COMP5318 Assignment 1 Report

Student name: Zedong Wen, Han Liu

SID: 500334863, 480010528

1 Introduction

1.1 What is the aim of the study?

This study aims to build powerful and applicable classifiers which could classify grayscale images of size 28x28 pixels into 10 given classes with high accuracy, which are *T-shirt/Top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.* More specifically, the training set is 30,000 observations each containing 784 features. According to the training set, the model is trained, and the result should be the classification of each example. Then, the method to verify the accuracy of the model is to fit the trained model with the provided test set of 5,000 observations.

1.2 Why is the study important?

The importance of this research is mainly reflected in the following aspects. First, using different machine learning models to analyze problems can deepen the understanding of machine learning models and strengthen basic knowledge. Secondly, due to the particularity of data sets, data preprocessing technology is the key to solve this research. It is necessary to find and learn preprocessing methods suitable for processing large quantities of images. Furthermore, model tuning is also an essential step to build an excellent model, and a complete grasp of this knowledge is conducive to the next machine learning content. In the long term, this image classifiers based on machine learning can automatically extract features from images, thus greatly improving the efficiency of image classification. The high accuracy image classifiers could also improve the convenience of image search. For example, customers can search through blurred clothing images to find exactly the category of clothing they want to know about or buy. Moreover, the technology of image classifier can develop various image search engines. A detailed analysis report of predicting image classification labels on the test dataset will be given below, including the data preprocessing methods for the given dataset, the brief introduction of the chosen classifier models and parameter tuning processes that been attempted as well as their results. In this study, after trying various classification model algorithms and comparing the accuracy of each model on test set, it was found that support vector machine and KNN were the best two models in this experiment, and their accuracy was respectively 92.8% and 90.3%.

2 Methods

2.1 Data description

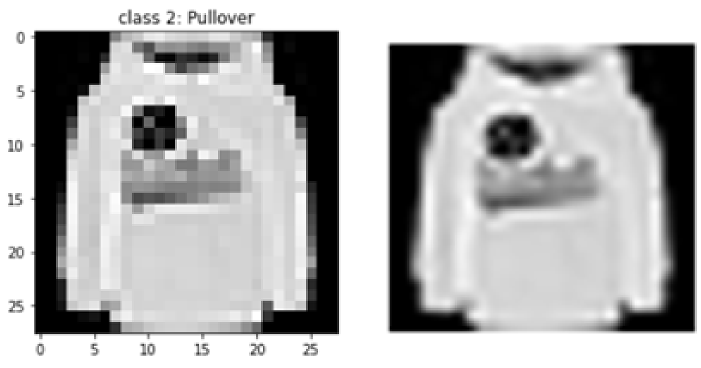
The provided train dataset contains in 784 latent features associated with the clothing classification label, with a sample space of 30,000 observations. Each row of sample data can be displayed as a grayscal e image with a size of 28x28, and these large number of feature factors provide various information of the grayscale image comprehensively. Since not all features are useful for label classification by the classifier, it is crucial to handle some redundant features in the data preprocessing steps to avoid overfitting and improve classification efficiency. The test set without a given response variable has the same 784 feature variables as the training set, with a total of 5,000 observations.

2.2 Data pre-processing

After confirming that there is not exist missing value in both training and testing datasets, we considered and tried three steps to preprocess the dataset, histogram of oriented gradient (HOG), normalization and principal component analysis (PCA) respectively.

