Assignment 2 Report of Machine Learning SVM Models and Clustering Methods

Prepared by: Ghadeer Sami Nafadi

ID: 44511798

Group: 2

Loading and Prerocessing:

First I loaded (Breast Cancer Dataset), from UCI repository using **ucimlrepo** library. The dataset was already clean there where no missing values or strange columns.

After loading, I used StandardScaler to scale features. This step is importantnbecause SVM models are sensitive to feature ranges, Then I split the data into training and testing sets using 80/20 ratio.

:Training SVM models

I trained three different SVM models using different kernal types: **Linear** kernal**, RBF** kernal**,** and **Polynomial** kernal**.**

Each model was trained on the scaled training data. After training I used **.predict()** to get predictions from each model on the test set.

Evaluating and Confusion Matrix:

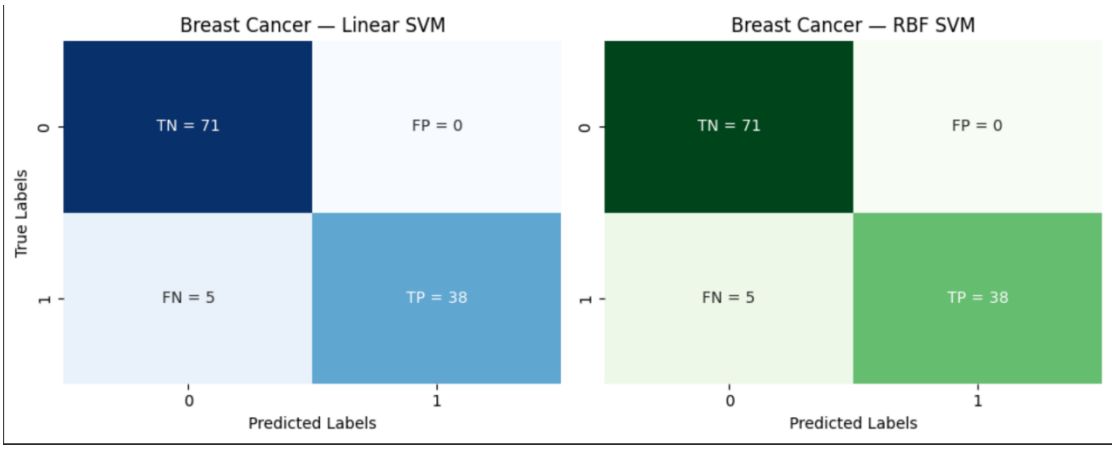
The Results:

Model	Accuracy	Recall	Precision	F1 Score
Linear SVM	95.6%	88.4%	100%	0.938%
RBF SVM	95.6%	88.4%	100%	0.938%
Poly SVM	88.6%	69.8%	100%	0.822%

So I noticed that both **Linear** and **RBF** SVM gave exactly the same results, which shows the data is linearly separable.

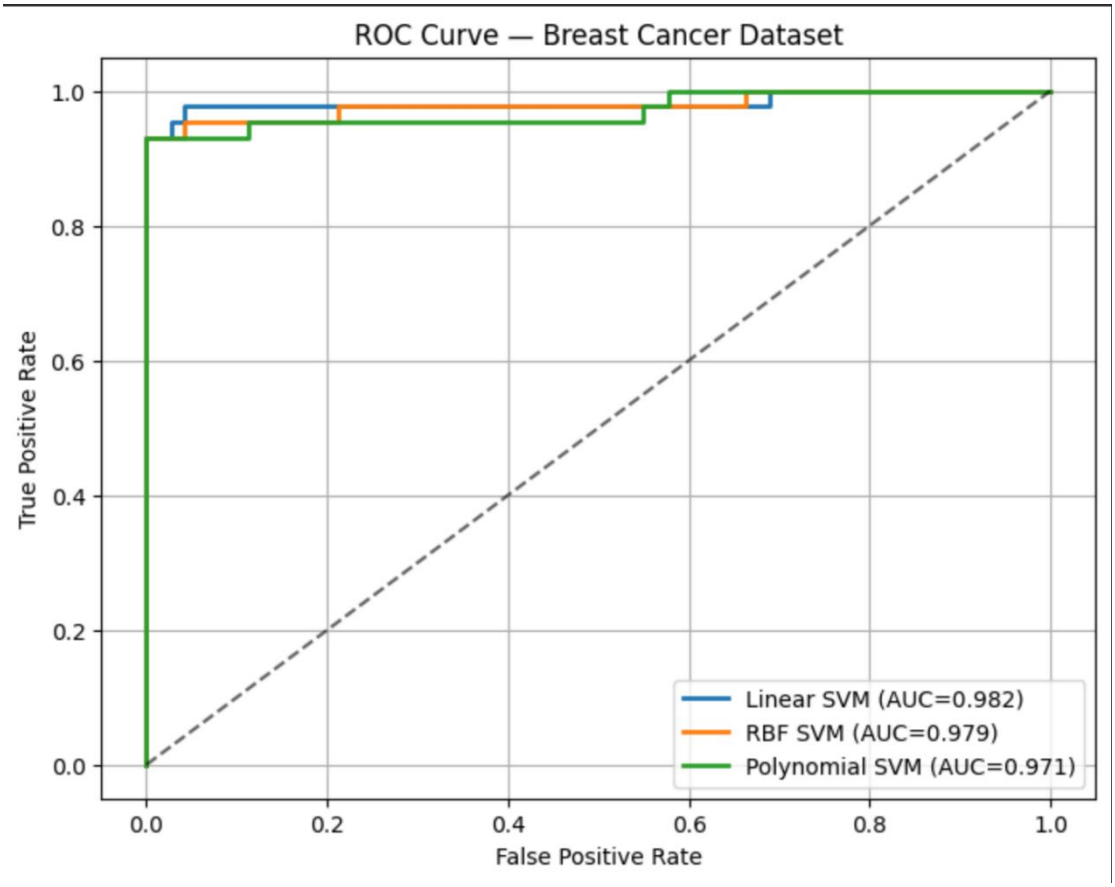
Rether than **Polynomial** SVM was weaker, especially in recall it missed more **Malignant** cases(FN).

