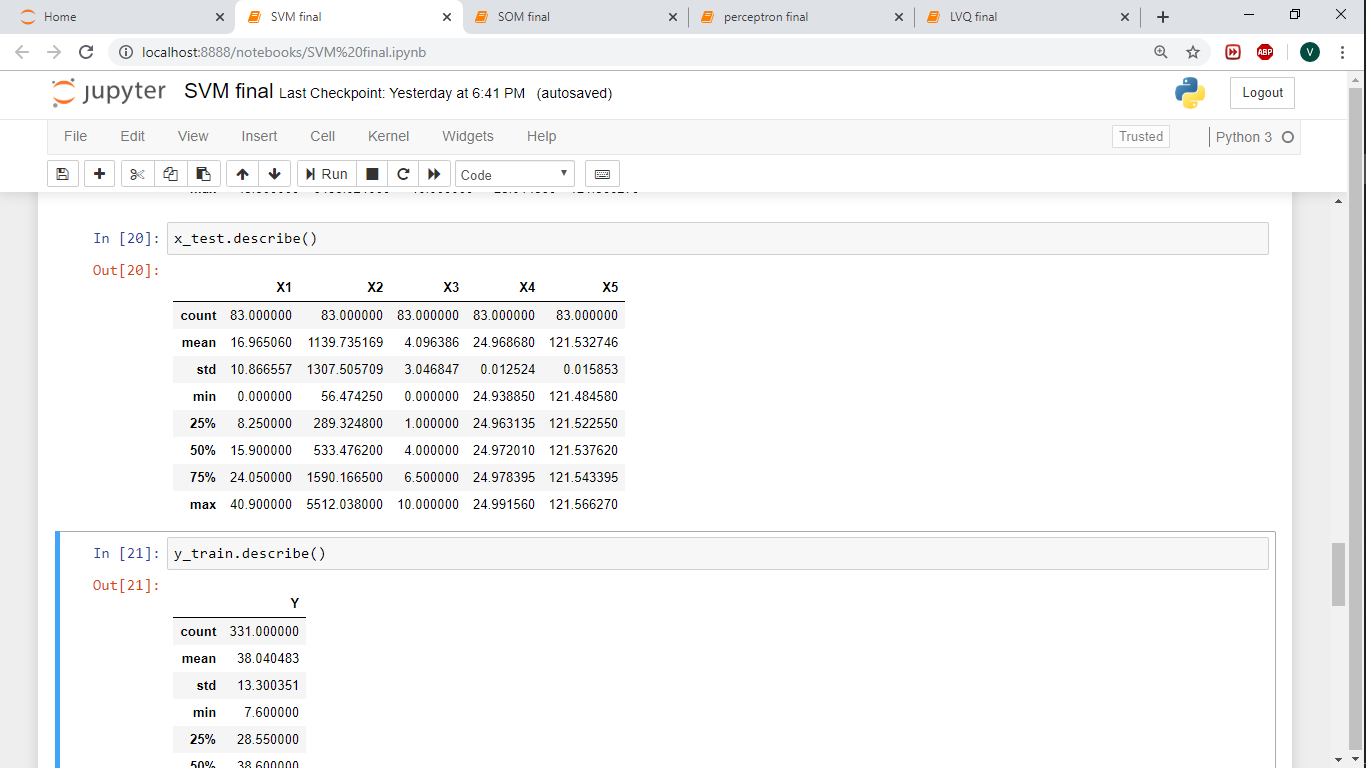
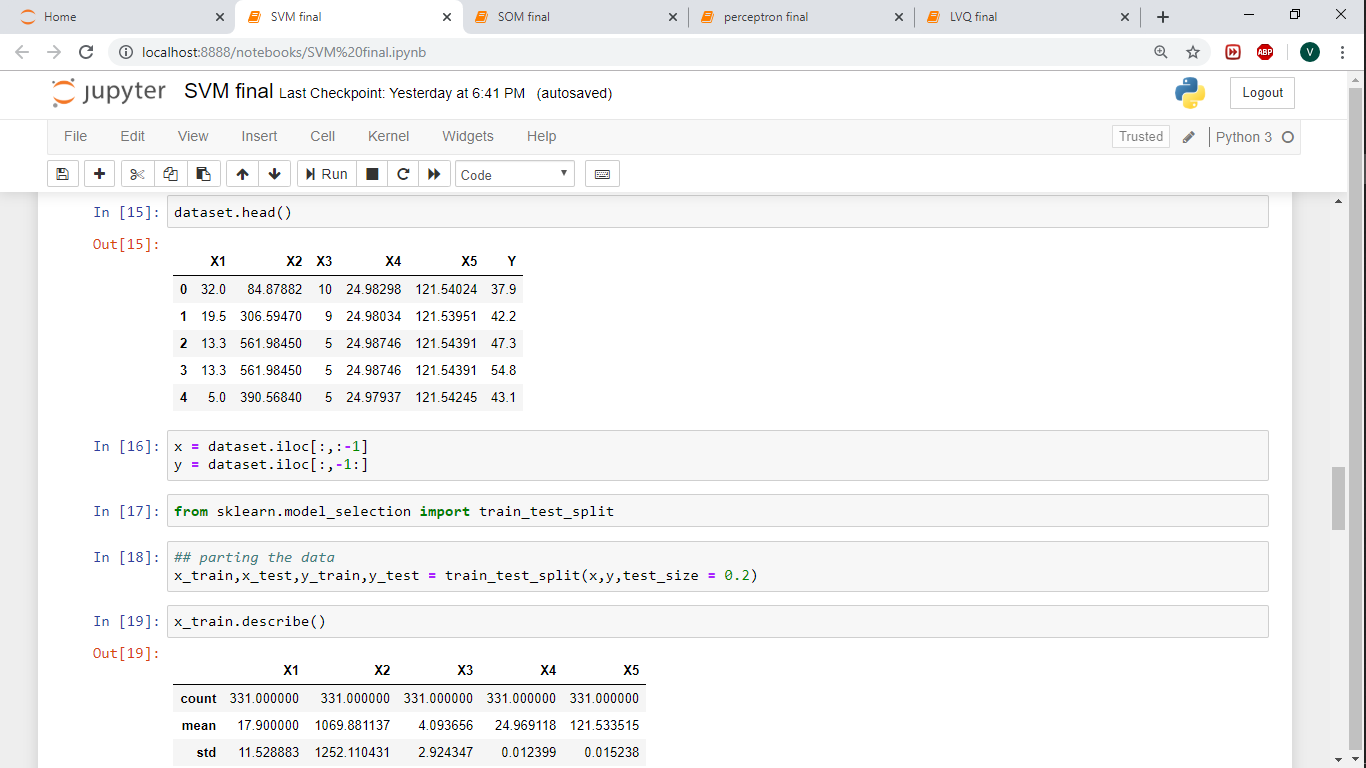
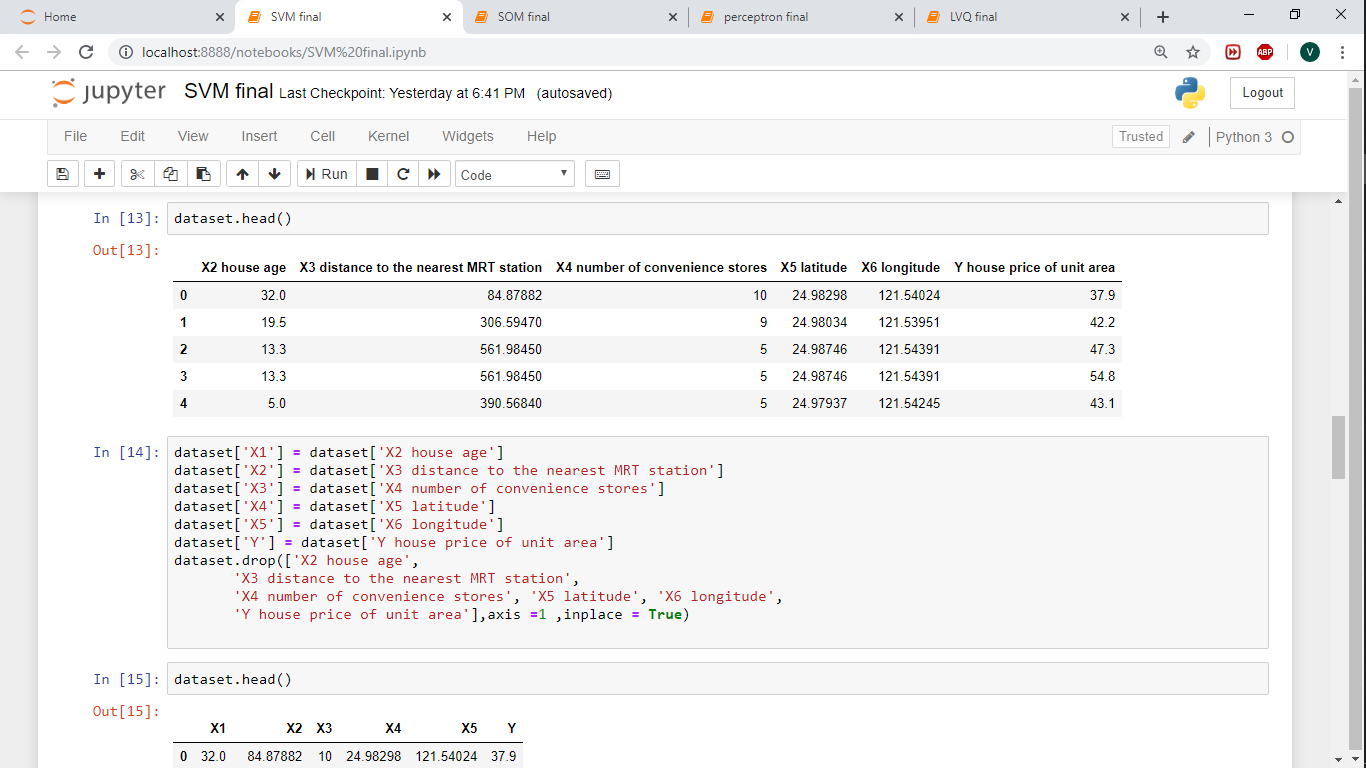
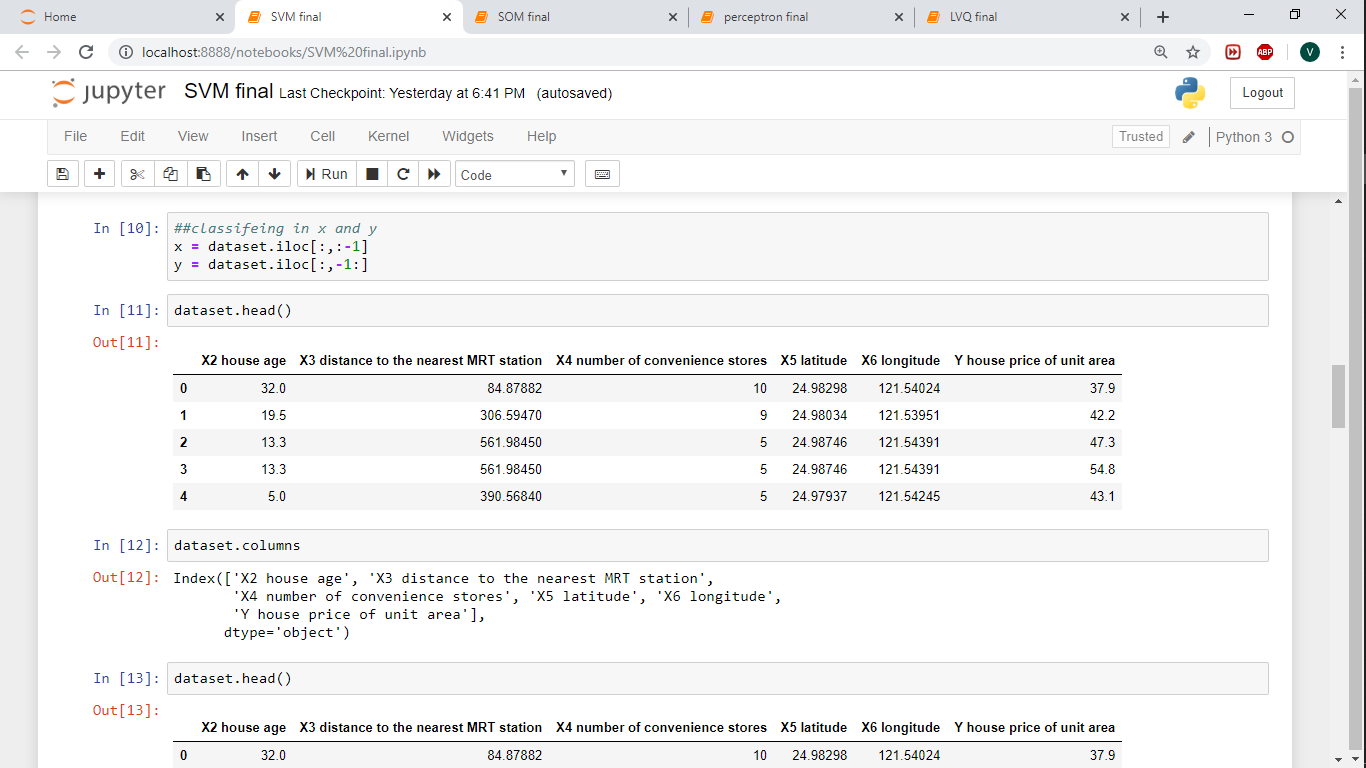
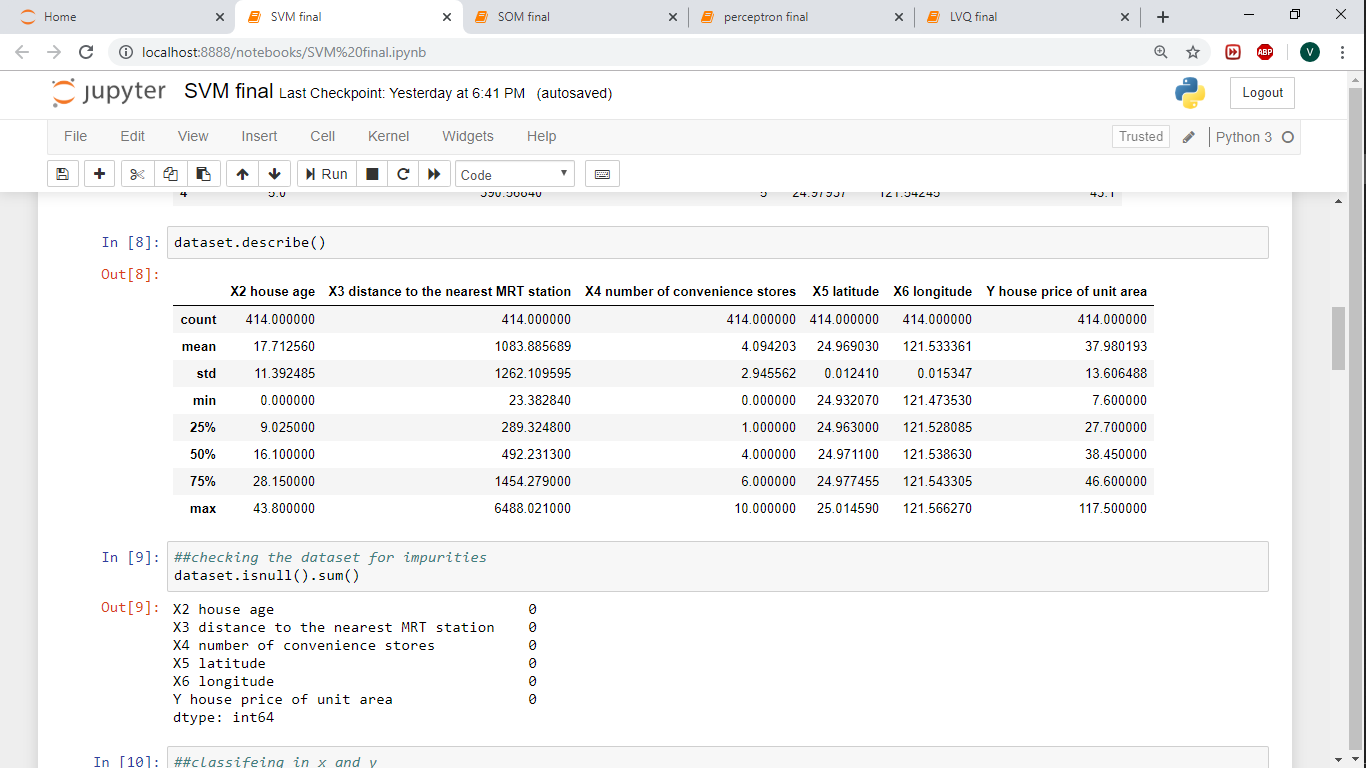
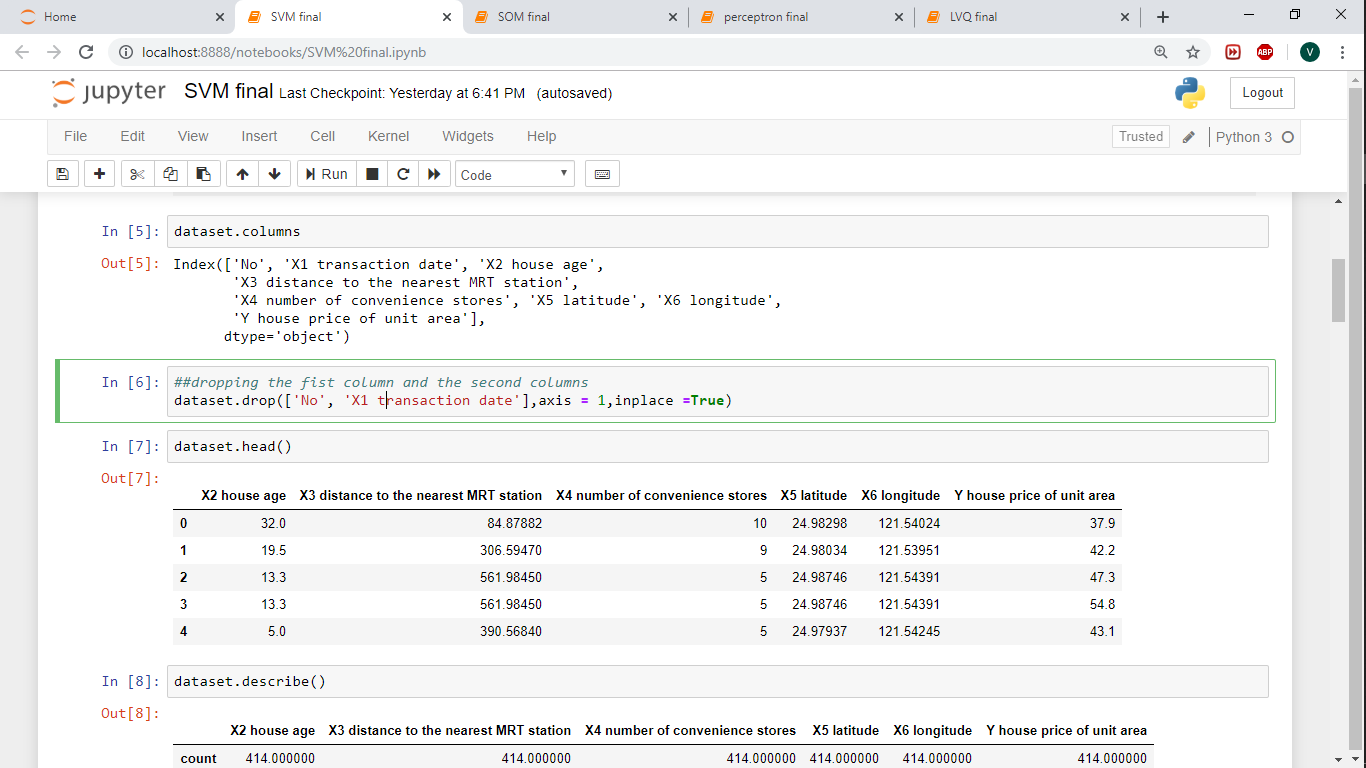
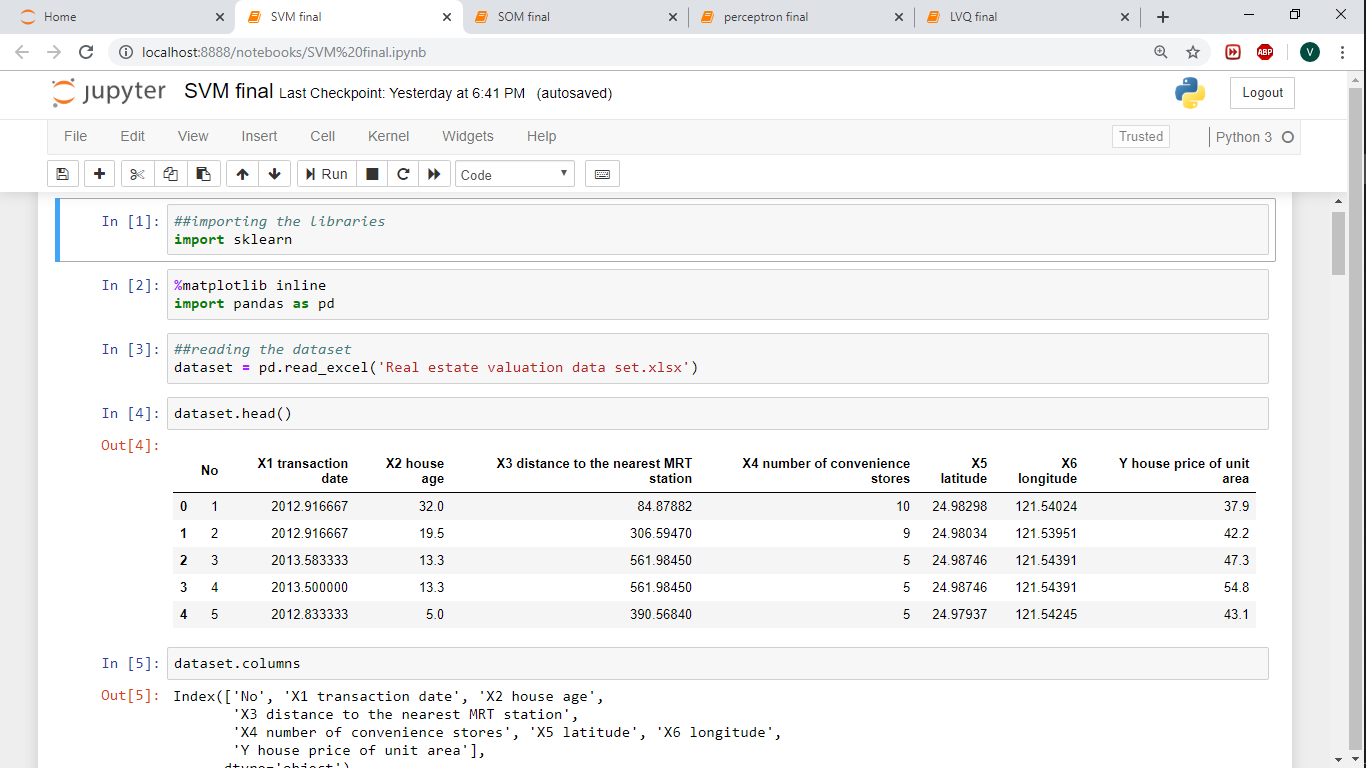
Nonlinearity: when we cannot find a linear separator, data points are projected into an (usually) higher-dimensional space where the data points effectively become linearly separable (this projection is realised via kernel techniques);

Problem solution: the whole task can be formulated as a quadratic optimization problem which can be solved by known techniques. A program able to perform all these tasks is called a Support Vector Machine. { Margin Support Vectors Separating Hyperplane Figure 1: Classification (linear separable case) Several extensions have been developed; the ones currently included in libsvm are: ν-classification: this model allows for more control over the number of support vectors (see Sch¨olkopf et al., 2000) by specifying an additional parameter ν which approximates the fraction of support vectors; One-class-classification: this model tries to find the support of a distribution and thus allows for outlier/novelty detection; Multi-class classification: basically, SVMs can only solve binary classification problems. To allow for multi-class classification, libsvm uses the one-against-one technique by fitting all binary subclassifiers and finding the correct class by a voting mechanism; -regression: here, the data points lie in between the two borders of the margin which is maximized under suitable conditions to avoid outlier inclusion; 2 ν-regression: with analogue modifications of the regression model as in the classification case.