2.2.1 Histogram of oriented gradient (HOG)

HOG is used to extract features from our image data. It divides the image into small connected areas, and then collects the gradient or edge direction histogram of each pixel in the cell unit. Then these histograms can be combined to form a feature descriptor [4]. The core idea is that the shape of the detected local object can be described by the distribution of gradient or edge direction. HOG can capture the local shape information well and has good invariance to geometric and optical changes. Through experiments, when using our original image size, HOG can only extract few useful features, which has little effect on subsequent classifiers training. Hence, in order to ensure the same number of HOG features extracted from each image, we resized all images to the same size 64x64 and obtained 1764 feature factors. The greyscale image after resizing is compared as follows (the image on the left is size 28x28 (Figure1(a)) and the one on the right is size 64x64(Figure 1(b))):



(a) 28x28 pixels (b) 64x64 pixels

Figure 1: Example of 28x28 pixels and 64x64 pixels

2.2.2 Normalization

For purpose of eliminate the dimensional influence between the features, Min-Max normalization was performed on the data set processed by HOG in the previous step. Min-max normalization is a linear transformation of the data such that the original resulting values are mapped between [0, 1]. In the normalized data set, each feature index is in the same order of magnitude, which is suitable for comprehensive comparative evaluation.

2.2.3 Principal component analysis (PCA)

The last process in data pre-processing of this study is principal component analysis. After completing Hog feature extraction, the original dataset retains a huge number of 1746 feature factors. It is indeed critical to reduce the dimensionality of this dataset and improve interpretability while minimizing information loss [3]. The technique of PCA helps to find the optimal dimensionality reduction parameters by creating new uncorrelated variables to continuously maximize variance. The PCA plot (Figure 2) of our dataset below shows that retaining 95% of the principal components is the best parameter, and the data dimension is reduced to 473 features.

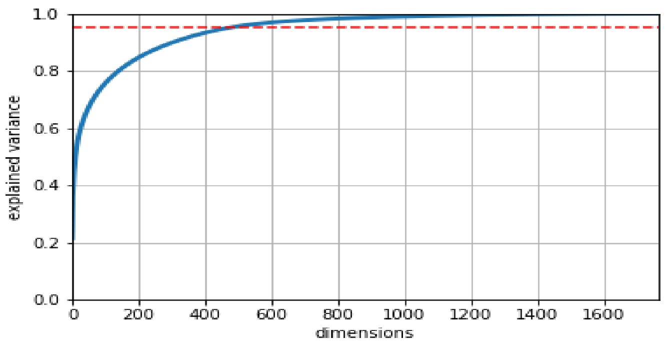


Figure 2: Cumulative variance contribution

Note that the same data pre-processing methods has been applied on both the training and testing datasets.

2.3 Classifier methods algorithms

In this study, we tried seven classifier methods to find the ones with higher label classification accuracy, the algorithms are K-Nearest Neighbor, Logistic Regression, Naive Bayes, Decision Tree, Support Vector Machine, Bagging and Random Forest respectively. Among them, the latter two are ensemble methods. The rationale of these seven classifier methods algorithms and explanations of their applicability will be described below. The choice of hyperparameters for each model and their effects are also presented after the model interpretation (Table 1).

2.3.1 KNN

As a supervised learning classification algorithm, the KNN classifier calculates the distance between the unclassified data features and the training data features then sorts them, extracts K training data features with the closest distance, and then the category of the test data is determined based on the categories of the k features. The advantage of this algorithm is simple to implement but it also has higher computation cost.

2.3.2 Logistic regression

In the logistic regression algorithm, a logistic function is used to model the probabilities of the possible outcomes of each trial and are classified according to the probabilities. This classifier is only valid when the predictor variables are binary whereas it is useful when understanding the effect of multiple independent variables on single response variable.

2.3.3 Naive Bayes

Naive Bayes algorithm based on Bayes' theorem, the advantage is the high learning efficiency. However, since the algorithm is based on the assumption of conditional feature independence, the accuracy of the algorithm will be affected.

2.3.4 Decision trees

The decision tree algorithm implements classification through the branching structure of the tree. For the given data features and their classes, the decision tree generates a series of rules to classify each sample. Decision tree is easy to understand but small changes in dataset can result in completely different tree.

2.3.5 SVM

Considering the training data as points in the hypergeometric space, the SVM algorithm finds a classifier that maximizes the distance between the hyperplane and the nearest points. Mapping unclassified test data into the same space and classify them based on where they fall[2]. Support vector machine algorithm is effective in high-dimensional sample space, but it is difficult to solve multi-classification problems.

