BOĞAZIÇI UNIVERSITY

MACHINE LEARNING AND OPTIMIZATION
IE 585

Project

Authors:
Y. Harun Kıvrıl
Kadir Pehlivan
Mısra Şimşir

22 January 2022

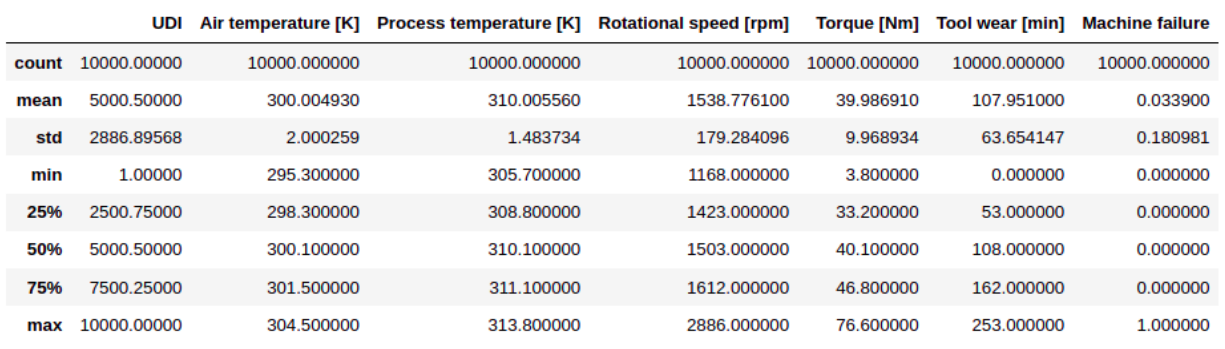
Introduction

In this project we are expected to solve binary classification problem for three different datasets. Binary classification means that output variable can only take any of the two class labels. Binary classification is a supervised machine learning task that employed frequently to solve problems in various areas. For example, in medical it is employed for identifying a disease, in production environment it is used to identify defects, in banks it is used to decide whether to give the credit or not etc. There are many supervised machine learning algorithms to solve binary classification problem in the literature. In this project, two well-known machine learning algorithms which are Logistic Regression and Support Vector Machine are applied for each of the dataset separately. Also, another part of the project is to focus on the results of the optimization algorithms in machine learning. In order to minimize the error rate of machine learning algorithms, two different extension of stochastic gradient and stochastic subgradient algorithms are used for logistic regression and svm respectively. Finally, results of optimization algorithms are compared and discussed at the end.

1 Datasets

1.1 AI4I 2020 Predictive Maintenance

Predictive maintenance is an important concept for the firms that use machines that needs to go down as low as possible by employing as less resource as possible. Firms tend to schedule maintenance activities and determine the size of the maintenance team as well as calculating their risk using this information. Predictive maintenance datasets are hard to find and researchers in this area requires a proper dataset to work on new methods. In order to address this data issue AI4I 2020 Predictive Maintenance dataset [8] is a synthetically generated. It has 10000 observations with 14 attributes in it. 6 of these attributes can be considered as targets. 5 of them is the type of the failure and the other one is machine failure which happens when one of the other failures has occurred. In this task, machine failure has taken as the main target and the rest of the targets removed from the data. However, the other gave an idea about what to expect from the model results.



	UDI	Air temperature [K]	Process temperature [K]	Rotational speed [rpm]	Torque [Nm]	Tool wear [min]	Machine failure
count	10000.00000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000
mean	5000.50000	300.004930	310.005560	1538.776100	39.986910	107.951000	0.033900
std	2886.89568	2.000259	1.483734	179.284096	9.968934	63.654147	0.180981
min	1.00000	295.300000	305.700000	1168.000000	3.800000	0.000000	0.000000
25%	2500.75000	298.300000	308.800000	1423.000000	33.200000	53.000000	0.000000
50%	5000.50000	300.100000	310.100000	1503.000000	40.100000	108.000000	0.000000
75%	7500.25000	301.500000	311.100000	1612.000000	46.800000	162.000000	0.000000
max	10000.00000	304.500000	313.800000	2886.000000	76.600000	253.000000	1.000000

Figure 1: Distribution of features in the dataset.

When the distributions are checked, it is noticed that the scales of the features differ and in order to have similar scale, features they scaled to 0-1 interval with min max scaling. The other thing to notice is the distribution of the target values. Only 3.49% of the data has a machine failure and this would create a bias toward to 0 class. Therefore, the data needed to be randomly oversampled for the training data to create a balance. In probabilistic approaches threshold tuning could be considered as an alternative to oversampling however SVM is not a probabilistic approach and does not allow for threshold trick.

