Home Assignment 3 – Semi-Supervised Learning

3945 – Advanced Machine Learning, Sem A 2024

Timor Baruch —— Hadar Pur

Feb 25, 2024

1 Dataset Generation

1.1 Conditionally Independent Dataset - Bonus

To describe the data generation process for the semi-supervised learning task, we utilize a synthetic dataset creation method. The process involves generating two views of the data, each containing information relevant to the classification task. Here's a detailed description of the process.

1.1.1 Initialization

In the initialization phase, we set the parameters for our dataset, specifying the number of samples as 1000 and the number of features as 10. Following this, we construct covariance matrices to capture the variability and relationships among features. The first covariance matrix, named covariance1, serves to denote no correlation between features, being an identity matrix. Conversely, covariance2, also an identity matrix, scales its diagonal elements by a factor of 2, indicating higher variability.

Next, we establish mean vectors, mean1 and mean2, representing the central or average values for each feature. Mean1 is initialized with zeros, suggesting low average values across all features. In contrast, mean2 is set to 0.7 for all features, reflecting a shift towards higher values compared to mean1.

1.1.2 Data Generation

The dataset is then split into two halves (5 features each) to accommodate binary classification, allocating an equal number of samples to each class. Data generation involves creating samples for each class (0 and 1) in both views using multivariate normal distributions. For class 0, samples are generated with mean1 and covariance1, signifying low variability centered around zero. Conversely, for class 1, samples are generated with mean2 and covariance2, representing higher variability centered around 0.7.

Following data generation, samples for each class in both views are concatenated, forming the complete datasets, view1 and view2. Subsequently, labels are assigned based on class membership, with the first half of samples in the concatenated datasets assigned to class 0 and the second half to class 1.

Finally, the dataset undergoes shuffling to randomize the order of samples within each class, ensuring uniform distribution and eliminating potential biases in subsequent analyses or model training. This data generation

process results in two views of the dataset, each containing samples with distinct features and class labels, while keeping the two sets of variables independent given the class label.