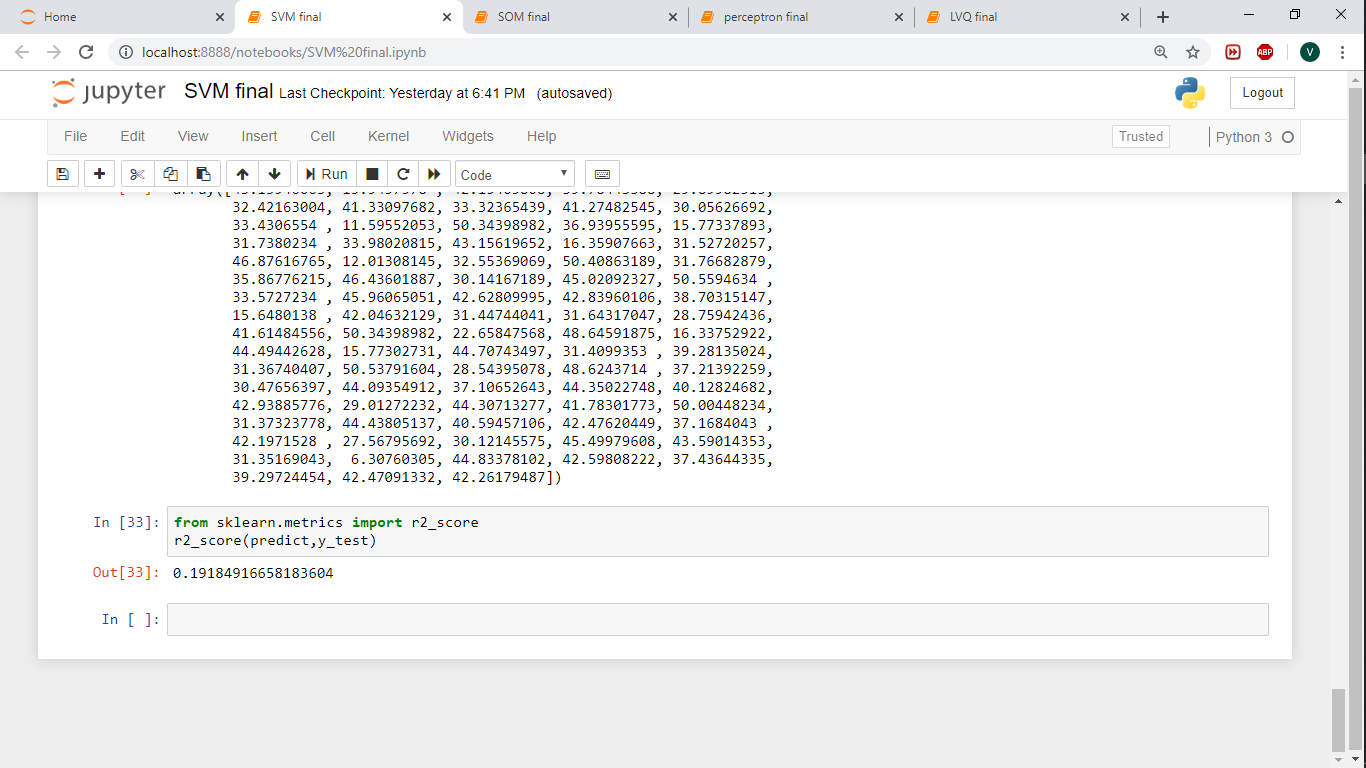
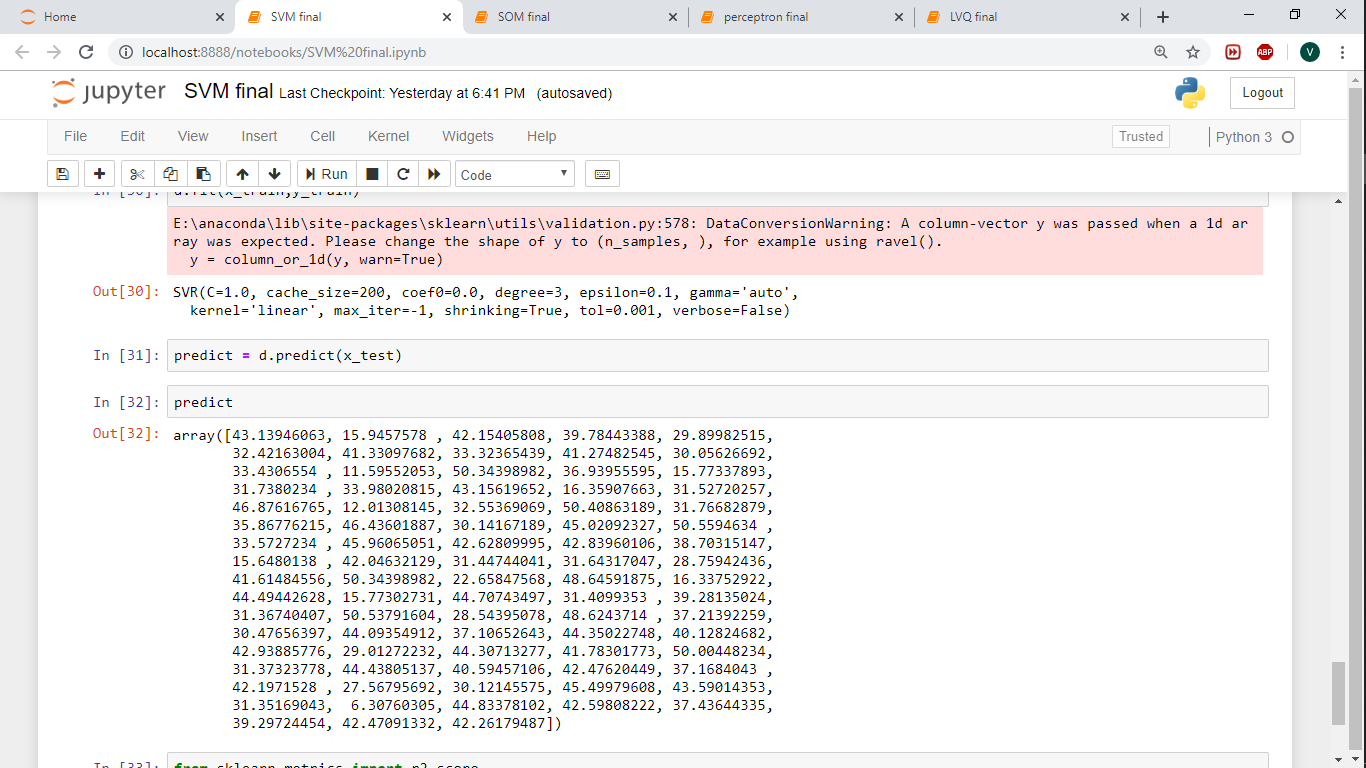
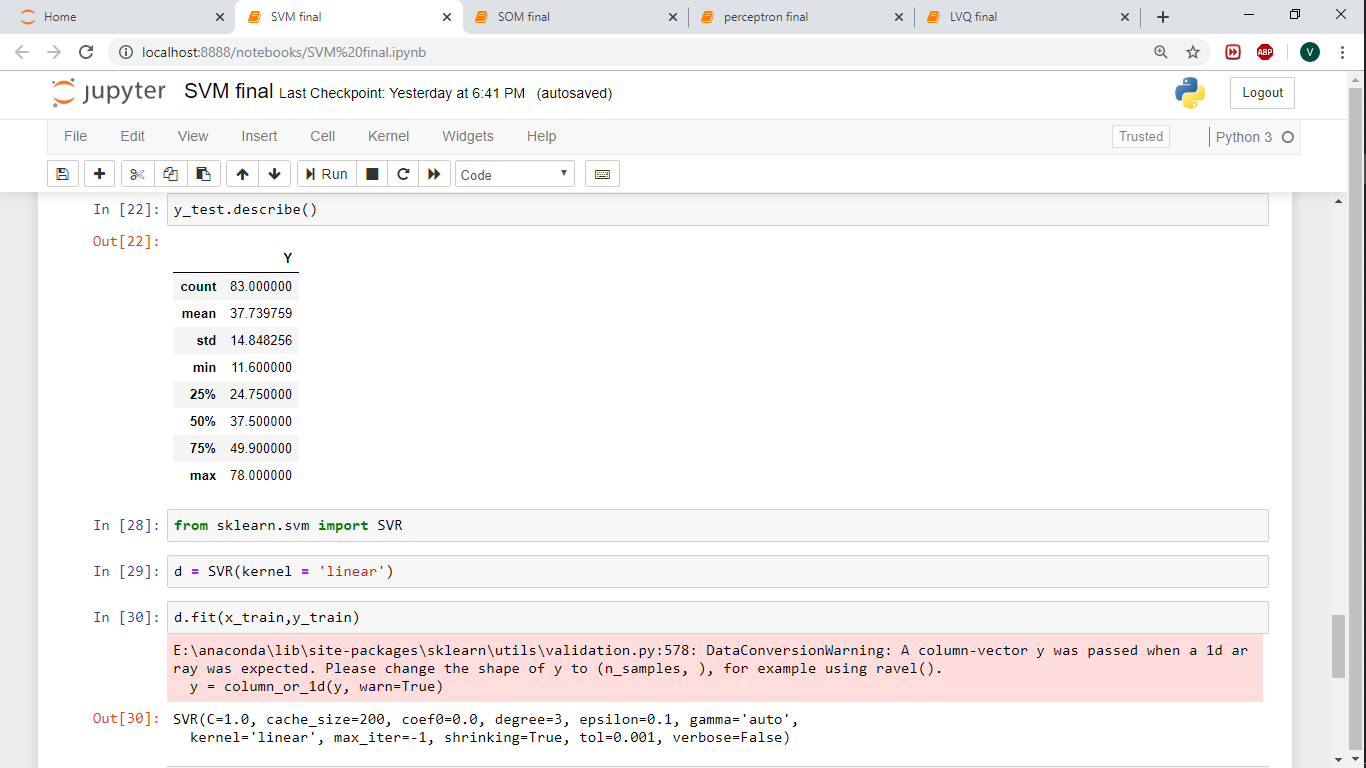