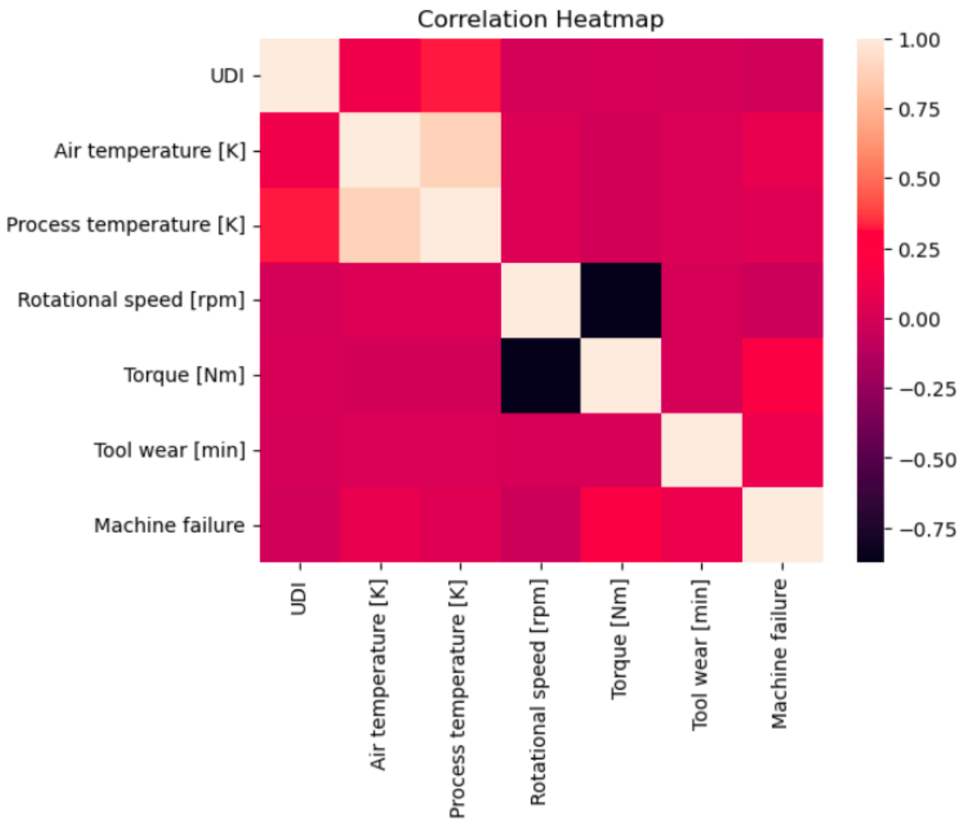


Figure 2: Correlation heatmap of the features in the dataset.

Also correlation heatmap is checked. There are highly negative correlations between some features which requires caution while interpreting results. Also, torque feature has the highest correlation with the target. It is expected that torque will get a large weights in the models. Also, there is a categorical variable which indicates the machine type. It is one-hot-encoded to make it usable in the algorithms.

1.2 Bank

Customer behavior has been a very important issue for a long time for the firms especially for the banks. With the increasing competition in the banking sector, banks started to become more consumer-oriented, and it is aimed to establish long-term relationships with customers by creating customer loyalty. For this reason, they use customer-oriented marketing strategies to be able to reach the customers and they have to constantly observe how effective their methods. In order to understand relation between marketing campaigns and customers' reaction to them, Bank Marketing Data Set was analyzed [9]. The data relates to direct marketing campaigns based on phone calls of a banking company in Portuguese. Data contains 45221 observations with 21 attributes. Input variables are related with customer personal information, last contact of the current campaign and social and economic attributes. 10 categorical variables and 10 numerical variables are used. Target is considered as whether the customer will subscribe a term deposit or not. Data is randomly over-sampled because when we analyze, 11% of the data is in category 1 which means only 4620 out of 45211 customers accept to subscribe a term deposit. This means that there is an unequal distribution of classes which we must handle.

