DAT410/DIT728 Design of AI systems
Module 3

Assignment 3: Group 54

February 6, 2024

Group members:

Member 1:
Name: Hongguang Chen
Personal number: 19961019-5795
Program: MPHPC
Uni email: chenhon@chalmers.se
Time spent: 16 hrs

Member 2:
Name: Sanjaya Thilak Bandara Asurasinghe Rajamantrilage
Personal number: 19890711-4733
Program: MPDSC
email: arstbandara@gmail.com
Uni email: sanraj@chalmers.se
Time spent: 16 hrs

The declaration:
"We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part of other solutions."

1 Reading and reflection

The paper delves into the challenges posed by technical debt within machine learning (ML) systems, which arises when software development prioritizes speed over long-term sustainability. Unlike traditional coding issues, this debt often hides within the system itself, exacerbated by the complex interaction between data and ML behavior.

ML models bring added complexity due to factors such as entanglement, where inputs are not truly independent. Strategies like isolating models or focusing on detecting prediction behavior changes can help mitigate this. Correction cascades, used to rectify model outputs, introduce further complexity and long-term costs despite short-term cost savings. Undeclared consumers, unexpected elements within the system, also contribute to complications. Overall, addressing these factors is crucial for managing the complexities introduced by ML models effectively.

Data dependencies in ML systems present a significant challenge as they carry higher costs than code dependencies, yet lack readily available tools for identification. Unstable or underutilized data

dependencies require novel detection methods, while feedback loops and common anti-patterns further compound the technical debt problem.

Configuration debt, stemming from the multitude of adjustable settings within ML systems, alongside the need to adapt to external changes, adds to the complexity. Fixed thresholds may become outdated in dynamic systems, requiring vigilant monitoring and testing to ensure continued effectiveness.

The paper also touches upon other areas of ML-related debt, such as data testing and reproducibility. It emphasizes the importance of proactive strategies for managing technical debt effectively, suggesting that re-evaluating development practices and fostering a culture of accountability are essential steps toward mitigating its impact.

In essence, the paper underscores the multifaceted nature of technical debt in ML systems and highlights the necessity of adopting proactive measures to address and manage it effectively.

2 Implementation

This task involves classifying whether a given day has high PM2.5 levels based on 10-dimensional data. While various methods can tackle this problem, we've been instructed to use k-Means as our primary method. However, k-Means is an unsupervised method, meaning it divides the data solely based on minimizing loss. When we aim to classify into "high PM2.5" (labeled as 1) and "not high PM2.5" (labeled as 0), using only 2 clusters may not yield optimal results. Additionally, as there could be various factors contributing to high PM2.5 levels, the number of clusters becomes a crucial parameter that requires thorough testing.

2.1 About the classifier

We implement the "fit-predict-score" pattern in our classifier class, utilizing a k-means model as the kernel (referenced as the 'KMeansClassifier' class in the code file).

1. Initially, we train the classifier using data from Beijing and Shenyang, with these two datasets combined into the training data within the code. Additionally, we randomly select 20% of the training data for validation purposes.

2. During the training loop, we utilize the default value of 300 iterations.

3. Following training, each centroid is assigned a label corresponding to the majority label of points within its cluster.

4. During inference, we first determine the cluster assignment of the new point using the trained k-means centroids. Subsequently, we predict the label associated with the centroid of the inferred cluster.

5. For scoring, we compare the predicted results with the ground truth labels.

2.2 About the dataset

Before delving into the results, we conduct a thorough statistical analysis of the dataset (see Figure 1). This analysis reveals clear distinctions between the distribution patterns of the training data (derived from Beijing and Shenyang) and the evaluation data (collected from Guangzhou and Shanghai).