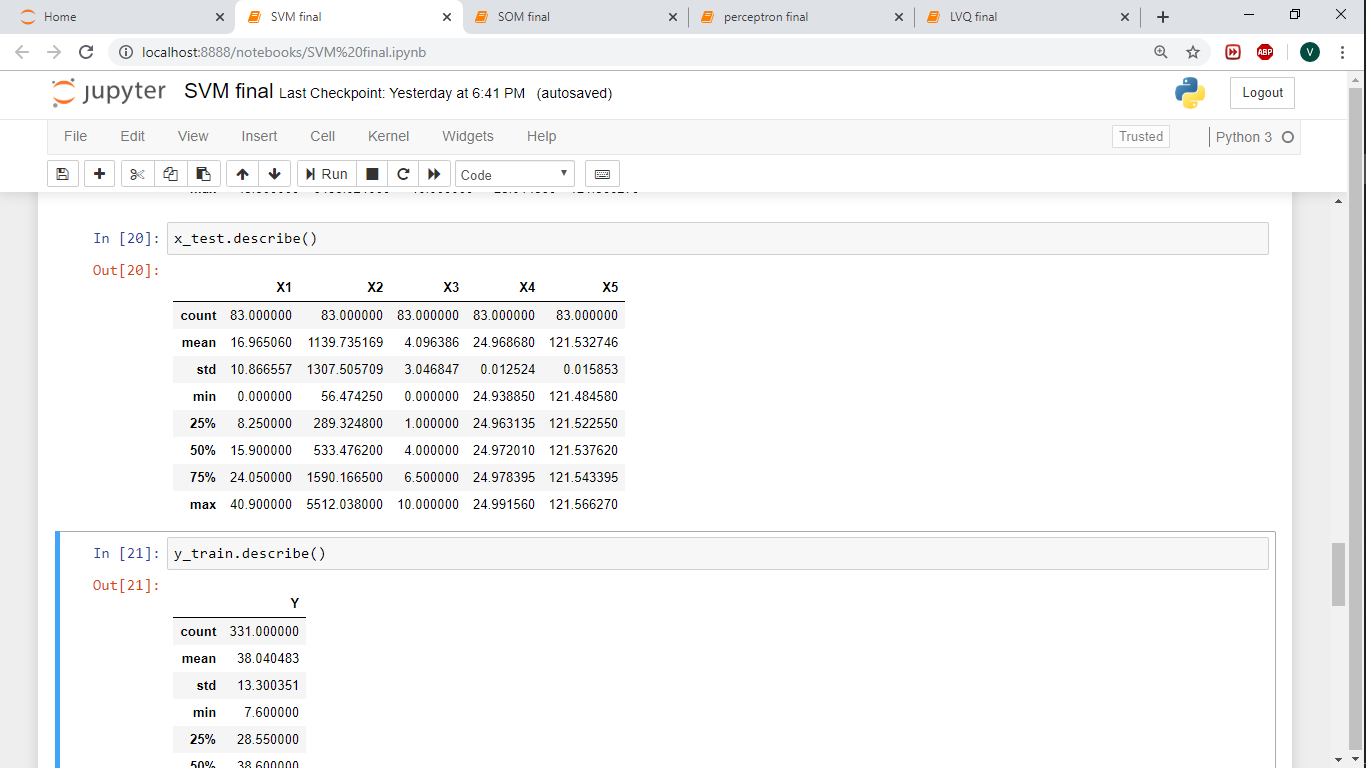
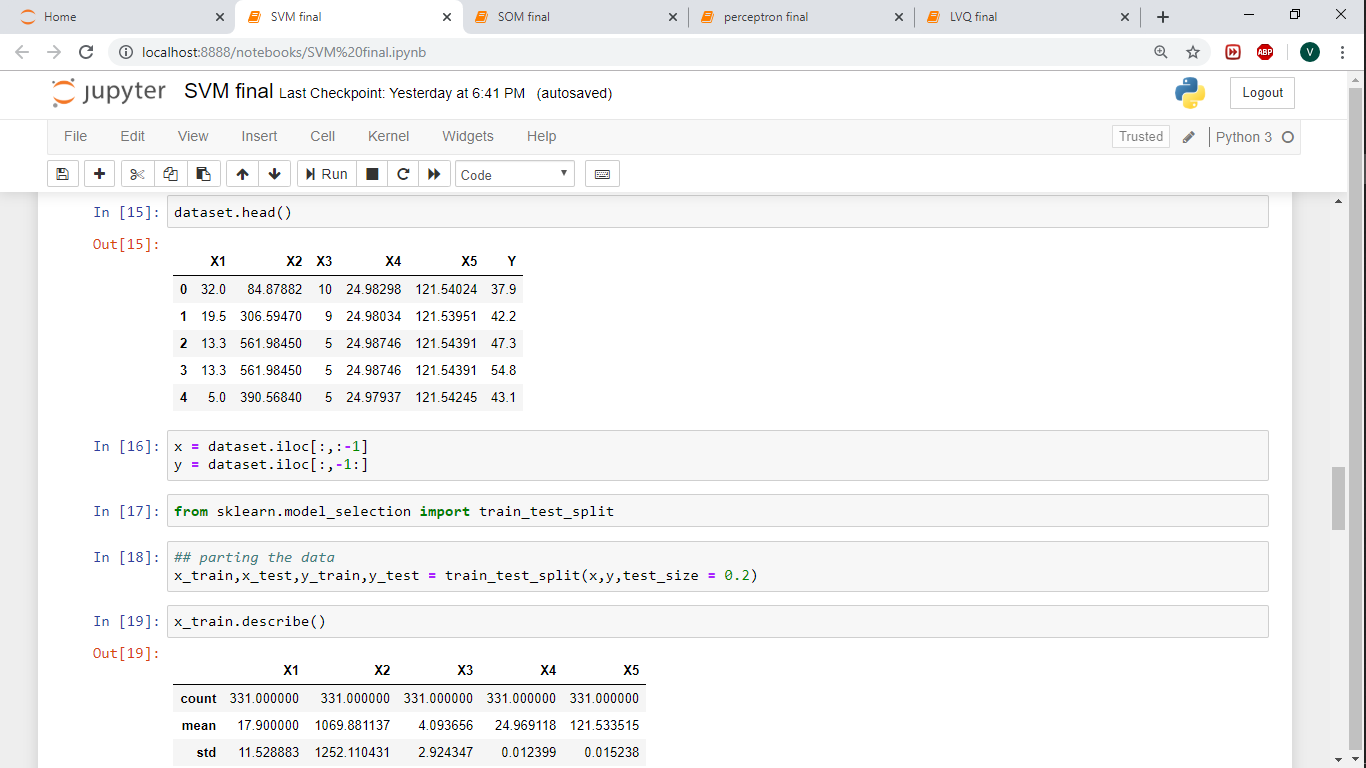
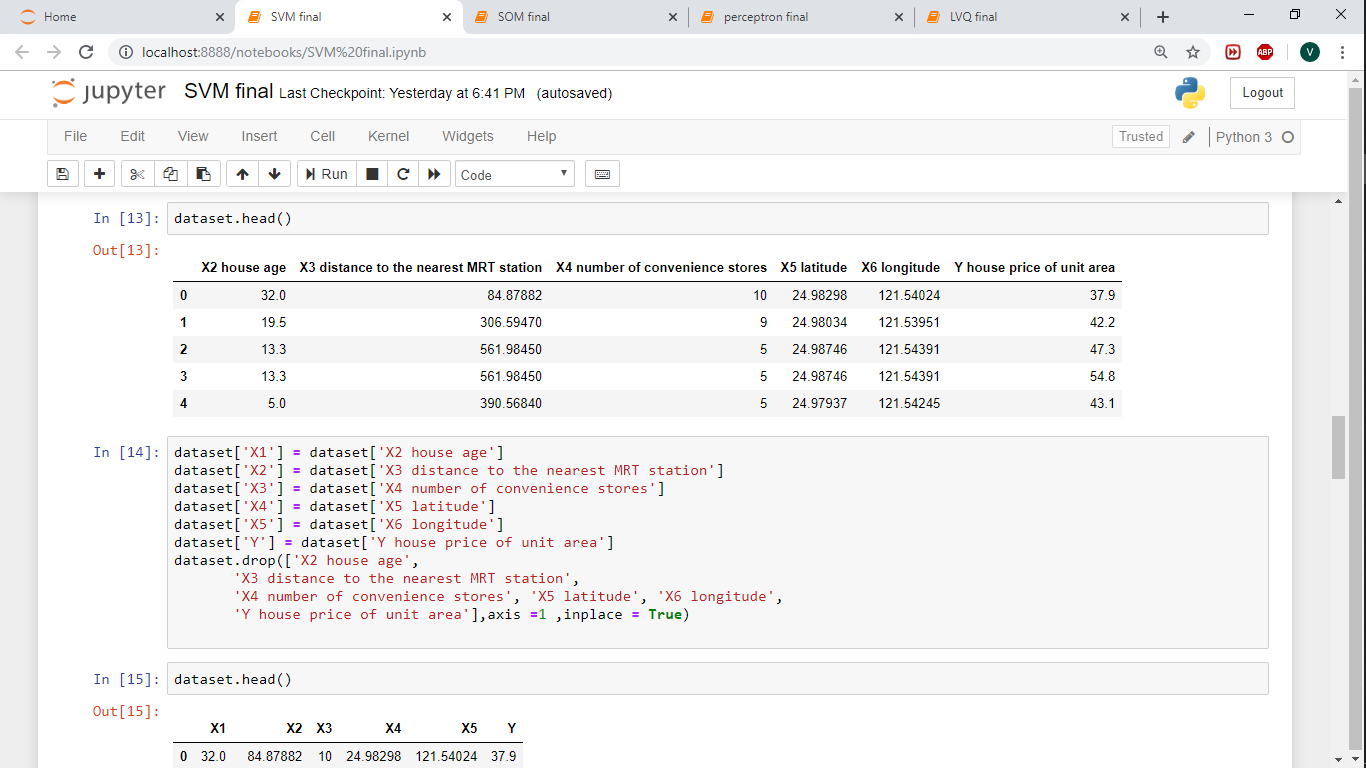