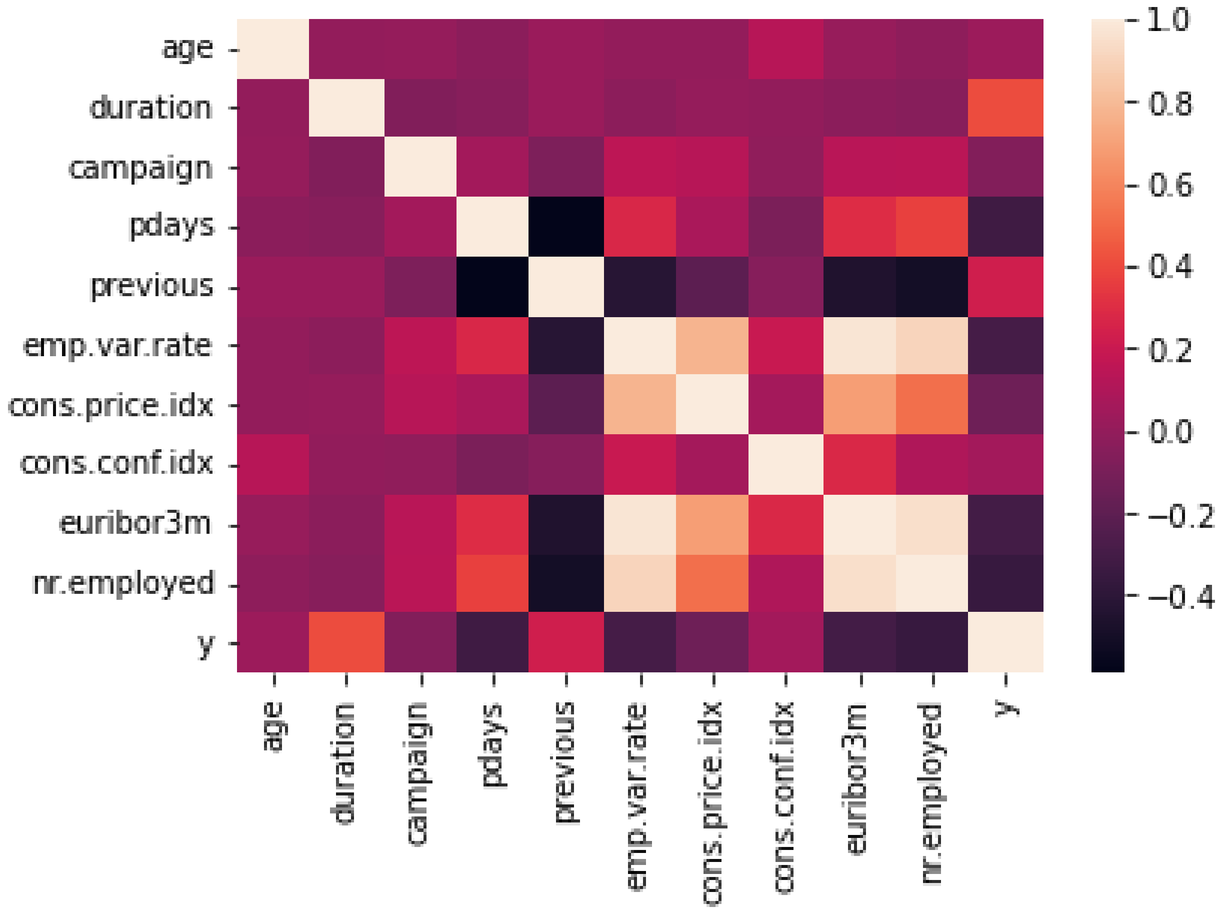


Figure 3: Correlation heatmap of the features in the dataset.

When we look at the heat map, we can see that there is high correlation between duration of last contact and target variable because end of the phone call we obviously know that whether a client subscribe a term deposit or not. Thus, strong correlations can produce high bias for realistic predictive models but since we used data for benchmarking purposes, duration attribute is not discarded from the model.

1.3 Higgs

One of the duties of the modern-day particle accelerators are to collide particles at high speed and observe exotic particles that only occurs at extremely high-energy densities. However, not every collision produces exotic particles and distinguishing such events is a major work in this field. To distinguish them, physicists must distinguish the signal from the noise which is the main motivation of this problem [1]. A data consists of signal as label 1 and noise as label 0 (Called Higgs Data Set) is created to use modern machine learning tools to overcome this problem. Higgs data set is produced by a Monte Carlo simulation. It consists of 11000000 instances and 28 features. 21 of these features are comes from measurements of a particle accelerator. Other 7 are functions of other 21 features [1]. However, in this project we use a friction of this data where our data only has 200000 instances. The data is a balanced data set, and all values are standardized.

2 Logistic Regression

2.1 Maximum Likelihood Estimate

Logistic Regression is a very popular linear modeling approach to binary classification. It assumes a linear decision boundary in the data and calculates the probability estimates by passing the linear boundary through a sigmoid function (σ). In the probabilistic approach this model corresponds to Binomial Distribution with $p_i = \sigma(w^T x_i)$. Therefore the maximum likelihood estimate for p_i values gives the best weights to construct the decision boundary. For this case the likelihood function given N points is:

$$L(w) = \prod_{i=0}^{N} p_i^{y_i}(1-p_i)^{1-y_i}$$

$$= \prod_{i=0}^{N} \sigma(w^T x_i)^{y_i}(1-\sigma(w^T x_i)^{1-y_i}$$

Then the negative log likelihood function becomes:

$$-l(w) = \sum_{i=0}^{N} -y_i \log\left(\sigma(w^T x_i)\right) - (1-y_i)\log\left(1-\sigma(w^T x_i)\right)$$

Finally its derivative gets the following form when the chain rule is applied considering the derivative of $\sigma(x)$ is $\sigma(x)(1-\sigma(x))$:

$$\frac{\partial - l(w)}{\partial w} = ((\sigma(Xw) - y)X$$

2.2 L2 Regularization

Using the L2 regularization in Logistic regression has two advantages. Firstly, it does not allow to fit very complex models and helps to not overfit to training data. Secondly, it can help to adjust conditioning of the loss function which may allow to converge in fewer iterations. When the L2 regularization term is added to the loss function it takes the following form:

$$-l(w) = \sum_{i=0}^{N} -y_i \log\left(\sigma(w^T x_i)\right) - (1-y_i)\log\left(1-\sigma(w^T x_i)\right) + \mu ||w||_2^2$$

and its derivative becomes:

$$\frac{\partial - l(w)}{\partial w} = (\sigma(Xw) - y)X + 2\mu w$$

2.3 Stochastic Gradient Descent

Stochastic gradient and its variants could be considered as the most popular algorithms in machine learning tasks [4]. Since the data in ML tasks tend to have similar observations in the data, selecting a random sample or observation to estimate the gradient makes sense and it decreases the computational cost of the algorithm significantly since the gradient of all points is not calculated at every iteration.