Confusion Matrix for (Linear & :RBF)



That means the model wasn't make any false positive(**FP**) erorrs, which is great for medical data. But it did miss 5 malignant cases, which is something to be careful about.

:ROC Curve & AUC



All models performed well, but **Linear SVM** had the highest AUC2. Cause in this dataset(Breast cancer) is small, clean, and linearly separable. This is means the tow classes (Malignant, Benign) can be split with a straight boundary in feature space.

More complex kernnal like RBF and Polynomial didn't improve performance, and in polynomial case even reduced recall, This shows that simprel model can outperform more complex ones when data structure is straightforward.

Hyperparameter Tuning:

After finishing from initial SVM models training, I manually tuned it to see if I coud improve their performance. I focused on the three kernall types and tested different values for **C, gamma, and degree.**

-For **Linear** I only tuned C=[0.1, 1, 10]

 reason is that C controls how strict the model is with misclassifications, Smaller C makes the model simpler and more flexible, and Larger C forces the model to fit the training data more tightly

About Testing these three values was good to understand how much sensitive the dataset is to C without overcomplicating things.

-For **RBF** I tuned combinations of C = [0.1, 1, 10] with .gamma = [‘scale’, ‘auto’]

Gamma affects how far each point influences the decision boundary: which is high gamma makes the model focus too closely on individual points (risk of overfitting), and Lower or automatic gamma values create smoother boundaries

I chose scale and auto because they are commonly used starting points and usually provide balanced .results without making the model overly complex

-For Polynomial I also tuned C = [0.1, 1, 10] and .[4 ,3 ,2] degrees

I avoided very high degrees because they tend to increase model complexity a lot and make training .slower, often without improving accuracy

Degrees 2,4 are usually the reasonable range that helps reveal whether the data has nonlinear patterns, .while still keeping the model stable

:Unsupervised learning -K Means

After working with SVM, I wanted to try something different—unsupervised learning. I chose K-Means clustering to see if the model could separate the malignant and benign cases without using the actual labels. Since the dataset has two classes, I set n_clusters = 2.

And about the scaled features I used the (X_s) to make sure all variables on the same level. K-Means is sensitive to scale, so this step was important.

One thing I had to be careful about is that K-Means doesn't know which cluster is "malignant" and which is "benign"—it just assigns 0 and 1 randomly. So I calculated accuracy twice: once with the original cluster labels, and once with the flipped version. Then I kept the better one.

About the results I was surprised by how well KMeans performed! Even though it didn’t use the labels at all, it still managed to separate the two groups with over 91% accuracy. That tells me the data has a strong natural structure—malignant and benign cases are pretty distinct.

[[344 13]

 [37 175]]

Looking at the confusion matrix, most benign cases were grouped correctly (244 out of 357), and also the most malignant cases clusterd well (175 out of 212)>

So that means ther are 13 benign cases misclassified as malignant, and 37 malignant cases misclassified as benign.

Actually it's not perfect enough, but for an unsupervised model that's very good.

:Hirarchical Clustering

I applied hierarchical clustering using both single and complete linkage methods to explore how well the Breast Cancer dataset separates without supervision. After adjusting for label flipping, both methods gave the same accuracy of 63.1%, which is significantly lower than K-Means. This suggests that the dataset’s structure does not align well with the distance based grouping used in hierarchical clustering.