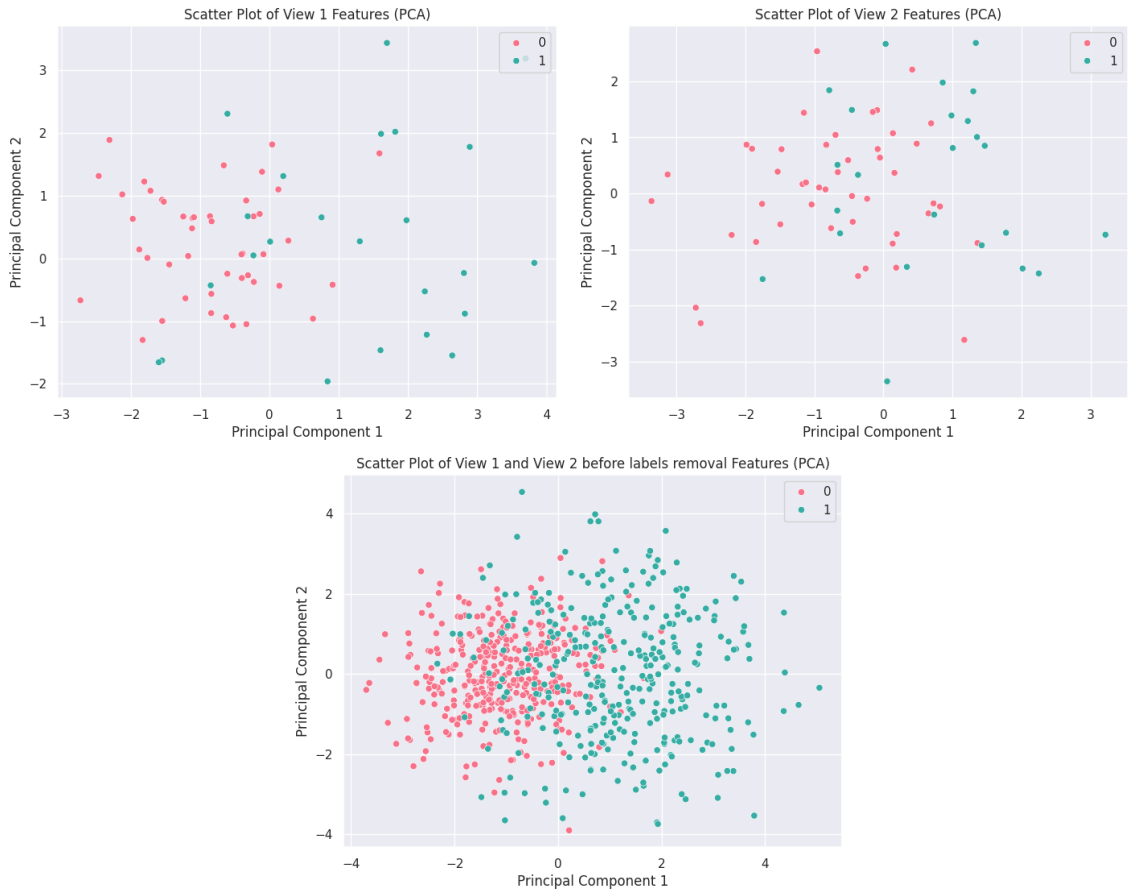


Figure 1: Generated model visualizations

1.1.3 Conditional Independence Across Views in Dataset Generation

Conditional independence across views given the class label is ensured by:

1. **Class-Conditional Distribution:** For each class, samples are drawn from its own multivariate normal distribution, with unique means and covariances defining each distribution. This approach captures both the common characteristics within each class and the distinguishing features between classes. Consequently, the feature distribution across views within each class is entirely determined by these class-specific parameters.

2. **Independent Views:** A crucial element of this data generation method is the autonomy of views for each class. Features in one view are generated without being affected by or related to those in another view. This ensures that understanding the distribution of one view based on the class label does not provide any additional information about the other view, confirming their independence.

This engineered conditional independence is highly beneficial for multi-view learning algorithms. It enables

these algorithms to utilize information from different views separately, improving their accuracy and reliability in classification tasks. Please note that in the notebook we made two statistic exams for determine weather or not the dataset has conditionally independent two sets of variables (the two views) given the class labels, as required.

1.2 Provided Dataset

We also utilized the dataset provided by the class instructor, which has similar characteristics as the previous dataset. It comprises 1000 samples labeled with binary classes 0 or 1, and each view contains 5 features. Notably, the features are presumed to be conditionally independent given the class label.