In order to solve the logistic regression problem, two variants of stochastic gradient descent (SGD) is implemented: SGD with momentum and AdaGrad. These variants are relatively known to work well in ML tasks and allowed to obtain reasonable results. Also a trial for averaging variant is made however did not completed due to not getting sufficient results.

During the implementation how to adjust learning rate during the iterations were the one of the challenging steps. Firstly, constant decay rate is considered however having a decay rate introduced another hyperparameter that needs to be tuned and set properly according to theoretical properties of the convergence. Also, constant learning rate scheme is tried and large fluctuations observed during the last epochs. Therefore, recommended the learning rate decay lr_0/k is used. This decay is performed at every epoch to get reasonable results. Another challenge was determining the initial weights and it is observed that having initial parameters in proper range is crucial for this approach

During the implementation, in order to observe the convergence and decrease in loss at every epoch the gradient and loss from whole train data is calculated and printed. The maximum epoch is determined as 40 by considering running times and convergence.

2.3.1 Momentum Stochastic Gradient Descent

Stochastic gradient can be slow and the momentum idea [11] is introduced to fasten learning. Basically it takes a exponentially decaying moving average of the gradients and the parameters continues to move in that direction. This method addresses two issues of the stochastic gradient descent. Firstly, it reduces the variance by accumulating gradient and secondly it helps with poor conditioning for the same reason. The downside of the momentum approach is introducing a hyperparameter that needs to be tuned.

2.3.2 Adaptive Stochastic Gradient Descent

Learning rate can be considered as the most important parameter to make a proper optimization with stochastic gradient and it is difficult to set and adjust. Momentum somehow solves the problem but it has a hyperparameter problem. Therefore, an idea to use separate learning rate for each parameter and adapting this parameters during the learning is presented. Adagrad [3] is one of the algorithms that uses this idea to have adaptive learning rates by scaling the update inversely proportional to the sum of squared historical gradients. Thus, parameters with small partial derivatives gets small learning rate and the ones with large partial derivatives get large learning rates which helps to the optimization procedure.

3 Support Vector Machine

Support Vector Machine is one of the most popular supervised machine learning algorithms. It can be used for both regression and classification problems, but it is mostly used for classification purposes. In this algorithm, we try to find a hyperplane/decision boundary that separates the classes. There can be more than one hyperplane that can possibly choose but we try to find hyperplane such that has maximum distance between closest data points of each class and hyperplane. Support vectors are the closest points of the hyperplane, and we try to maximize the margin between support vectors.

3.1 Cost Function

As a part of SVM algorithm, hinge loss function is applied. The cost or penalty is zero if the predicted value and true value are in the same class, otherwise loss value is calculated as follows:

$$\sum_{i=0}^{N} max\{0, 1-(w_0+w^T x_i)y_i\}$$

which is a sum of maximum of convex functions which means it is convex. Also, regularization term is added to loss function. After that cost function becomes the following form:

$$\frac{1}{2C}w^T w + \sum_{i=0}^{N} max\{0, 1-(w_0+w^T x_i)y_i\}$$

1/2C is a nonnegative parameter that weights the relative importance of optimality and simplicity. [13] Weighting factor which is 1/2C is not known in advance, so it is another tuning issue. We solved each algorithm for a range of values of weighting factor and based on this best C value is chosen.

3.2 Stochastic Subgradient

Subgradient methods are the extension of gradient method for unsmooth convex functions. Instead of using gradients it uses subgradients which is defined at unsmooth parts of a convex function unlike gradients. The stochastic subgradient method is essentially the subgradient method, but using noisy subgradients and a more limited set of step size rules. [14]. For subgradient method, direction is not necessarily a descent direction but overall it moves towards to minimization. When we use stochastic subgradient method, instead of using whole data we use a random sample to estimate subgradient at each iteration. Two well-known variants of stochastic subgradient algorithm were used for solving SVM problem.

