INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, ALLAHABAD
VI Semester B.Tech in Information Technology
Report - Group Assignment 3

Data Mining and Warehousing

Generalization ability of SVM classification based on Markov Sampling

By :-

Palak Mittal (IIT2018117)

Nehal Singh (IIT2018119)

Abhishek Kumar Gupta (IIT2018187)

Puja Kumari (IIT2018191)

Prabha Kumari (IIT2018195)

Why do we do Markov Sampling?

There are many methods of sampling which are dependent (e.g., α mixing, β mixing and φ mixing) learned in machine textbooks. In this article, we focus only on the analysis where input samples are Markov's chains, the reasons are as follows. First, in real-world problems, Markov chain samples appear frequently and naturally in applications, such as biological analysis (DNA or protein), web-based content search, tag prediction, and so on. Second, ample strong evidence suggests that learning algorithms often work well with Markov chain samples (e.g., biological sequencing analysis, speech recognition). The reason for this, however, was unknown (in particular, it is not known how well it works in terms of general learning and practice). Or we can say that it is a normal approach.

Markov Chains

The Markov chain is a kind of **stochastic process**. This situation means that the future depends on the present, but not on the past. Therefore, the Markov series roams through the realm of the state, remembering only when they were just in the final stage. The accumulation of transformation opportunities is sometimes called the **transformation matrix** when working with discrete provinces, or more generally, the **kernel**.

Why MCMC Works: Reversible Markov Chains

The Markov chain Monte Carlo mimics the Markov chain where a particular activity of interest (e.g. the co-distribution of parameters of a particular model) is a unique, random division.

The flexibility is guaranteed with any Markov series being returned. Think of Markov's chains in a repetitive sequence: $\{\theta^{(n)}, \theta^{(n-1)}, ..., \theta^{(0)}\}$. This sequence is still Markovian, because:

$$Pr(\theta^{(k)} = y \mid \theta^{(k+1)} = x, \theta^{(k+2)} = x_1, ...) = Pr(\theta^{(k)} = y \mid \theta^{(k+1)}) = x$$

Forward and reverse transition probabilities can be related to Bayes theorem:

$$Pr(\theta^{(k)} = y \mid \theta^{(k+1)} = x) = \frac{Pr(\theta^{(k+1)} = x \mid \theta^{(k)} = y)\pi^{(k)}(y)}{\pi^{(k+1)}(x)}$$

Though not homogeneous in general, π becomes homogeneous if:

- $n \to \infty$
- $\pi^{(i)} = \pi$ for some $i<k$

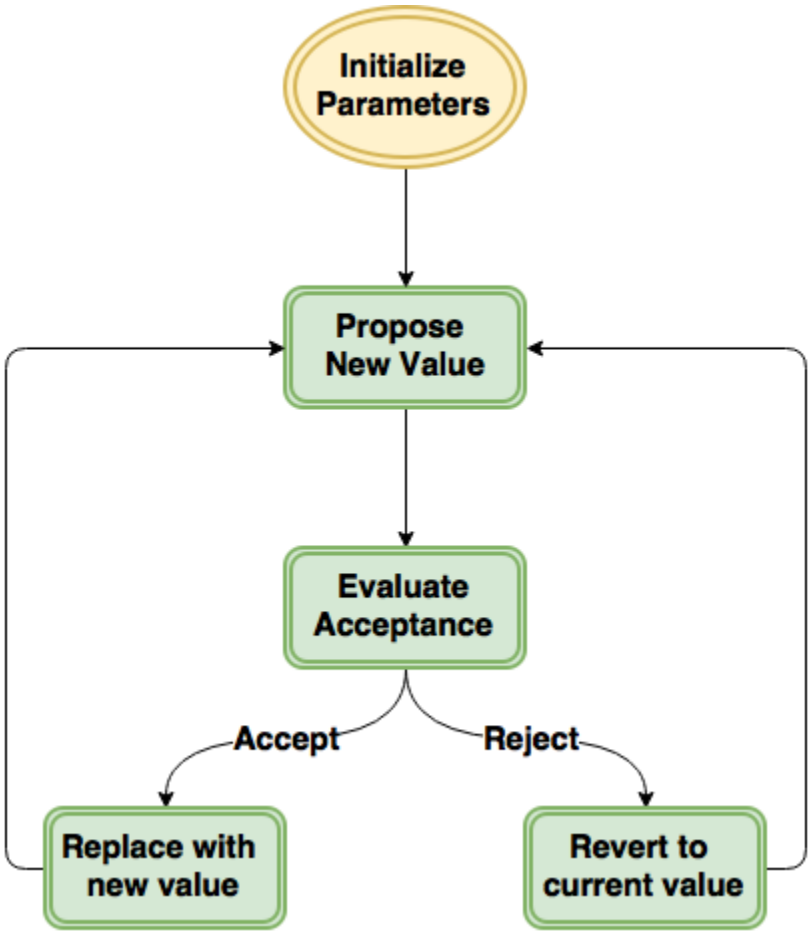
If chain is homogeneous it is called reversible, Because it satisfies