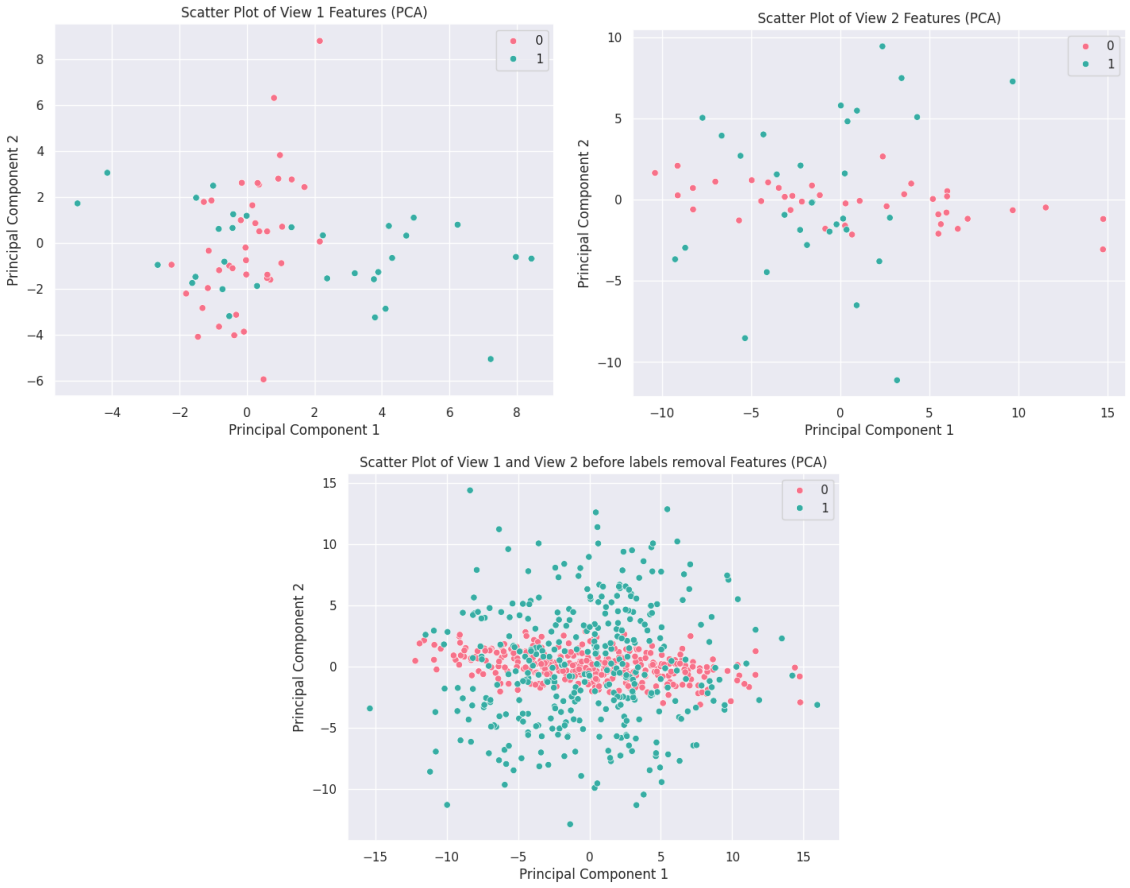


Figure 2: Provided model visualizations

1.3 Data Pre-Processing

In this step, we divided the dataset into training, validation, and test sets where 70% of the dataset is used for training, 15% is for testing and 15% is for validation. Then, we removed 90% of the labels from the training set by randomly selecting a subset of labels and marking them as unlabeled (-1). This process aimed to simulate a semi-supervised learning scenario where a large portion of the training data lacks class labels.

2 Co-training Implementation

2.1 Our Co-training Algorithm

Co-Training is a semi-supervised learning technique that leverages the information contained in multiple views of the data to improve classification performance, particularly in scenarios where labeled data is scarce.

We implemented the Co-Training algorithm inspired by the article by Blum & Mitchell [1]. The implementation follows the pseudo code outlined below:

Algorithm 1 Co-training in Pseudo-Code from the article

Input: L - set of labeled training examples, U - set of unlabeled examples

Create a pool U' of examples by choosing u examples at random from U

for k iterations **do**
    Use I to train a classifier h_1 that considers only the x_1 portion of x
    Use I to train a classifier h_2 that considers only the x_2 portion of x
    Allow h_1 to label p positive and n negative examples from U'
    Allow h_2 to label p positive and n negative examples from U'
    Add these self-labeled examples to L
    Randomly choose $2p + 2n$ examples from U to replenish U'
end for

The Co-Training process involves several iterative steps, which are outlined below:

1. **Initialization**: Two base classifiers, denoted as $model_1$ and $model_2$, are initialized using labeled data available initially.

2. **Iteration**:

    - **Training**: We trained two base classifiers, using the labeled data available initially. This step involved fitting the classifiers to the labeled data, allowing them to learn the underlying patterns and relationships between the features and labels.

    - **Pool of Unlabeled Data**: Initially, a pool of unlabeled data, was created by randomly selecting a subset of $pool\_size$ unlabeled examples from the dataset (similar to U' in the article). After pseudo-labeling in each iteration, this pool was reoccupied by randomly choosing samples from the remaining unlabeled dataset.

    - **Prediction**: After training the classifiers, we generated predictions for the unlabeled data points in the pool of unlabeled samples, using both $model_1$ and $model_2$. This step involved using the trained classifiers to predict the class labels for the unlabeled samples, even though their true labels were unknown.

- **Pseudo-Labeling**: In this step, we pseudo-labeled the unlabeled data points based on the classifiers' predictions. We utilized a parameter called top_k (which serves the same purpose as the p and n parameters in the article) to determine how many of the highest probability samples were chosen for labeling in each iteration of the co-training loop. Samples with high confidence (exceeding a predefined threshold denoted by $confidence\_level$ parameter) were selected to be added to the labeled dataset.

- **Update**: After pseudo-labeling the unlabeled data points, we augmented the labeled dataset with the newly pseudo-labeled samples. These samples were then removed from the pool of unlabeled data to prevent duplication and ensure the integrity of the training process.