3.2.1 Subgradient of SVM Problem

In order to apply the stochastic subgradient method, we need to calculate the subgradient of the loss function. Calculating the gradient of the l2 regularization part was not a problem but hinge loss side is unsmooth threfore we will have an conditional expression. At the end following sub gradient is obtained.

$$\frac{\partial L}{\partial w} = \frac{1}{2C}w + \sum_{i=1}^{N} \begin{cases} -x_i y_i & \text{if } 1-(w_0+w^T x_i)y_i > 0 \\ 0 & \text{if } 1-(w_0+w^T x_i)y_i < 0 \\ \lambda(-x_i y_i) \quad \{\lambda : 1 \geq \lambda \geq 0\} & \text{if } 1-(w_0+w^T x_i)y_i = 0 \end{cases}$$

Similarly

$$\frac{\partial L}{\partial w_0} = \sum_{i=1}^{N} \begin{cases} -y_i & \text{if } 1-(w_0+w^T x_i)y_i > 0 \\ 0 & \text{if } 1-(w_0+w^T x_i)y_i < 0 \\ \lambda(-y_i) \quad \{\lambda : 1 \geq \lambda \geq 0\} & \text{if } 1-(w_0+w^T x_i)y_i = 0 \end{cases}$$

Since any subgradient can be used in the method we choose to set lambda as 0 for simplicity.

3.2.2 Momentum Stochastic Subgradient

Momentum idea is one of the extensions to Stochastic Subgradient Algorithm. While searching for the optimum point in SS, there is a lot of oscillation since algorithm works with just subset of observations so direction of the update has some variance. [11] Momentum method is recommended to reduce these oscillations and therefore to increase the speed of reaching the target. In this method, momentum gradient is used instead of existing gradients.

3.2.3 Adaptive Stochastic Subgradient

It is another extension of Stochastic Subgradient Algorithm. Since it can be a problem to use constant learning rate for both Momentum and Stochastic Subgradient Algoritms, this algorithm is proposed to address this issue. It eliminates the need to tune the learning coefficient manually. At every step, algorithm has learning rate by scaling the update inversely proportional to the sum of squared historical gradients.

4 Tuning Hyper Parameters

After building algorithms we developed a procedure to tune algorithm and model parameters. We perform this procedure and decide best parameter set for each data set separately. We did not develop an algorithm for this procedure, and we followed a brute force method since we observe that the number of possible parameter values are small enough and it will not be computationally challenging. We first split 20% of the data as test data. We perform 5-fold cross validation in the remaining data to decide parameter values and then evaluate them with test set. We first find best values of algorithm parameters (learning rate, batch size, number of epochs we run and momentum (only for momentum stochastic gradient methods)) and fix them to decide model parameters (μ and C) afterwards.

4.1 Algorithm Parameters

To start tuning we first need to decide a set of possible values and run the algorithm for each of them in order to find the best performing one. Even though, we had an idea about good parameter values while developing algorithms, we search in the literature for good values. In the paper written

by Hu, there are significant clues about good values of algorithm parameters[6]. Even with significantly larger data, he run maximum of 100 epochs for his problem, thus we limited our max epoch number with 100 but we reach optimum value before 100 epochs almost every time. Therefore, at certain data sets we decrease the maximum number of epochs to increase the computational speed. In his paper, Hu also mentions that batch sizes from the set {4, 16, . . ., 512} can achieve good generalization results. We observe that it is a common practice to set batch size as power of two. While developing the algorithm we experiment with different learning rates and batch sizes, and we observe that batch size and learning rate are related in which decay of learning rate is equivalent to increase of batch size in terms of convergence rate of the algorithm. We experiment with initial learning rates ranging between 0.1 and 10, and we can always find a suitable batch size that algorithm converge properly. In his paper Smith suggest that instead of decreasing learning rate, increasing batch size yield a better result[12]. Therefore, while tuning we did not consider initial learning rate values smaller than 0.1 but we consider a larger batch size, namely 1024. The moment parameter, which is only used in Momentum Stochastic Gradient methods, must be in between 0 and 1. We choose 4 levels of momentum parameter as 0, 0.5 and 0.9. As a conclusion we set our possible parameter values as follows:
Learning Rate: $\{0.1, 1, 10\}$
Batch Size: $\{16, 64, 256, 512, 1024\}$
Momentum: $\{0, 0.5, 0.9\}$
Max Epochs: $\{100\}$

While computing loss value for different parameter values, we fixed regularization parameters to 0 (We fix C to a large number since it has the form of 1/2C in the hinge loss function). We run our algorithms for every possible combination and since we use 5-fold cross validation, we use mean of their result as the result. At this step we only consider the loss function value. We select the set of parameters that gives the minimum loss value as tuned parameters of the algorithm.