$$\pi(x)Pr(y \mid x) = \pi(y)Pr(x \mid y)$$

reversibility is crucial and useful because it has the effect of balancing movement throughout the state space. When Markov's series comes back, a different, consistent, consistent distribution of that series. Therefore, if there is interest, we need to find only Markov's recoverable chain which is a limited distribution. This is done by MCMC!

The Metropolis-Hastings Algorithm

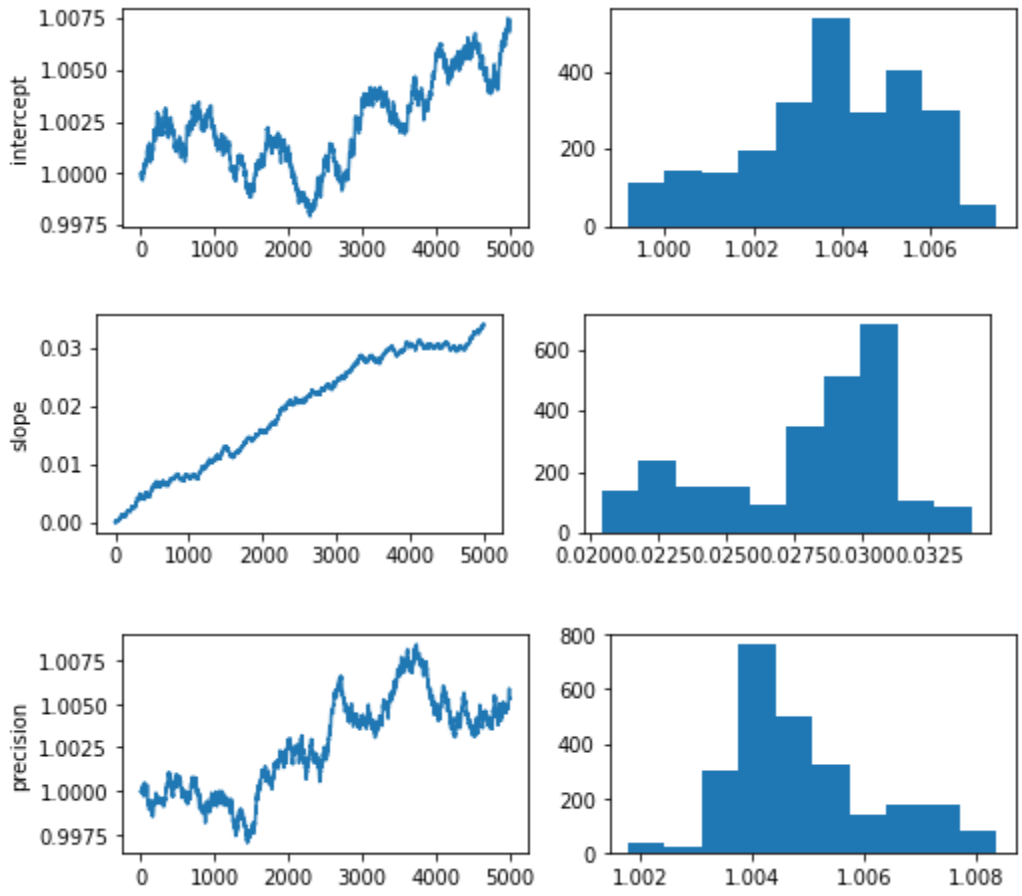
One of the simplest and most flexible algorithms making Markov's flexible chains is the **Metropolis-Hastings** algorithm. Since we cannot take a direct sample in the background (unknown) distribution, this algorithm uses auxiliary distribution that is easy to sample. These samples produce a change in the state of candidates, which is accepted or rejected as much as possible.