- **Retraining**: Finally, we retrained both classifiers using the updated labeled dataset, which now included the newly pseudo-labeled samples. This step involved fine-tuning the classifiers based on the augmented labeled data to further improve their performance and adapt to the newly acquired information.

3. **Our Model's Parameters:** Our model's parameters consist of several key components. Firstly, we have $model1$, which represents the first classifier utilized within the co-training algorithm, and $model2$, which denotes the second classifier employed in the same algorithm. Additionally, there are optional parameters such as $num\_iterations$, specifying the number of iterations with a default value of 100, and $pool\_size$, indicating the size of the pool of unlabeled samples with a default of 70. Another parameter is the $confidence\_level$, determining the confidence level for label assignment, set to a default of 0.65. Lastly, the top_k parameter dictates the number of highest probability samples chosen for labeling in each iteration of the co-training loop, with a default value of 4. Throughout our experiments with both datasets, we choose these default parameters because they achieved the best classification results, while trying different models, as detailed in subsequent sections.

4. **Termination Condition**: The iterative process continues until reaching $max\_iterations$ or until all the unlabeled data is being labeled. We allow for maximum of $pool\_size$ labels to be unlabeled (due to not exceeding the threshold number). We wanted to classify only the most certain samples. Reaching a confidence level below 0.65 (default value for $confidence\_level$) implies that the model is uncertain about how to classify a particular sample and it's not a definitive classification.

5. **Evaluation**: We evaluate each co-training iteration on the validation dataset.

The Co-Training process capitalizes on the diversity of information provided by multiple views of the data and iteratively refines the classifiers' predictions by incorporating pseudo-labeled data. This approach effectively utilizes both labeled and unlabeled data to enhance classification accuracy, making it particularly useful in scenarios where obtaining labeled data is challenging or expensive.

2.1.1 Training on Generated Dataset

In order to train our co-training classifier on the generated dataset, we chose the Naive Bayes classifier (for both models) after experimenting with various classifiers because of its suitability for conditionally independent datasets. We used default parameters as provided in section 3. The results are presented below.

	Precision	Recall	F1-score	Support
0	0.83	0.93	0.88	76
1	0.92	0.80	0.86	74
Accuracy			0.87	150
Macro Avg	0.87	0.87	0.87	150
Weighted Avg	0.87	0.87	0.87	150

Table 1: Classification Report on Test set (Generated Dataset)

Figure 3: Validation Evaluation Matrices on the Generated Dataset

2.1.2 Training on Provided Dataset

In order to train our co-training classifier on the provided dataset, we chose the Random Forest classifier as *model*1 and the SVM classifier as *model*2 after experimenting with various classifiers and achieving best results with this combination. We wanted to combine two different classifiers to observe how our co-training algorithm handles two distinct models, aiming to create a more robust model while hoping to achieve better classification results. We used default parameters as provided in section 3. The results are presented below.

	Precision	Recall	F1-score	Support
0	0.89	0.97	0.93	79
1	0.97	0.86	0.91	71
Accuracy			0.92	150
Macro Avg	0.92	0.92	0.92	150
Weighted Avg	0.92	0.92	0.92	150

Table 2: Classification Report on Test set (Provided Dataset)

Figure 4: Validation Evaluation Matrices on the Provided Dataset

3 Evaluation & Comparisons

3.0.1 Generated Dataset

We assessed the performance of our co-training approach by removing 90% of the labels. We utilized Naive Bayes classifiers as our models. This was compared against the fully labeled dataset trained with the Naive Bayes classifier. Below are the results of this comparison.