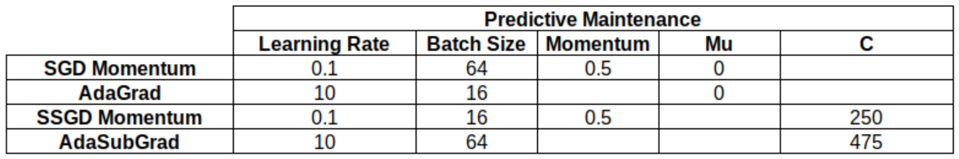
4.2 Model Parameters

After deciding the best parameters for the algorithm, we tried different μ and C values to find the best model parameters of Logistic Regression and Support Vector Machine respectively. Since there is only one parameter to tune at this step, we could try more parameter values. Our approach was to try several larger and smaller values at the same time. The selection of possible values is as follows:
μ: $\{0, 0.001, 0.01, 0.1, 1, 10, 100\}$
C: $\{0.1, 1, 10, 100, 1000, 10000, 100000\}$
These values are only our estimates for possible values, and they are helpful to understand whether the problem requires a strong regularization or not. After finding out the best value, it is possible to extend the search and look for the values different than the one above. However, we found unnecessarily to do so because we observe that it does not change the result in our problem. Unlikely to the tuning of algorithm parameters, we evaluate our candidates based on balanced accuracy score for svm and roc auc score for logistic regression since μ and C values are also part of the loss function and observing only the loss function could be misleading. We preferred roc auc score in logistic regression in order to eliminate the effect of threshold selection. We again use 5-fold cross validation and use the mean of their results as the result. Finally, after tuning the model parameters we evaluate all the selected parameters with the test data that we split before. Since we deal

with imbalanced classification problems, we evaluate them using balanced accuracy score. If we get satisfactory results we have done with the problem. Otherwise, we needed to turn back to the beginning. Luckily this was not the case in any of the data sets and algorithms.

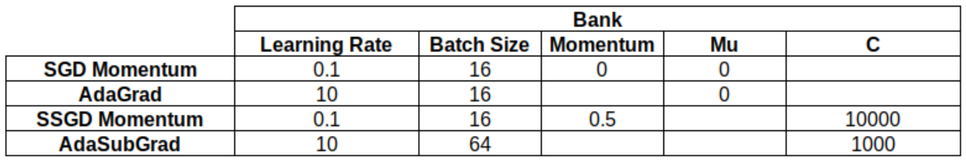
5 Results & Discussions

Results have two parts, first part presents the selected algorithm parameters during the cross validation evaluation with training loss together with selected model hyperparameters that evaluated using roc auc score or balanced accuracy. Second part shows the predictive performance of the algorithms in the datasets. We also presented some example loss plots for train and test data to show how our algorithms converge during iterations.



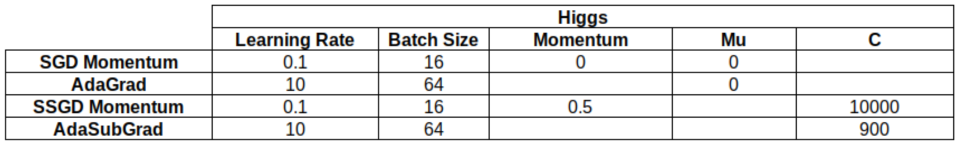
	Predictive Maintenance				
	Learning Rate	Batch Size	Momentum	Mu	C
SGD Momentum	0.1	64	0.5	0	
AdaGrad	10	16		0	
SSGD Momentum	0.1	16	0.5		250
AdaSubGrad	10	64			475

Figure 4: Selected Parameters for Predictive Maintenance Dataset



	Bank				
	Learning Rate	Batch Size	Momentum	Mu	C
SGD Momentum	0.1	16	0	0	
AdaGrad	10	16		0	
SSGD Momentum	0.1	16	0.5		10000
AdaSubGrad	10	64			1000

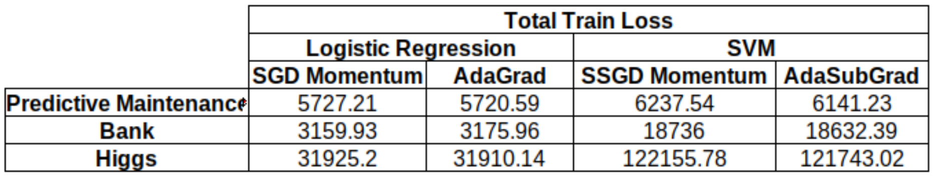
Figure 5: Selected Parameters for Bank Dataset



	Higgs				
	Learning Rate	Batch Size	Momentum	Mu	C
SGD Momentum	0.1	16	0	0	
AdaGrad	10	64		0	
SSGD Momentum	0.1	16	0.5		10000
AdaSubGrad	10	64			900

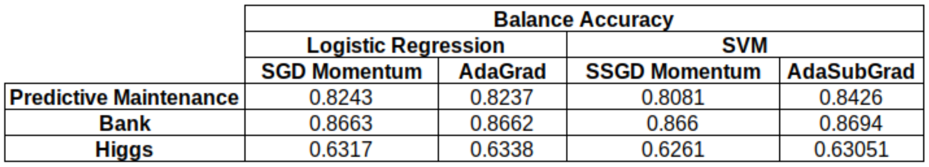
Figure 6: Selected Parameters for Higgs Dataset

It appears that parameters change according to dataset and learning algorithm. Even for model parameters we see changes in the same dataset. We reasoned this situation with algorithms optimization performance. When one of the is not as good as the other another model parameter can be selected. It seems like μ always gets 0 value. This may due to the search range of the parameter. Since tuning is computationally expensive we had to limit ourselves to a small search space. Also, there may no need for a regularization since the datasets does not have many features and overfitting may not be the case.