2.3.6 Bagging

The Bagging algorithm is a technology that reduces the generalization error by combining several models, improves the accuracy of the classifier, and meanwhile reduces the variance of the results to avoid overfitting.

2.3.7 Random forest

The random forest classifier fits multiple decision trees by drawing samples that with replacement and uses the average to improve the model's classification accuracy and control overfitting. Its disadvantages are that the parameters are more complex, and the model training and label classification are slow.

Algorithms	Method	Hyperparameter	Effects
KNN	GridsearchCV	n_neighbors	Indicates the number of votes to select the nearest K points
		P	Indicates the selected distance type
Logistic Regression	GridsearchCV	C	The smaller the C, the stronger the regularization ability
Naive Bayes	K-folder CV	CV	Indicates the k-folder cross validation
Decision Tree	GridsearchCV	criterion	Feature selection criteria
		max_depth	Maximum depth of decision tree, commonly used to solve overfitting
SVM	GridsearchCV	C	When C is larger, the accuracy of the classifier will be higher, but the error tolerance rate will also be lower, and the generalization ability will be worse
		gamma	Implicitly determines the distribution of the data after mapping to the new feature space
Bagging	GridsearchCV	n_estimators	Generate n weak classifiers
		max_samples	Controls the size of the subset (for samples and features)
Random Forest	GridsearchCV	n_estimators	It is the number of trees in the forest. Generally, the larger the number, the better the effect, but the calculation time will also increase
		max_leaf_nodes	The larger the number of samples required for splitting or the larger samples required for leaf nodes, the more simple the submodel
		max_feature	It is the size of a random subset of features considered when splitting nodes. The lower the value, the more the variance is reduced, but the more the error is increased

Table 1: Accuracy of algorithms with default settings

3 Experiments result and discussion

In the above, the selection and role of hyperparameters in the model are explained. In this section, it will be focused on choosing the best parameters for each classifier and finding the best model among the seven models tried. Additionally, the fact sheets and in-depth exploration of these experiments are discussed later.

3.1 Algorithms with default settings

Initially, implement the algorithm on the default settings, the default accuracy is used to compare with the accuracy after hyperparameter tuning. The training set accuracy corresponding to each model is shown in the table below (Table 2).

Classifier Model	Training set accuracy
KNN	0.927
Logistic Regression	0.928
Naive Bayes	0.787
Decision Tree	1.000
SVM	0.961
Bagging	0.994
Random Forest	1.000

Table 2: Accuracy of algorithms with default settings

3.2 Fine tuning the hyper-parameters

In most of the classification models we experimented with, the method of Grid search cross-validation was used for optimal hyperparameter search, except for Naive Bayes classifier. Grid search is a hyperparameter optimization method that iterates through loops, trying each candidate parameter possibility, and selecting the hyperparameter that gives the best performance of the estimator. The following table (Table 3) details the parameter tuning method, hyperparameter choices, parameter ranges, optimal parameter values, and best cross-validation scores for each algorithm.

3.2.1 KNN

For KNN, the n_neighbors indicates the number of votes for the nearest K points, p indicates the selected distance type thus the best parameter of n_neighbors is 5 means selecting the nearest 5 points to vote and the best parameter of p is 2 means choosing the Euclidean distance. The best cv score based on this estimator is 0.89.

3.2.2 Logistic Regression

For Logistic Regression, the best estimator when C = 0.1 means the regular coefficient is locally adjusted to 0.1 to preserve image features and fine part.

3.2.3 Naive Bayes

Since Naive Bayes has no hyperparameters to adjust, the K-folder cross-validation method is used and sets K to 5, it also has the lowest average cross-validation score of 0.78 among all models.

3.2.4 Decision Tree

In decision tree modeling, the best parameters obtained by grid search are Criterion is entropy and Max_depth is 10.

3.2.5 SVM

The best cross validation score of SVM is 0.91 which is the highest among the seven models. The penalty coefficient of the objective function (C) is 10 and the coefficient of kernel function (gamma) is 0.01.