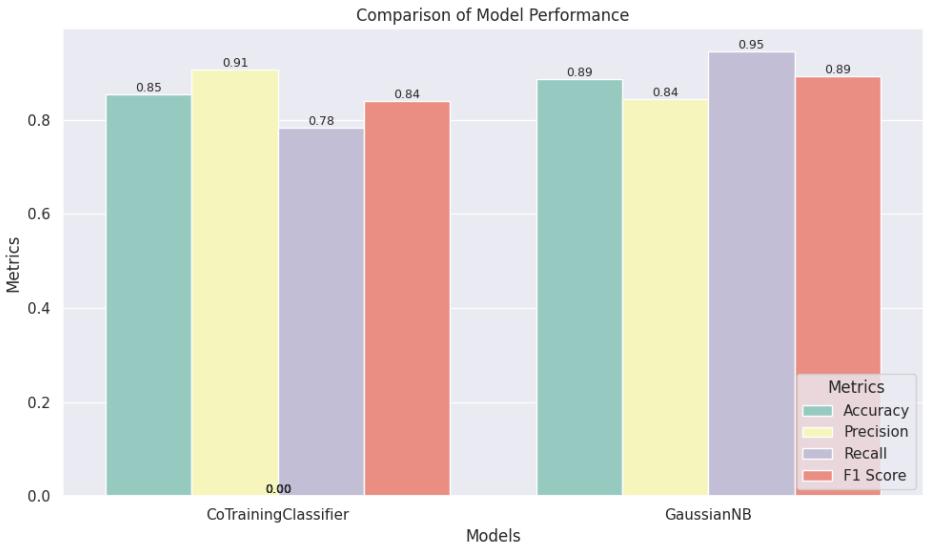
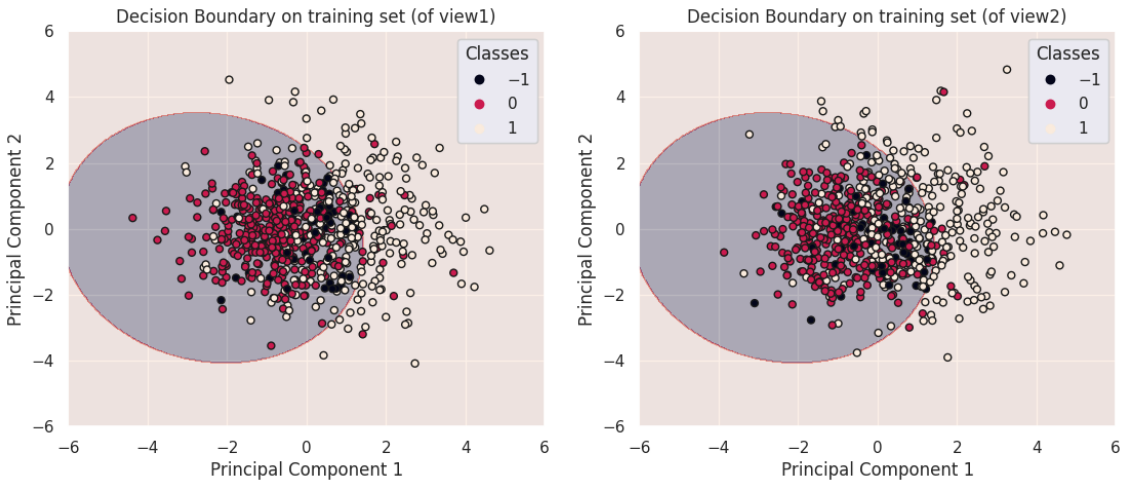


Figure 5: Comparison with Fully Labeled Model (Naive Bayes Classifier)

We observed that our model, even when trained with only 10% of the labels, achieved competitive results compared to models trained on fully labeled data. Specifically, our model attained an accuracy of 85% on the test data, whereas the fully labeled model achieved 89% accuracy.

We also examined the decision boundaries of our co-training classifier on the test and train sets. We observe that the algorithm handles the data well despite the absence of 90% of the labels. It can still distinguish effectively between the different classes, which is impressive given that the features are conditionally independent given the labels.