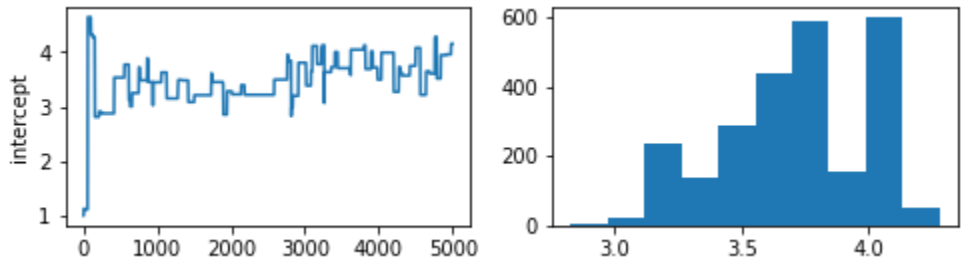
On Letter Dataset

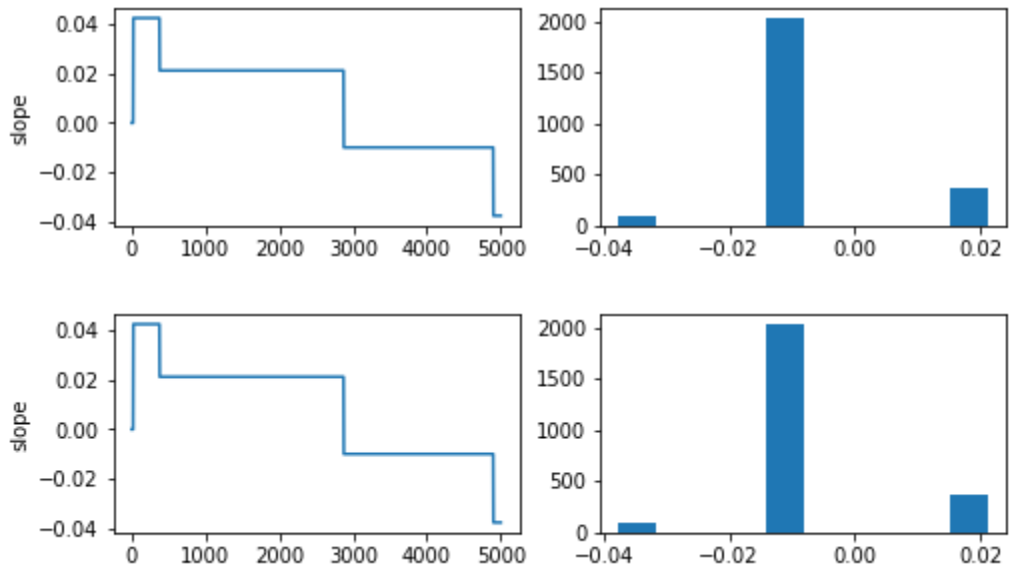
Observation:

- Result of markov sampling with low variance

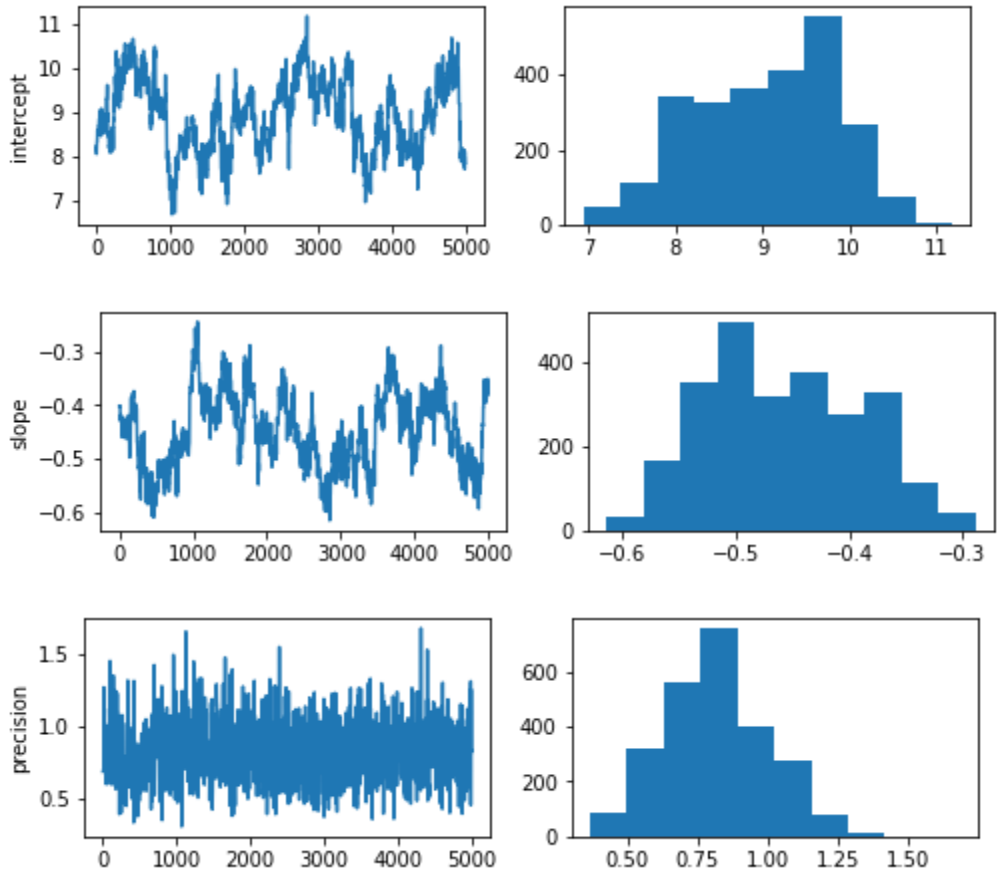


- Result of markov sampling with high variance

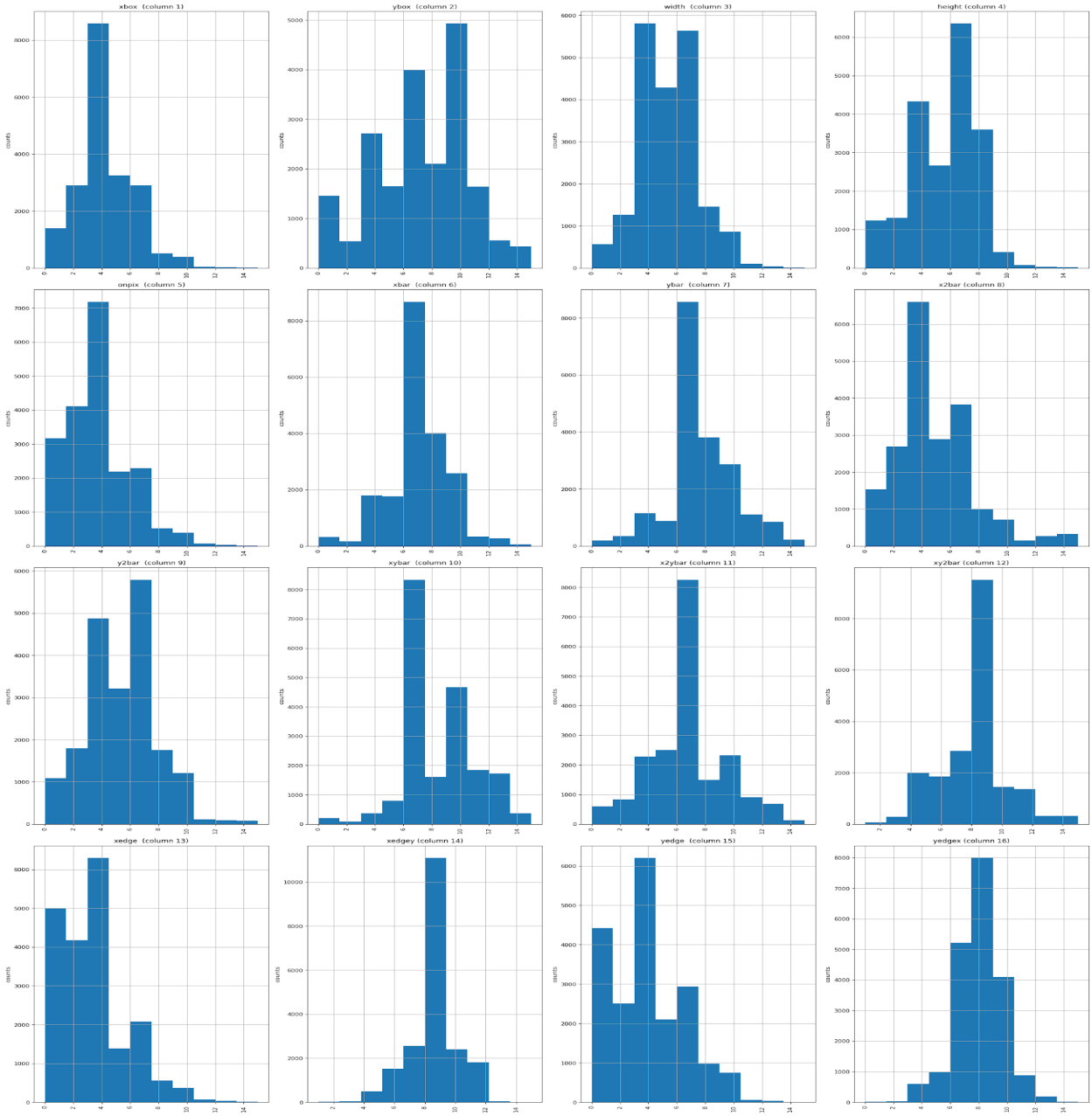