3.2.6 Bagging

As an integrated method, bagging should take the running time into consideration when tuning parameters. Therefore, we set the range of n_estimators to [200, 300, 500] and max_samples to [100,300,500] and best values are 500 and 500 respectively.

3.2.7 Random Forest

The parameter tuning process of random forest is relatively complicated. First, it is necessary to define a large range for hyperparameters that need to be optimized. The best estimator based on the given range is RandomForestClassifier (max_features = 'log2', max_leaf_nodes = 30, n_estimators = 500).

Algorithms	Method	Hyperparameter	Range of parameter	Best parameter	Best CV score
KNN	GridsearchCV	n_neighbors	[1,3,5]	5	0.89
		P	[1,2]	2	
Logistic Regression	GridsearchCV	C	[0.1,1,10,100]	0.1	0.90
Naive Bayes	K-folder CV	CV		5	0.78(average)
Decision Tree	GridsearchCV	criterion	['gini','entropy']	entropy	0.80
		max_depth	[5,10,15]	10	
SVM	GridsearchCV	C	[0.1,1,10]	10	0.91
		gamma	[0.01,0.1]	0.01	
Bagging	GridsearchCV	n_estimators	[200,300,500]	500	0.80
		max_samples	[100,300,500]	500	
Random Forest	GridsearchCV	n_estimators	[200,300,500]	500	0.83
		max_leaf_nodes	[20,30]	30	
		max_feature	['auto','sqrt','log2']	log2	

Table 3: Accuracy of algorithms with default settings

3.3 Results of experiments

After fitting the seven models selected above, we use GridSearchCV to automatically perform cross-validation on the given model, then track the results by adjusting each parameter and finally output the optimal score. The test set accuracy rate was used as the main indicator for selecting the best model for this classification case among a set of candidate models. Support Vector Machine and K-Nearest Neighbor significantly outperform the other algorithms in the case of the highest and second highest accuracy on the test set. Here is a list of used classifier models and their training set accuracy, test set accuracy and GridSearchCV score (Table 4)

Classifier Model	Training set accuracy	Test set accuracy	CV score	Running time(seconds)
KNN	0.927	0.903	0.890	13.110
Logistic Regression	0.920	0.768	0.900	16.150
Naive Bayes	0.787	0.768	0.780	0.360
Decision Tree	0.883	0.897	0.800	67.930
SVM	0.994	0.928	0.910	93.240
Bagging	0.811	0.798	0.800	174.450
Random Forest	0.840	0.829	0.830	105.250

Table 4: Accuracy of algorithms with default settings

It clearly shows that the best 2 models are SVM and KNN, the test set accuracy are 0.928 and 0.903 respectively; the Logistic Regression and Naïve Bayes are the worst 2 models, the same accuracy is 0.768. The result of Figure 3 is more visual.

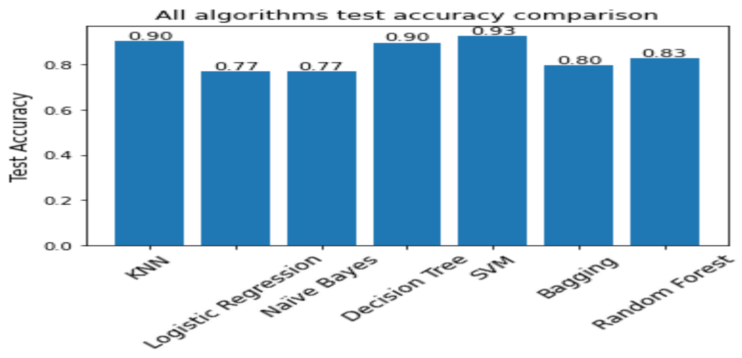


Figure 3: Cumulative variance contribution

Furthermore, training and inference time of each model also serves as a reference for model evaluation. As shown in the histogram (Figure 4) below, the model fitting time of SVM is 93.24 seconds. As the model with the highest accuracy in the test set, its time is significantly lower than that of the ensemble methods.