2.3 Result

Here are the results of our experimentation with varying cluster numbers, ranging from 2 to 50, as depicted in Figure-2. Notably, with a small number of clusters (2-4), performance appears to excel,

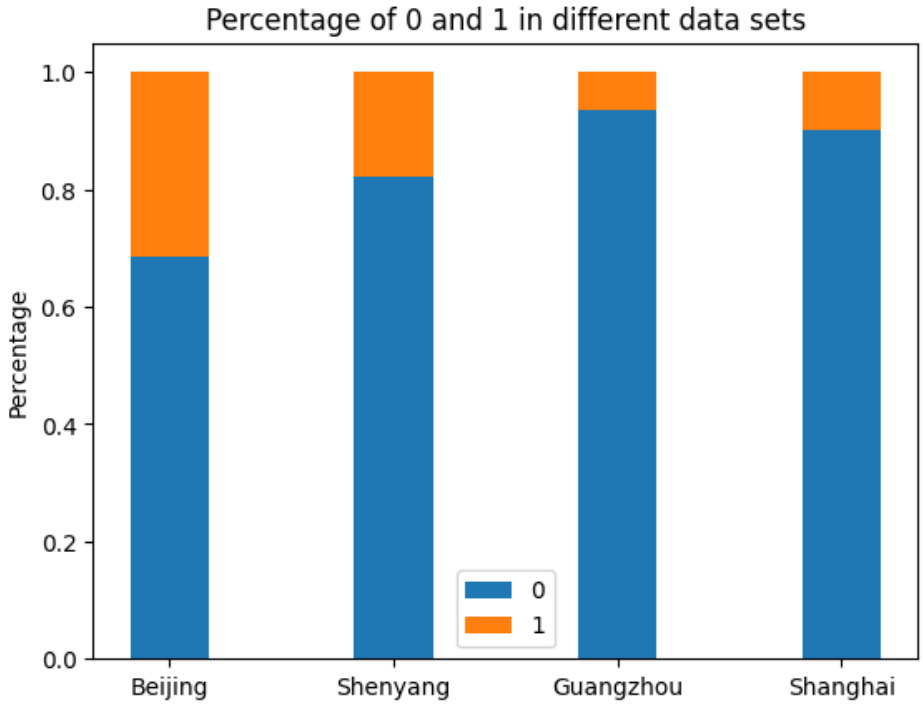

Figure 1: Percentage vs Cities

achieving up to 90% accuracy on the evaluation set. As the cluster number increases (5-8)(accuracy is about 40-60%), performance initially declines before recovering (8-50), eventually stabilizing at a relatively consistent level.

As a result, we conducted a more detailed analysis of the accuracy for each label (0 and 1) separately, as depicted in Figure-3. Interestingly, we observed that the accuracy for label 1 is significantly lower compared to label 0, both in the training and evaluation datasets. This suggests that the model consistently leans towards predicting label 0, resulting in a high overall accuracy on the simulation set (as indicated by the distribution of labels 0 and 1 in Figure-1).

Given the limitations of K-Means for prediction, we explored alternative methods such as Support Vector Machines (SVM) and a fully connected neural network consisting of 5 layers. Remarkably, both models yielded comparable results, as summarized in Table-1. Further details regarding the implementation of both models can be found in the accompanying code.

Table 1: Model Performance

Method	**Training acc**	**Validation acc**	**Evaluation acc (Guangzhou)**	**Evaluation acc (Shanghai)**
FC(5)	0.728411	0.711572	0.936391	0.901554
SVM	0.805699	0.789292	0.780325	0.743893

3 Discussion

Given the way our data is distributed and the conditions we're dealing with, our method doesn't seem to work as well as we'd like. Surprisingly, our system actually shows higher accuracy during the