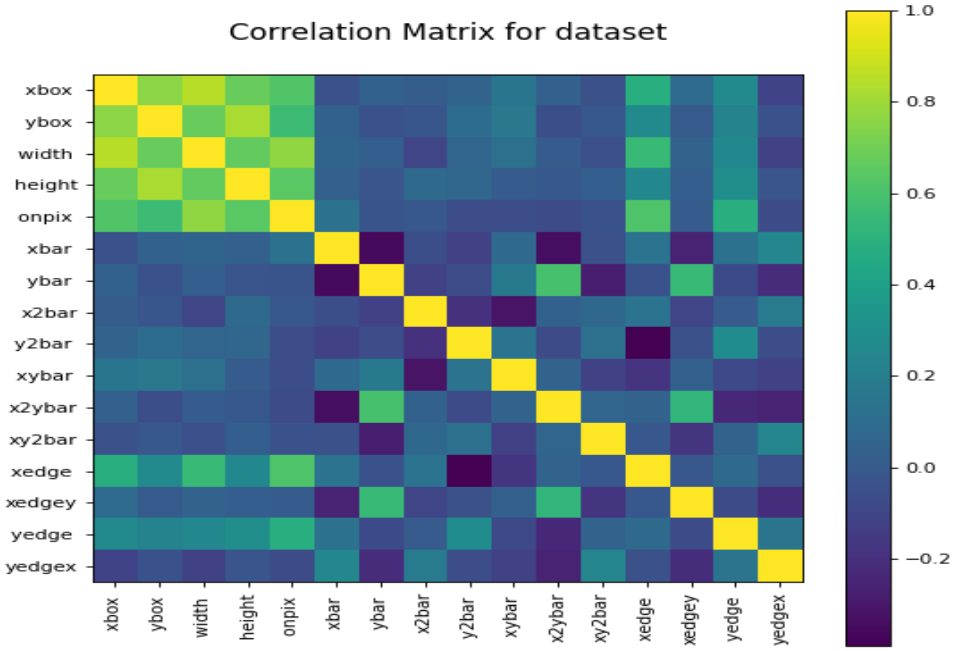
- Result of auto tuned markov sampling



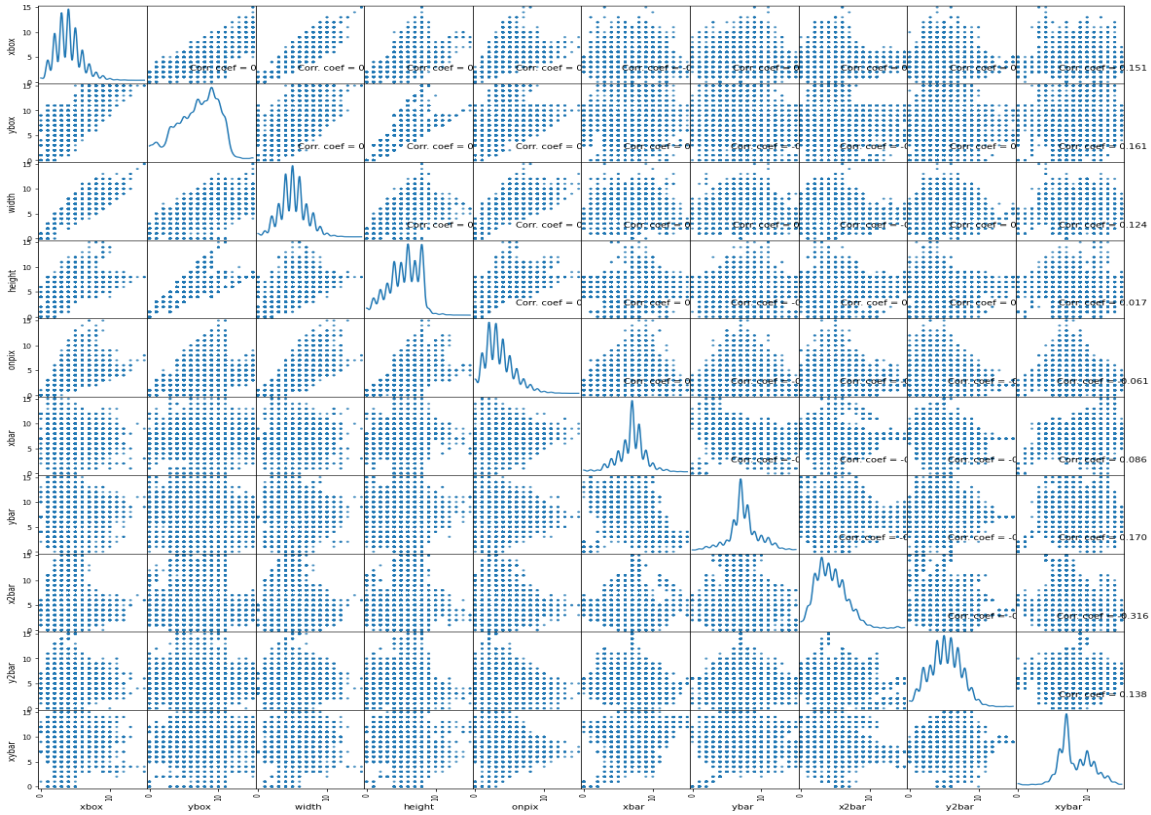
- Data Analysis of column distribution of dataset