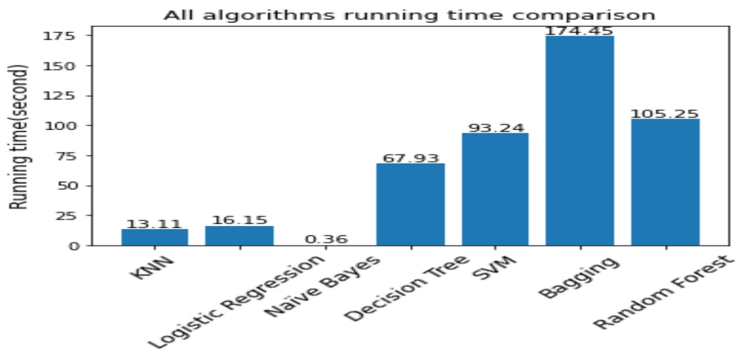


Figure 4: Cumulative variance contribution

3.4 Discussion

The following discussion revolves on the interpretation of the experimental results, the comparison between the pre-experimental expectations and the experimental results, and the advantages and limitations of this experiment.

3.4.1 Interpretation of experiments results

These results show that among the fitted image classifiers for this training set, the most accurate classification algorithm is Support Vector Machine, followed by K-Nearest Neighbor, and the least accurate algorithms are Naive Bayes and Logistic Regression. The interpretation of the results of the model and the possible reasons for are expounded in turn.

Two most accurate models

SVM

Based on the test performance on Kaggle, the fine-tuning model of Support Vector Machine algorithm has the highest accuracy of 92.8%. Its favorable outcome is closely related to the fact that the tuning of hyperparameters affects the generalization error [5]. Another matter which causes the high classification accuracy is the point that all features have the same pixel density and similar range. Since the classifier is only determined by the support vector, SVM can also effectively avoid overfitting. In view of the execution time is 93.24s, SVM deserves to be the most successful model in this study.

KNN

The high accuracy of the KNN model for label classification shows that the normalization in the data preprocessing is effective and improves the classification performance due to the reduction in the number of features. In addition, the KNN model is very sensitive to the selection of the K value, which also confirmed that the appropriate K value is selected in the process of parameter tuning.

Two lowest accurate models

Naive Bayes
On the contrary, the Naive Bayes model performs poorly on the label classification of the test set because it is a simple probabilistic classifier based on the impendence of features, but this dataset contains 784 features on grayscale images, although principal component analysis has been done on the dataset, the high correlation between features is unavoidable.

Logistic Regression

regression classifiers performed poorly as well as Naive Bayes. The test set accuracy is only 0.768. The reason for its low accuracy is under-fitting. Although Logistic Regression model is highly interpretable with the output value falls between 0 and 1, it is essentially a linear classifier, Therefore, it cannot deal with high dimensions and high complexity data set.

Models that behave in general

Decision Tree
The accuracy of the decision tree is 0.897, which is inferior to the KNN model. In hyperparameter tuning of the decision tree, the criterion is set to entropy. Entropy is a measure of the impurity in given features, specifying the randomness of the data, it could calculate how much information each feature gives us about the class. The main problem with implementing decision trees is that the optimal values for the root node and sub-node are not chosen when tuning the hyperparameters.

Random Forest
Unexpectedly, accuracy of random forest on the test set is 0.829. As a meta-estimator, random forest fits multiple decision tree classifiers in an ensemble way. Considering the best segmentation of features, log2 is chosen as max_features then max_features=log2(473 features). The max_leaf_nodes is set to 30 to reduce feature impurity relatively. The reasons for the poor performance will be analyzed in detail in the comparison section below.

Bagging
Low accuracy of Bagging algorithm is 0.798. Bagging mainly focuses on reducing variance. If the sub-models are completely independent of each other, the variance can be significantly reduced; if they are completely correlated, the variance cannot be reduced. Therefore, the poor performance of bagging may be due to the correlation between the sub-models.