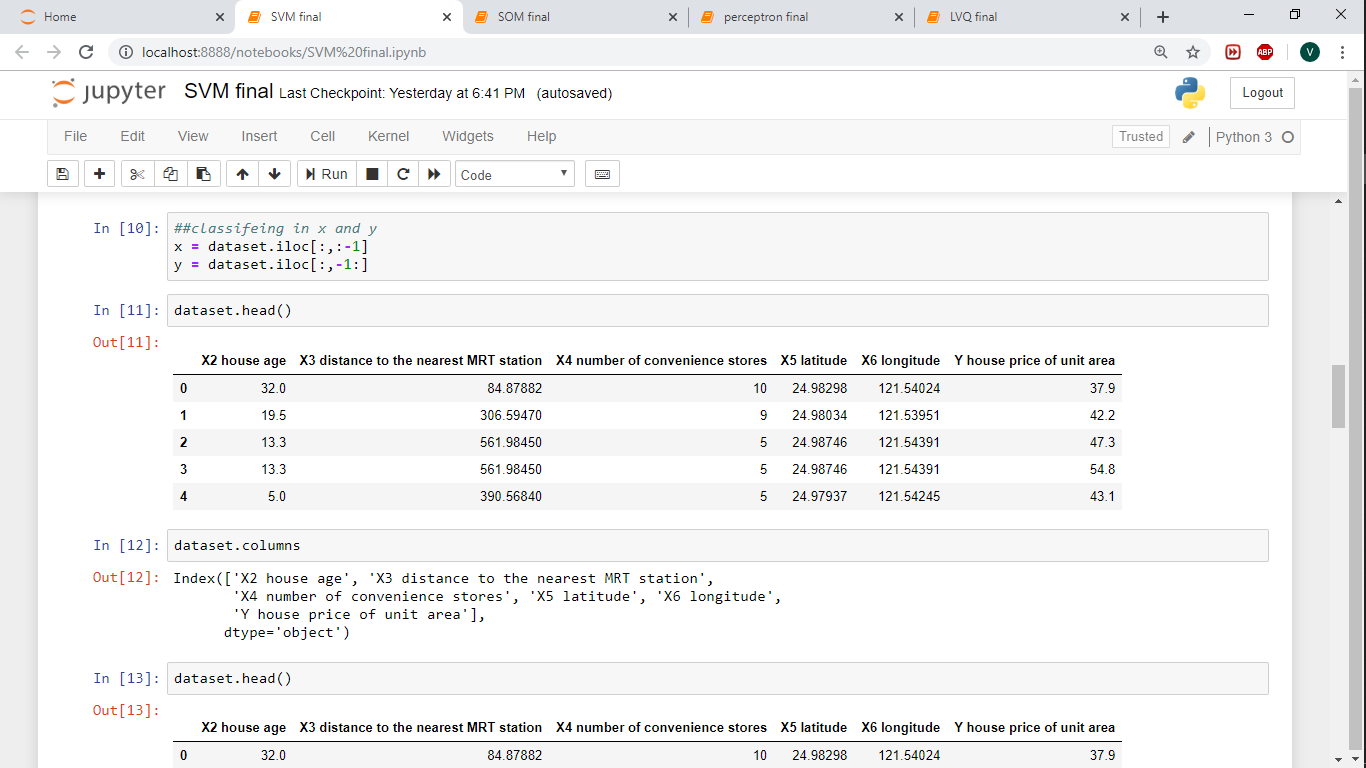
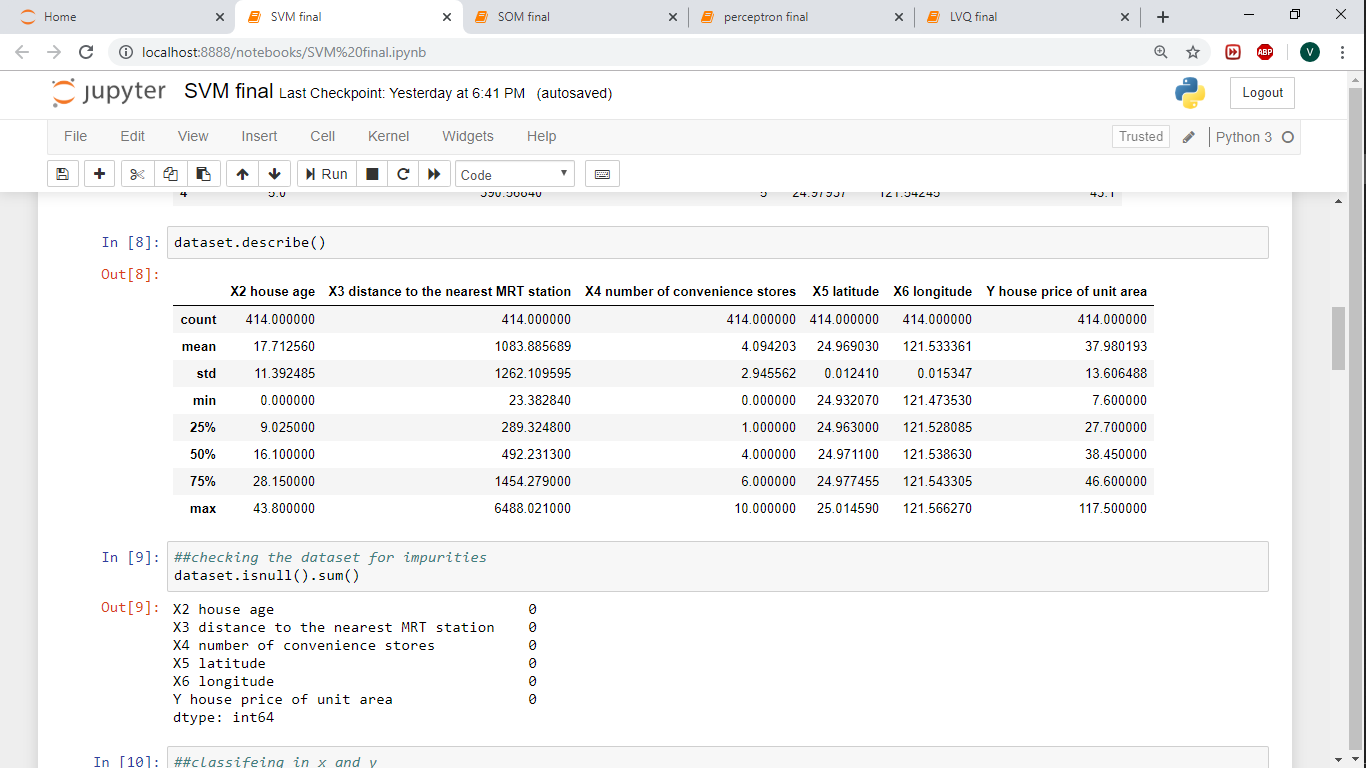
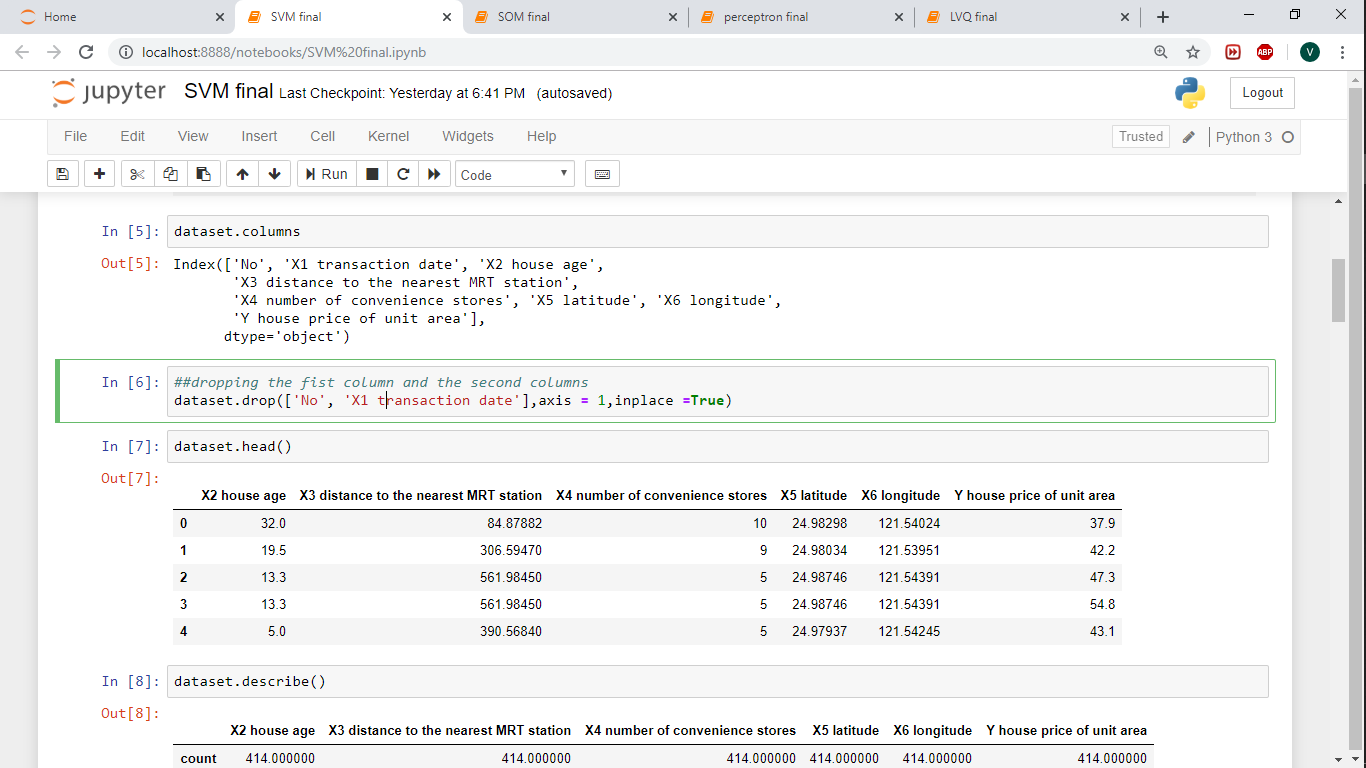
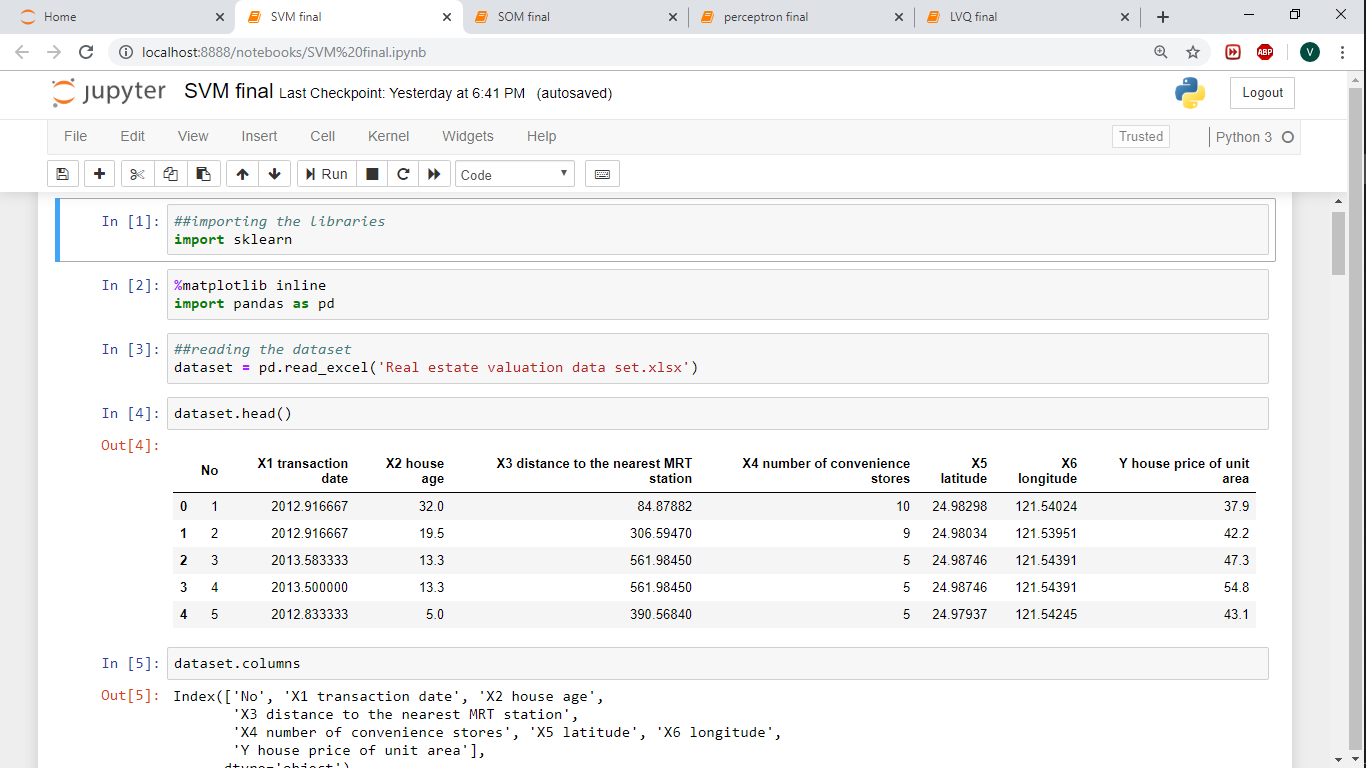