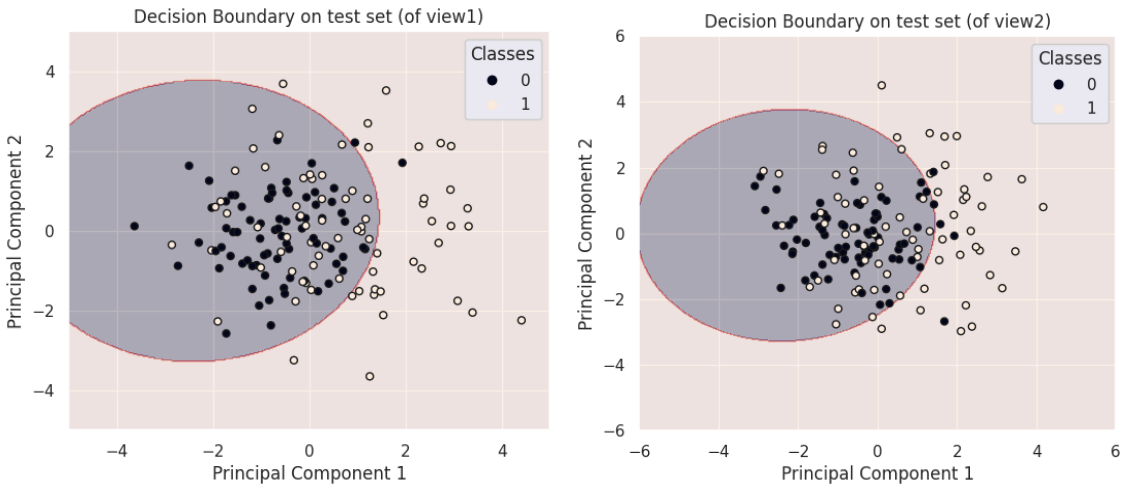


Figure 6: Decision Boundaries on Generated Dataset

3.0.2 Provided Dataset

We assessed the effectiveness of our co-training approach by removing 90% of the labels. We utilized Random Forest as our first model and SVM classifier as our second model. We compared this against models trained with the fully labeled dataset. The results of this comparison are outlined below.

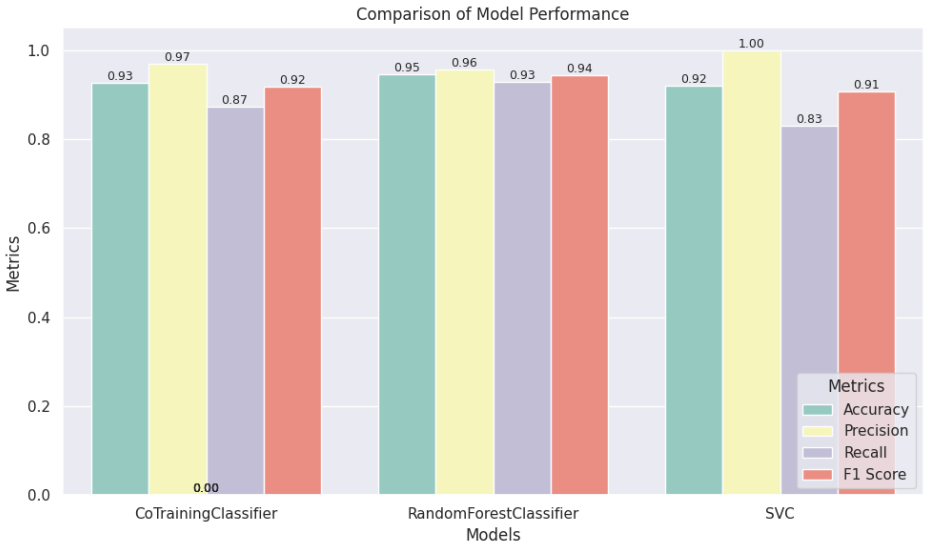


Figure 7: Comparison with Fully Labeled Models (RF & SVM classifiers)

We observed that our model, even when trained with only 10% of the labels, achieved competitive results compared to models trained on fully labeled data. Specifically, our model attained an accuracy of 93% on the test data, whereas the fully labeled SVM achieved 92% accuracy and the fully labeled RF achieved 95% accuracy.

In the provided dataset, similar to the generated dataset, we examined the decision boundaries created by our co-training algorithm. We observed well-defined contours of the different classes, indicating the effectiveness of the algorithm in classification. These results were particularly surprising considering that our co-training classifier combines two different models to create the decision boundaries.