3.4.2 Comparison between expectations and findings

In our pre-experiment expectations, the random forest and SVM models are the most promising. Random forest is suitable for large-scale data and the characteristics of strong generalization ability often make the results have high accuracy. Also, SVM have shown good performance in classification problem of dataset. However, Random Forest achieved only 82.9% accuracy on the test set in this experiment, ranking fourth among the seven models built. It may because the number of decision trees is not enough, although the number of trees in the random forest is critical to the performance of the model, it is also accompanied by an increase in the running time cost.

3.4.3 Strengths and limitations

Although the classification accuracy of the SVM and KNN models is significant in this case, it is also limited in some special scenarios. SVM model may face challenges in terms of running time and memory usage when the data volume reaches 100,000 or more, and it need to be very careful about data preprocessing and parameter tuning when use this model. For the KNN model, all it does in the training phase is to save samples, there is no obvious training process, and the accuracy is difficult to improve within a certain range.

Through learning more about relevant cases for this study, it was found that compared to other similar grayscale image classification experiments, the Convolutional Neural Network and Pytorch are also great ways to get high accuracy results. To be more specific, the example of MNIST handwritten digit classification, whose researchers developed convolutional neural network models and achieved classification accuracy rates of more than 99% [1]. Another similar example, whose authors also classified grayscale images of clothes, ended up with 94.64% accuracy by using CNN model. In-depth learning of machine learning and deep learning can help solve such problems more effectively and improve model accuracy.

4 Conclusion

In summary, this study is contrived to build an ideal image classification model that can classify a given grayscale image of clothing into one of 10 classes representing integer values ranging from 0 to 9. This study mainly conducts experiments from four aspects: data preprocessing, model selection, hyperparameter tuning and model evaluation, and draws the conclusion that the model with the highest classification accuracy is the support vector machine and the KNN model is the second-best model.

Besides, these findings provide additional information about the advantages of these two models. The SVM model effectively distinguishes categories because of the advantages of feature mapping. The KNN model is simple and robust to noisy dataset.

Nevertheless, there are still flaws in data processing and model selection:

1. The correlation between the features is ignored in the data pre-processing, which may change this feature.

2. In model selection, the random forest may find a better range of hyperparameters to improve the accuracy of the model while allowing time consumption.

The potential improvements of this can be pick larger range for the hyperparameters and train the model for a longer time.

In future work, through more advanced machine learning techniques or some mathematically proven parameter tuning methods, more complex and high-accuracy models can be built, such as combining convolutional neural networks and support vector machines to improve model performance. Correspondingly, the classification model we established based on this grayscale image data set can also be improved to classify and study more problems, such as the recognition and classification of HD and RGB images, the realization of face recognition system and the real-time detection of closed-eye features of fatigued driving, etc.

References

[1] J. Brownlee. How to develop a cnn for mnist handwritten digit classification. https://machinelearningmastery.com/how-to-develop-a-convolutional-neural-network-from-scratch-for-\mnist-handwritten-digit-classification/. April 9, 2022,.

[2] R. Garg. 7 types of classification algorithms. https://analyticsindiamag.com/7-types-classification-algorithms/. April 9, 2022,.

[3] I. T. Jolliffe and J. Cadima. Principal component analysis: a review and recent developments. *Philos Trans A Math Phys Eng*, 374(2065):20150202, 2016.

[4] A. Singh. Feature engineering for images: A valuable introduction to the hog feature descriptor. https://analyticsindiamag.com/7-types-classification-algorithms/. April 9, 2022,.

[5] D. K. Srivastava and L. Bhambhu. Data classification using support vector machine. *Journal of Theoretical Applied Information Technology*, 12(1):243–248, 2010.