1. **Perceptron Model**

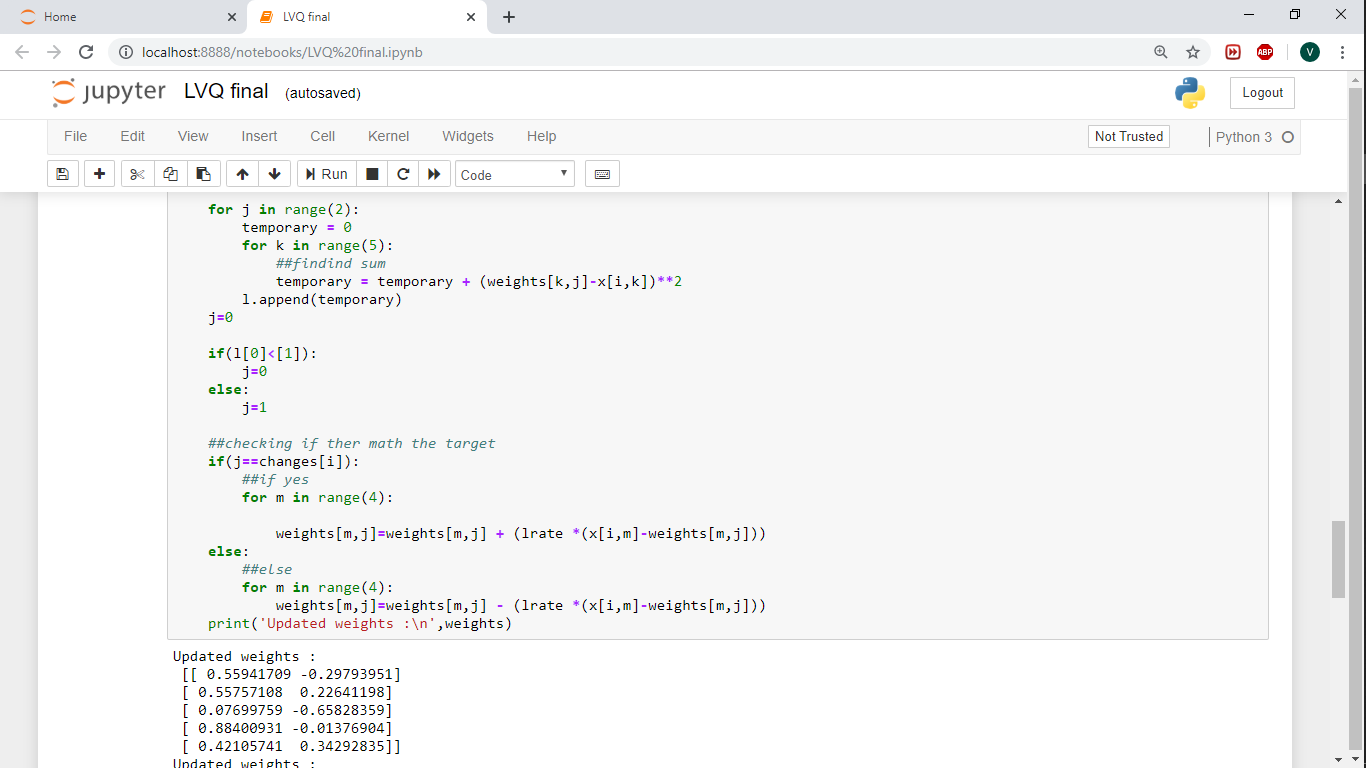
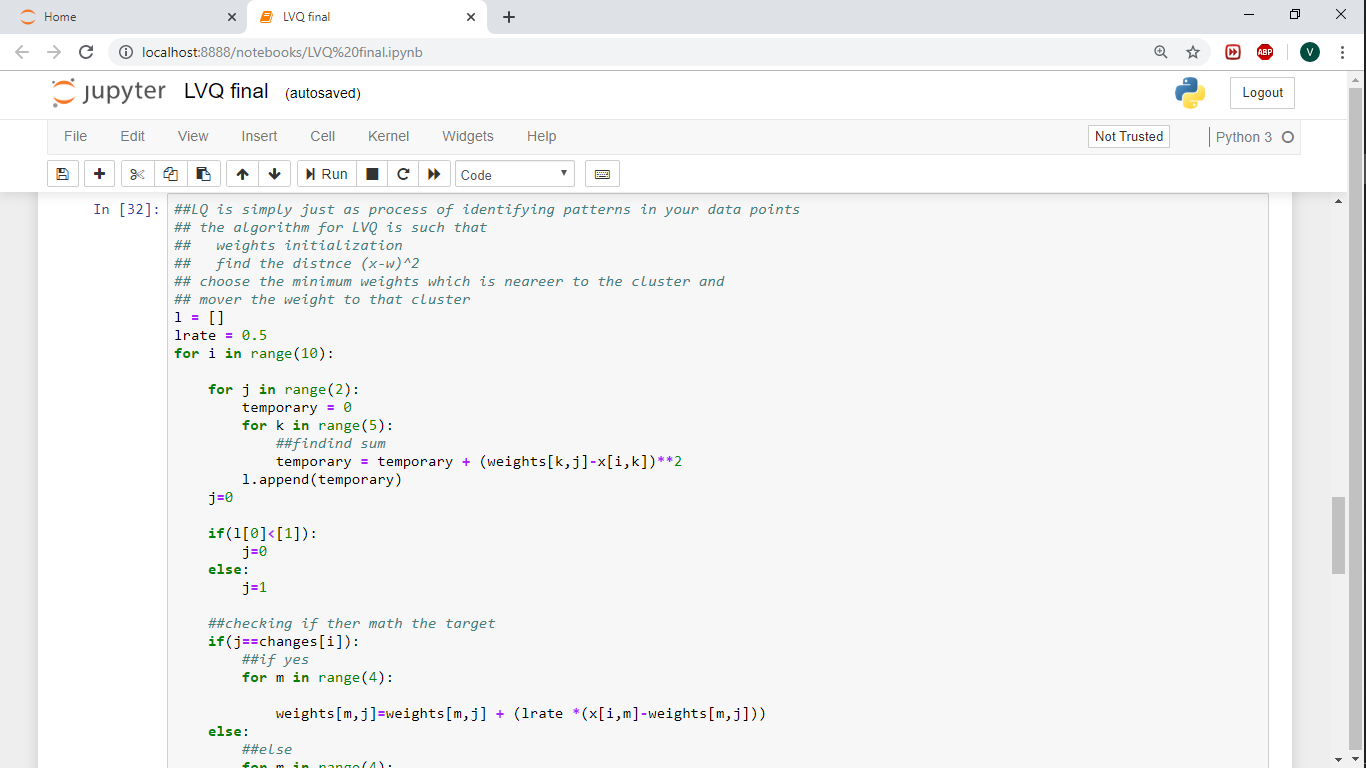