	Total Train Loss			
	Logistic Regression		SVM	
	SGD Momentum	AdaGrad	SSGD Momentum	AdaSubGrad
Predictive Maintenance	5727.21	5720.59	6237.54	6141.23
Bank	3159.93	3175.96	18736	18632.39
Higgs	31925.2	31910.14	122155.78	121743.02

Figure 7: Loss Values on Train Data



	Balance Accuracy			
	Logistic Regression		SVM	
	SGD Momentum	AdaGrad	SSGD Momentum	AdaSubGrad
Predictive Maintenance	0.8243	0.8237	0.8081	0.8426
Bank	0.8663	0.8662	0.866	0.8694
Higgs	0.6317	0.6338	0.6261	0.63051

Figure 8: Balanced Accuracy Values on Test Data

For logistic regression, the model parameter μ is the same for each variant therefore their training loss is comparible. It seems like the differences are small and algorithms perform better than the other in at least one dataset. Also when the predictive performance checked. It appears that the same situation applies to balanced accuracy values on the test data. For svm the loss values are not exactly comparable, however we see better loss values with less C value for AdaSubGradient therefore it is possible to say that this algorithm reached better parameters in our setting. Additionally in terms of test performance AdaSubGrad algorithm outperforms the other SVM optimization algorithm in every dataset.

When the performance of Logistic Regression and SVM algorithms are compared, the performances does not differ significantly for higgs and bank datasets however AdaGrad SVM optained significantly better balanced accuracy for the predictive maintenance dataset. Therfore in our setting SVM could be considered as a better option to Logistic Regression. However it was harder to optimize due to its unsmooth nature. Also this results may be checked with a larger parameter tuning search space with faster implementations.

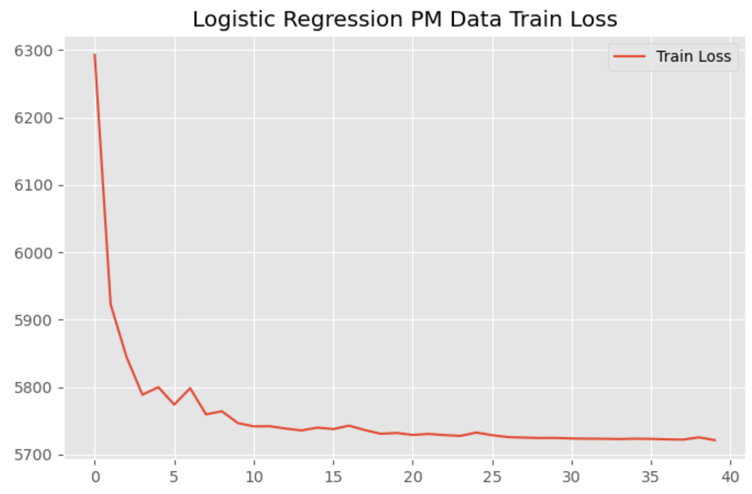


Figure 9: Predictive Maintenance Training Loss for Adagrad Logistic Regression.

This example train plot for AdaGrad algorithm shows that the loss values are converged and expectedly at every iteration it decreased. The few spikes are interpreted as the result of stochasticity in the algorithm.

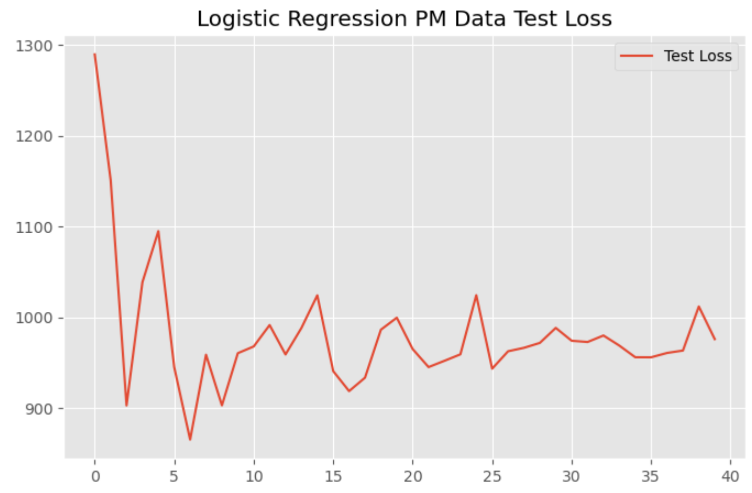


Figure 10: Predictive Maintenance Test Loss for Adagrad Logistic Regression.

In terms of test we don't expect a graph constantly decreasing since these point are not optimized during the process. However it decreases at first epochs stays around this level which shows the behavior of generalization.