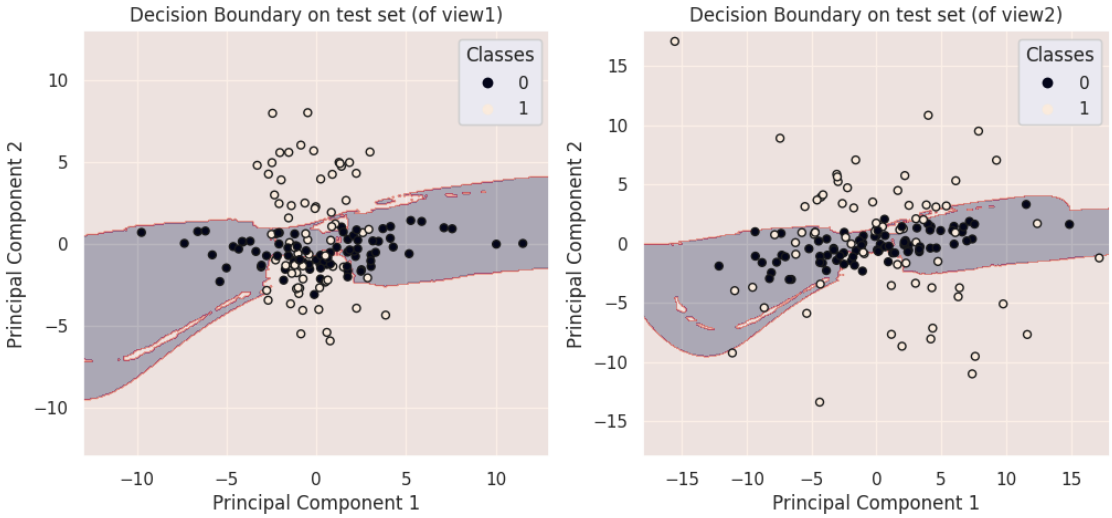


Figure 8: Decision Boundaries on Provided Dataset

4 Visualization - Bonus

In this section, we created an animation to visualize how our Co-Training classifier learns from the data over time. We used it to visualize the process on the generated dataset. Here's a simplified explanation:

1. **Update Function**: We defined a function called `update` that updates the animation frame by frame. This function adjusts the decision boundaries of our classifier and plots them on two separate views of the data.

2. **Animation Creation**: We used Matplotlib's `FuncAnimation` class to create the animation. It iterates through different frames, with each frame representing a step in the Co-Training process. The `update` function is called for each frame to update the plot accordingly.

The animation is shown in the notebook in the section **Decision Boundaries & Animation** and the video is also attached to the zip.

5 Summary

In this project, we explored the application of Co-Training, a semi-supervised learning technique, in the context of binary classification tasks. We began by generating a synthetic dataset with conditionally independent features, simulating a scenario where labeled data is limited. The dataset consisted of two views,

each containing distinct feature sets, and binary class labels of 0/1. Through careful data generation, we ensured that the features across views remained conditionally independent given the class label. We also used the provided dataset' which is also a binary dataset 0/1 consists of conditionally independent variables given the labels. we removed 90% of the labels for the co-training process.

We then implemented the Co-Training algorithm, following the framework outlined in the work by Blum & Mitchell [1]. Our implementation involved iteratively training two base classifiers, pseudo-labeling unlabeled data points, and updating the classifiers using the augmented labeled dataset. We introduced parameters such as the number of iterations, pool size, confidence level, and top-k for fine-tuning the algorithm's behavior.

Our experiments involved training the Co-Training algorithm on both the generated dataset and a provided dataset. For the generated dataset, we employed Naive Bayes classifiers as our models and observed competitive performance even with 90% of the labels removed. On the provided dataset, we utilized Random Forest and SVM classifiers, achieving comparable results to models trained on fully labeled data.

Notably, our Co-Training approach demonstrated robustness and effectiveness in handling datasets with limited labeled data. Even when compared to models trained on fully labeled datasets, our approach achieved very similar accuracy, precision, and recall results on the validation data. By leveraging two different views of the data and iteratively refining classifiers, we were able to achieve competitive classification accuracy, highlighting the potential of semi-supervised learning techniques in real-world scenarios where labeled data is scarce or expensive to obtain. Additionally, our visualization of decision boundaries showcased the algorithm's ability to create well-defined contours of different classes, further validating its efficacy in classification tasks.

Bibliography

[1] Avrim Blum and Tom Mitchell. "Combining Labeled and Unlabeled Data with Co-Training". In: *Proceedings of the Annual ACM Conference on Computational Learning Theory* (Oct. 2000). DOI: 10.1145/279943.279962.