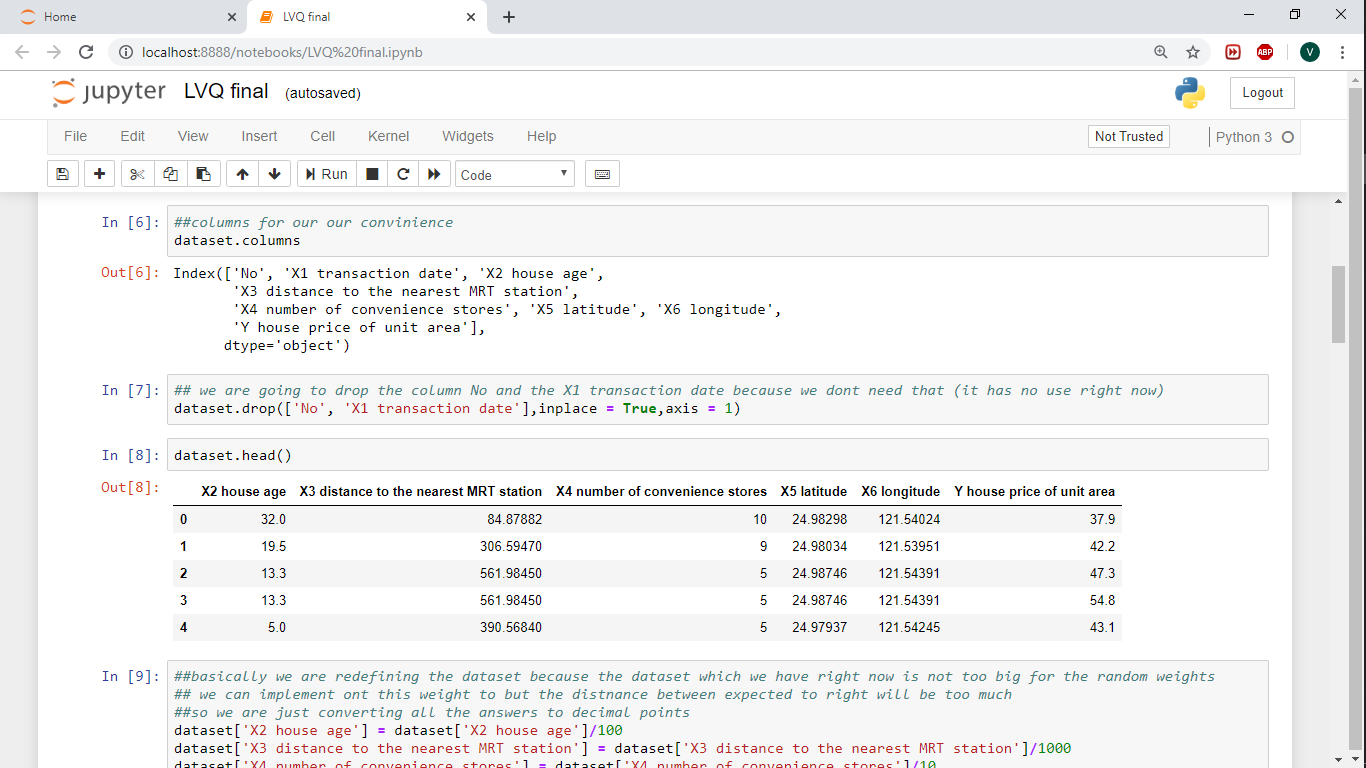
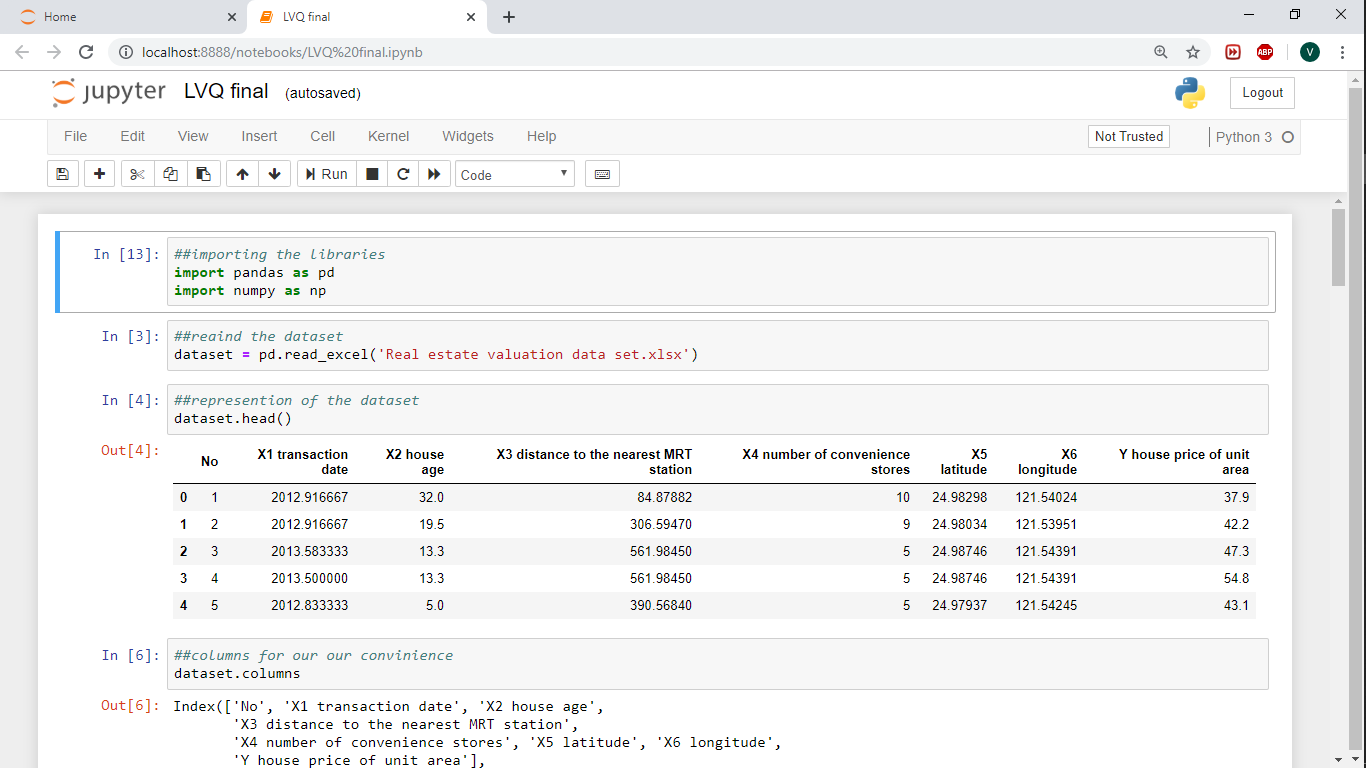
****

1. **SOM**

**3) SVM**

****

**4) LVQ**

****