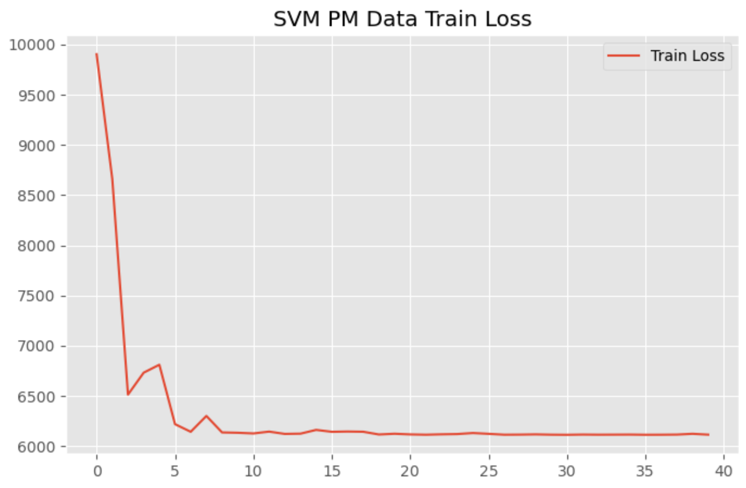


Figure 11: Predictive Maintenance Training Loss for Adagrad SVM.

In the AdaSubGrad train loss we see a similar plot to the logistic regression case and it is possible to say it is converged. Also it seems like it reached convergence in 10 epochs which is much faster than the logistic regression example above.

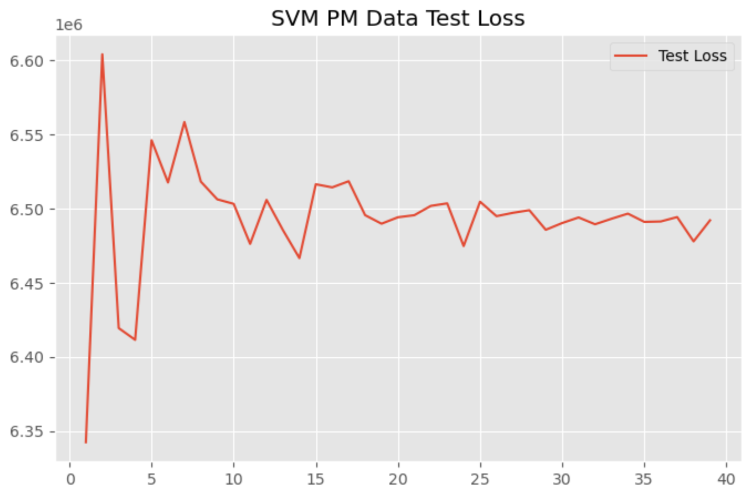


Figure 12: Predictive Maintenance Test Loss for Adagrad SVM.

Finally, the test loss for AdaSubGrad test loss seems like logistic regression case.

6 Conclusion & Future Work

In conclusion, we had three data sets at hand where a classification problem can be implemented. We use two different methods to solve this classification problems. For this purpose, we implemented a stochastic gradient algorithm and a stochastic subgradient algorithm. We further developed our algorithms by using more well known different variants of them. We tuned the parameters of algorithms and models using cross validation and measured the performance based on a out of sample data. Lastly, we evaluated and compared implemented methods in general.

As a future work, in our setup batch size is fixed at all epochs. It is possible to implement an adaptive batch size selection procedure. It can improve the algorithm speed and accuracy by selecting smaller batch sizes at the beginning of the algorithm and larger batch sizes towards the end of the algorithm. In this problem we implement two different versions of stochastic gradient method namely Momentum Stochastic Gradient Method and Adaptive Stochastic Gradient Method. They perform well in our setup and gives satisfactory results. It is possible to improve the efficiency of the algorithms by using different versions of Stochastic Gradient Method such as, Averaging, Lbfgs etc. Additionally, in the tuning phase, we restricted our search space to conclude in a reasonable time. However, it is possible to extend the value range of the parameters or even it is possible to build efficient implementations for parameter tuning.

References

[1] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*, 5(1), 2014.

[2] Alejandro Celis, Victor Ilisie, and Antonio Pich. Lhc constraints on two-higgs doublet models. *Journal of High Energy Physics*, 2013(7):1–44, 2013.

[3] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[5] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[6] Yaochen Hu, Amit Levi, Ishaan Kumar, Yingxue Zhang, and Mark Coates. On batch-size selection for stochastic training for graph neural..., Mar 2021.

[7] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[8] Stephan Matzka. Explainable artificial intelligence for predictive maintenance applications. In *2020 Third International Conference on Artificial Intelligence for Industries (AI4I)*, pages 69–74. IEEE, 2020.

[9] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Boris T Polyak. Some methods of speeding up the convergence of iteration methods. *Ussr computational mathematics and mathematical physics*, 4(5):1–17, 1964.

[12] Samuel L. Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V. Le. Don't decay the learning rate, increase the batch size, Feb 2018.

[13] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for machine learning*. Mit Press, 2012.

[14] Stanford University. Ee 364b stochastic subgrad notes. `https://web.stanford.edu/class/ee364b/lectures/stoch_subgrad_notes.pdf`.