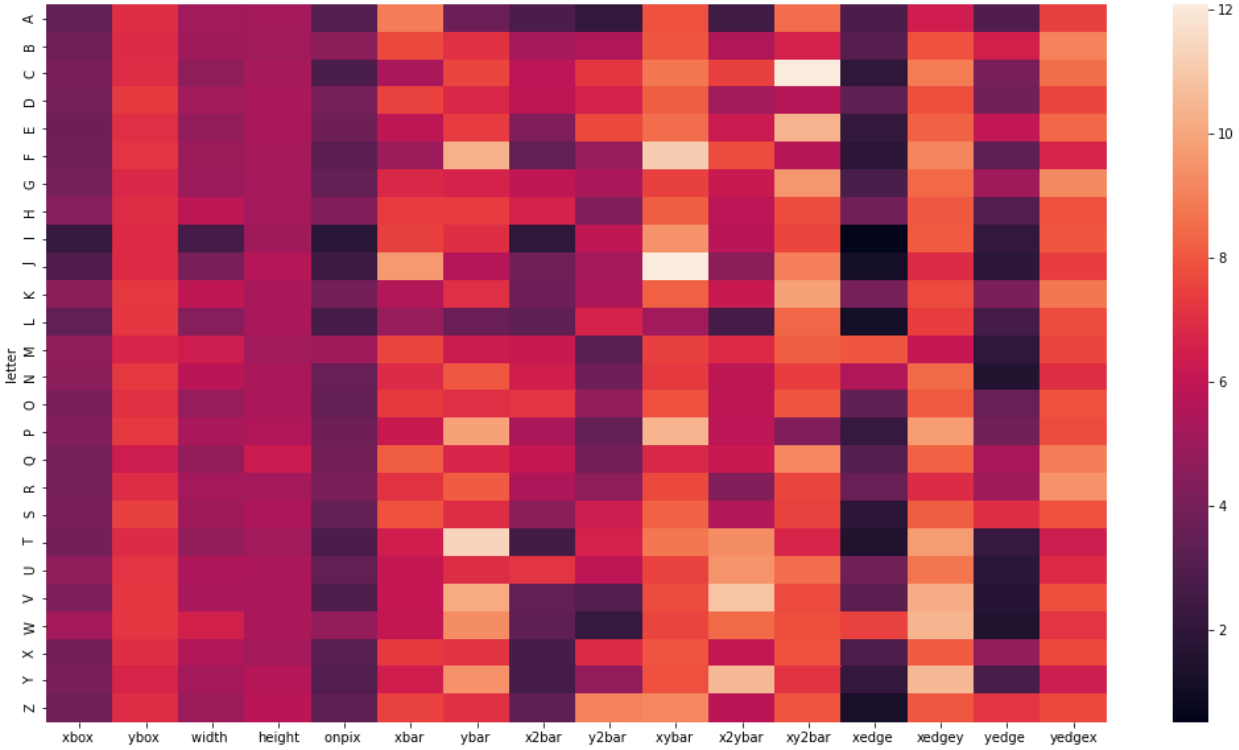
- Result of correlation matrix of dataset