A Appendix

How to run the code

Our code steps are divided into 7 steps. The first step is to import data and run it in order from top to bottom. The second step is to Pre-processing data. It needs to be done according to HOG,Normalization, PCA. The third step is the algorithms on default settings. The fourth step is Fine-tune hyper-parameters. You need to run the data visualization module first. Then each algorithm has three parts. The first part is GridSearchCV, and the second part is the optimal parameter fitting and prediction for each algorithm. For the convenience of detection, you can only run this part. The last part is data visualization. The fifth step is Classifier comparisons. The running time is compared. The sixth step is hardware and software specifications. The seventh step is to export Prediction. The default we set is SVM model: test_prepared_best__SVM.

Hardware and software environments

April 9, 2022

[1]:
```
!cat /proc/cpuinfo
```

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2199.998
cache size      : 56320 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 0
initial apicid
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni
pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx
f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb
stibp fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx
smap xsaveopt arat md_clear arch_capabilities
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds
swapgs taa
bogomips        : 4399.99
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:

processor       : 1
vendor_id       : GenuineIntel
```

```
cpu family      : 6
model           : 79
model name      : Intel(R) Xeon(R) CPU @ 2.20GHz
stepping        : 0
microcode       : 0x1
cpu MHz         : 2199.998
cache size      : 56320 KB
physical id     : 0
siblings        : 2
core id         : 0
cpu cores       : 1
apicid          : 1
initial apicid  : 1
fpu             : yes
fpu_exception   : yes
cpuid level     : 13
wp              : yes
flags           : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov
pat pse36 clflush mmx fxsr sse sse2 ss ht syscall nx pdpe1gb rdtscp lm
constant_tsc rep_good nopl xtopology nonstop_tsc cpuid tsc_known_freq pni
pclmulqdq ssse3 fma cx16 pcid sse4_1 sse4_2 x2apic movbe popcnt aes xsave avx
f16c rdrand hypervisor lahf_lm abm 3dnowprefetch invpcid_single ssbd ibrs ibpb
stibp fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm rdseed adx
smap xsaveopt arat md_clear arch_capabilities
bugs            : cpu_meltdown spectre_v1 spectre_v2 spec_store_bypass l1tf mds
swapgs taa
bogomips        : 4399.99
clflush size    : 64
cache_alignment : 64
address sizes   : 46 bits physical, 48 bits virtual
power management:
```

[2]:
```
!df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
overlay         108G   40G   69G  37% /
tmpfs            64M     0   64M   0% /dev
shm             5.8G     0  5.8G   0% /dev/shm
/dev/root       2.0G  1.2G  817M  59% /sbin/docker-init
tmpfs           6.4G   32K  6.4G   1% /var/colab
/dev/sda1        81G   44G   38G  54% /etc/hosts
tmpfs           6.4G     0  6.4G   0% /proc/acpi
tmpfs           6.4G     0  6.4G   0% /proc/scsi
tmpfs           6.4G     0  6.4G   0% /sys/firmware
```

[3]:
```
!cat /proc/meminfo
```

```
MemTotal:       13302920 kB
MemFree:        10840032 kB
MemAvailable:   12468904 kB
Buffers:          105488 kB
Cached:          1682152 kB
SwapCached:            0 kB
Active:           980584 kB
Inactive:        1292412 kB
Active(anon):     443796 kB
Inactive(anon):      468 kB
Active(file):     536788 kB
Inactive(file):  1291944 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:             0 kB
SwapFree:              0 kB
Dirty:              7808 kB
Writeback:             0 kB
AnonPages:        485332 kB
Mapped:           226308 kB
Shmem:              1176 kB
KReclaimable:      83232 kB
Slab:             123912 kB
SReclaimable:      83232 kB
SUnreclaim:        40680 kB
KernelStack:        4976 kB
PageTables:         6916 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     6651460 kB
Committed_AS:    3013052 kB
VmallocTotal:   34359738367 kB
VmallocUsed:        7312 kB
VmallocChunk:          0 kB
Percpu:             1448 kB
AnonHugePages:      2048 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

```
Hugetlb:               0 kB
DirectMap4k:       86848 kB
DirectMap2M:     5152768 kB
DirectMap1G:    10485760 kB
```