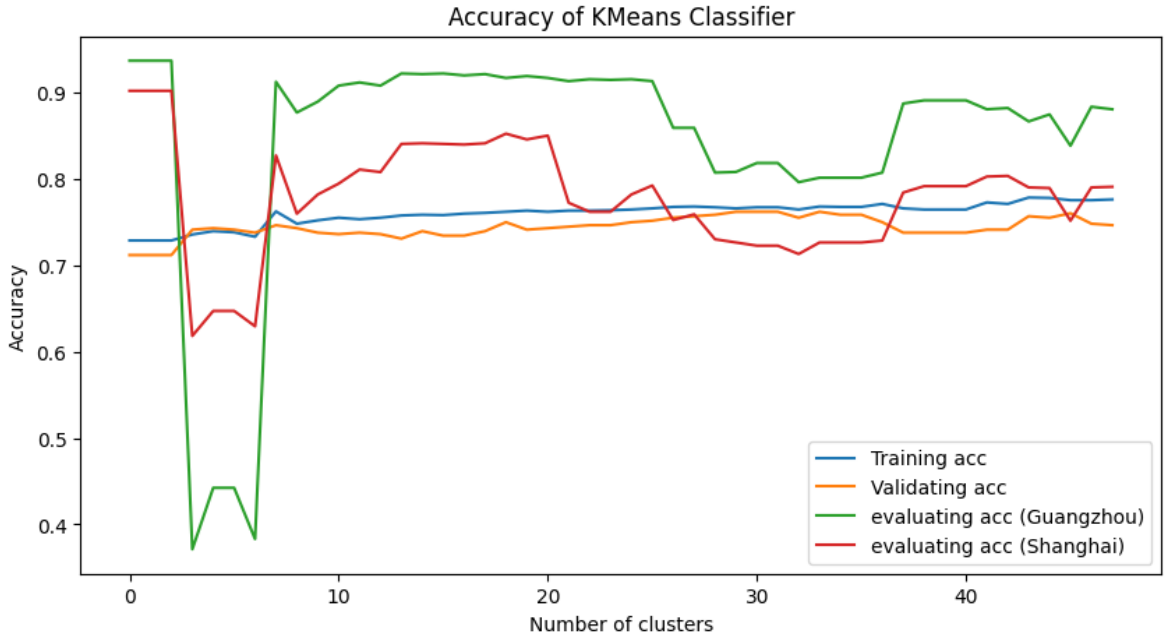

Figure 2: Number of clusters vs Accuracy

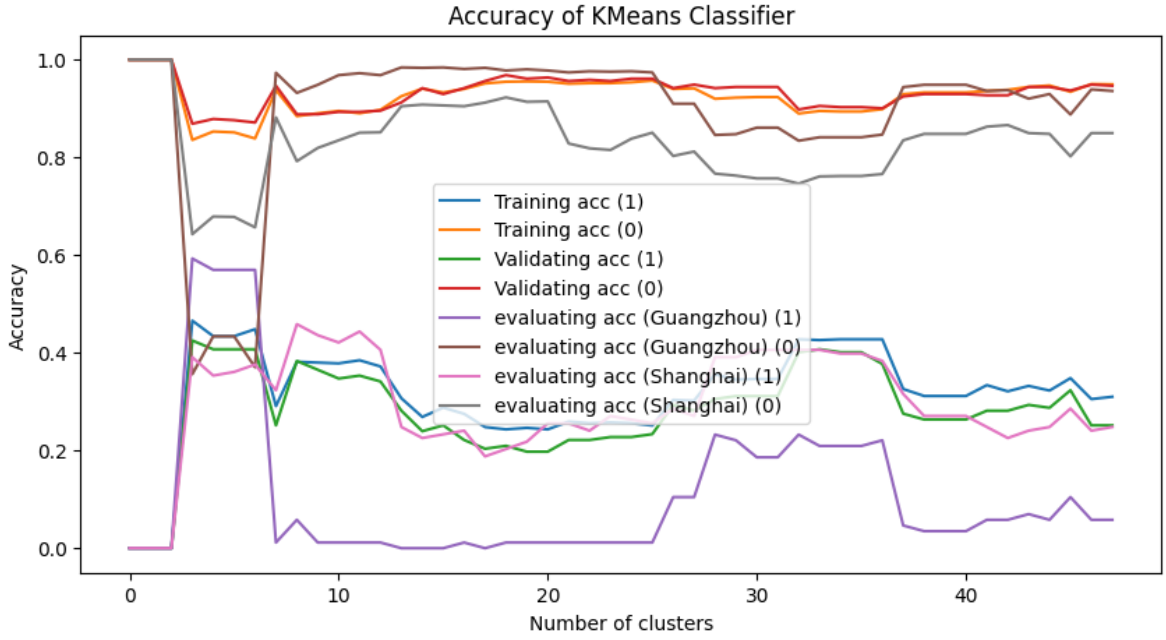

Figure 3: Enter Caption

evaluation stage than it does during training, which goes against the usual trend.

To try and improve accuracy, we could look into removing unnecessary columns from our data. This might involve some research to figure out which columns aren't helpful, or even make things worse. Another idea is to balance how we select data labeled as 0 and 1 when training the model.

These changes need more testing and experimenting to see if they make our system work better. But it's important to remember that these ideas might not work the same for every problem, so we need to adjust our approach based on the specific characteristics of our data and what we're trying to do.