- Analysis of scattering and density of dataset



- Analysis of HeatMap



- **Result**

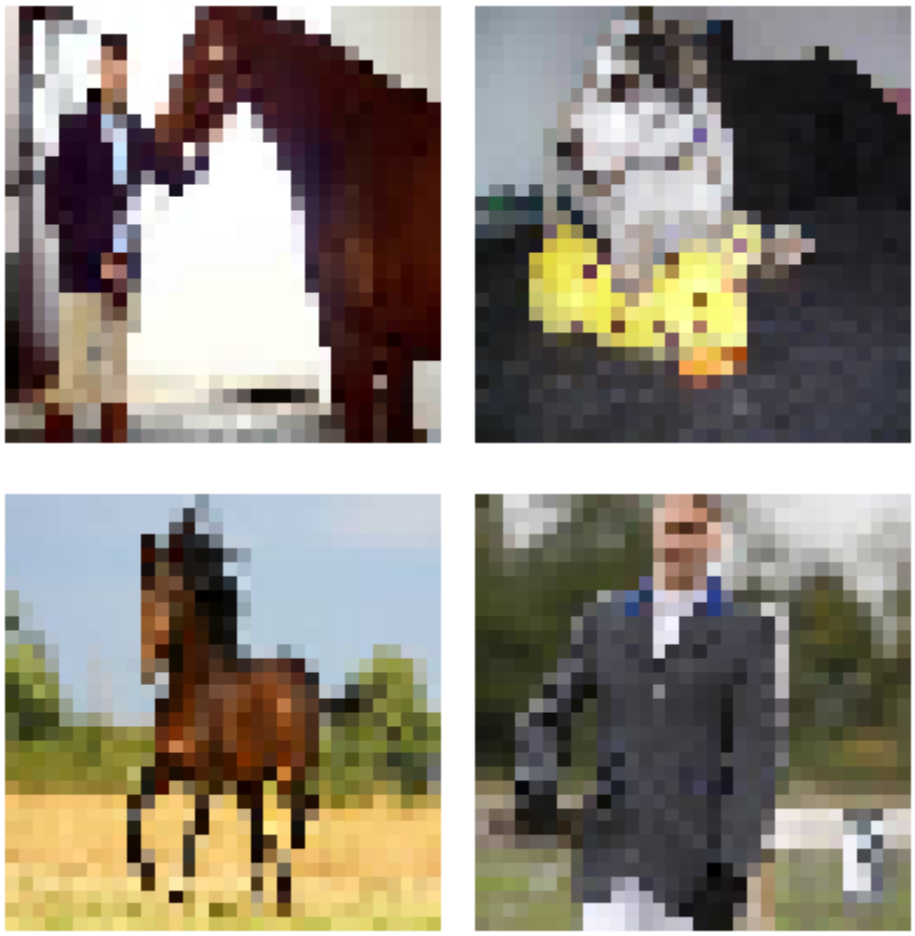
Accuracy of SVM based on markov sampling with different Kernels for letter-recognition dataset.

Kernel	KPCA	SVDD	OCSVM	OCSSVM with SMO	MS_SVM
Linear	0.02	0.09	0.01	0.07	0.8498
RBF	0.05	0.07	0.14	0.09	0.9453
Hellinger	0.01	0.02	0.02	0.13	0.824
chi_square	0.18	0.0	0.02	0.18	0.806

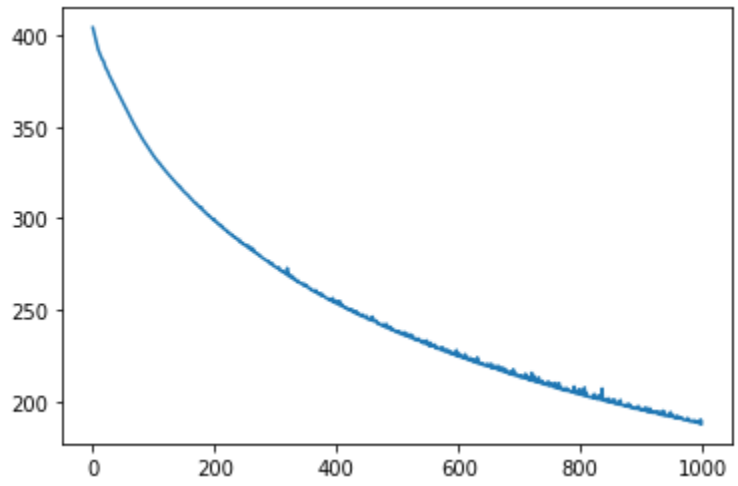
On Pascal Dataset

Observation:

- Preprocessing



- Loss rate for SVM

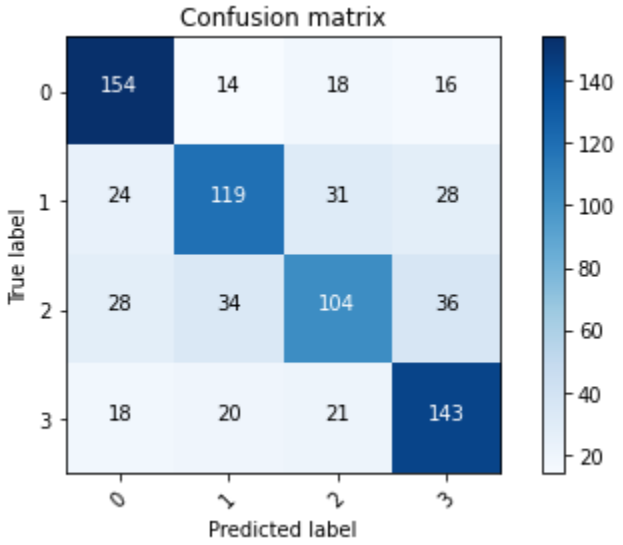


- **Linear kernel**

Accuracy

`0.6435643564356436`

Confusion matrix

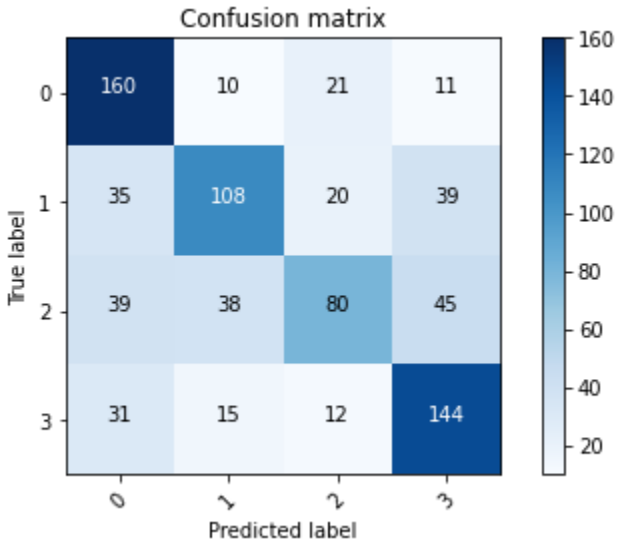


- **RBF kernel**

Accuracy

`0.6089108910891089`

Confusion matrix

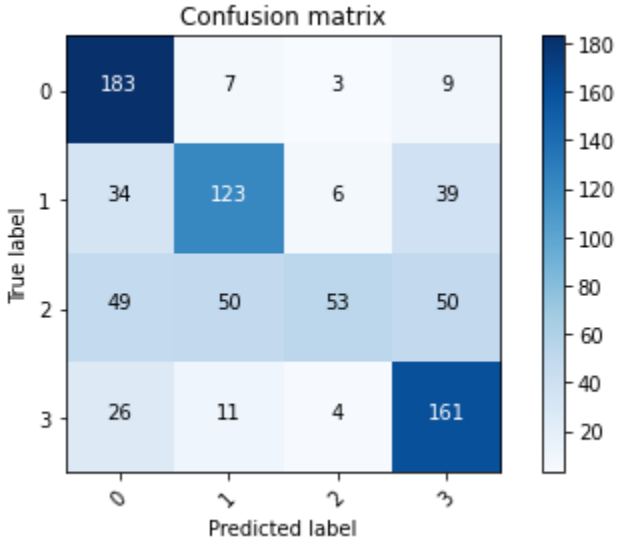


- **Chi_square kernel**

Accuracy

`0.6435643564356436`

Confusion matrix

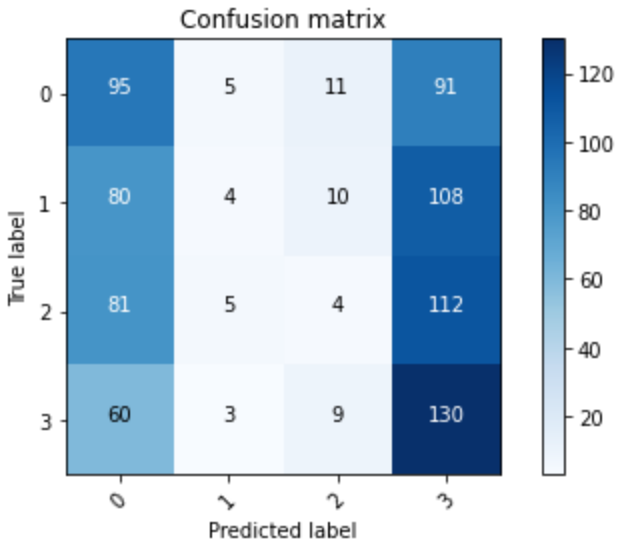


- **Hellinger kernel**

Accuracy

`0.28836633663366334`

Confusion matrix



- **Result:**

Kernel	KPCA	SVDD	OCSVM	OCSSVM with SMO	MS_SVM
Linear	0.02	0.09	0.01	0.07	0.643
RBF	0.05	0.07	0.14	0.09	0.608
Hellinger	0.01	0.02	0.02	0.13	0.28
chi_square	0.18	0.0	0.02	0.18	0.64