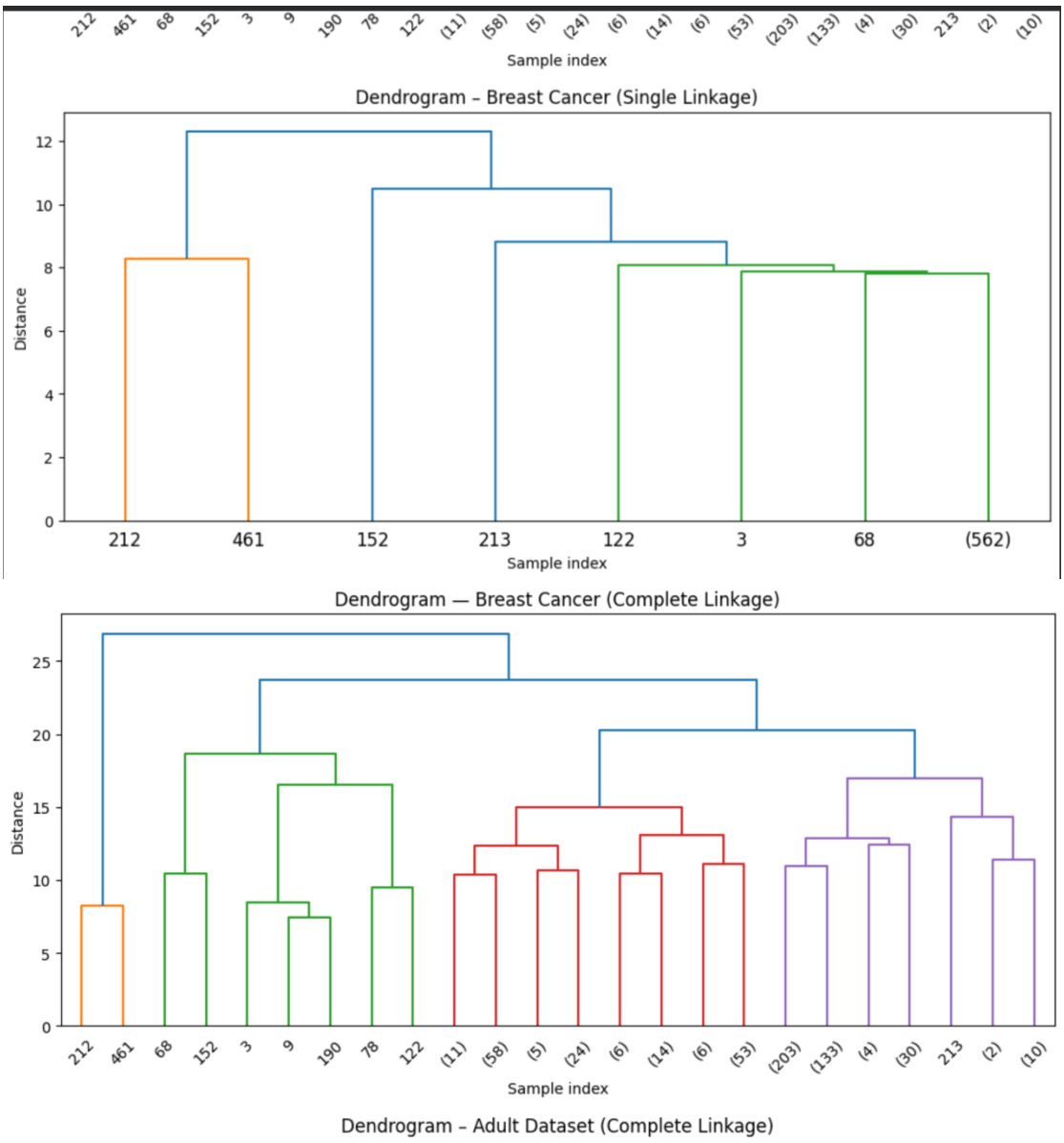
While complete linkage produced a more balanced dendrogram, it did not improve classification performance. Overall, hierarchical clustering was less effective at revealing the true class boundaries in this .case

Dendorgram for Breast Canccer Dataset:

The Cancer dataset is too small which is make hierarchical clustering and plotting fast and manageable, rather than Adult dataset, it has over than 32000 samples, which makes dendrogram plotting extremely slow. It's not just about time, but it can over loading and also can crash, which actually happened to me, my session crashed when I tried running clustering right after tuning. Even if it worked, the plot would be too crowded to interpret. That’s why I limited dendrogram visualization to the smaller dataset, where it was both practical and meaningful.

For this reason I generated a dendrogram for this dataset using complete linkage to visualize how the samples were grouped hierarchically.

Now I'll talk about the dendrogram results:



Here I used Complete Linkege, which connects clusters based on the distance between points, and this method tendes to create more balanced clusters unlike the Single Linkage.

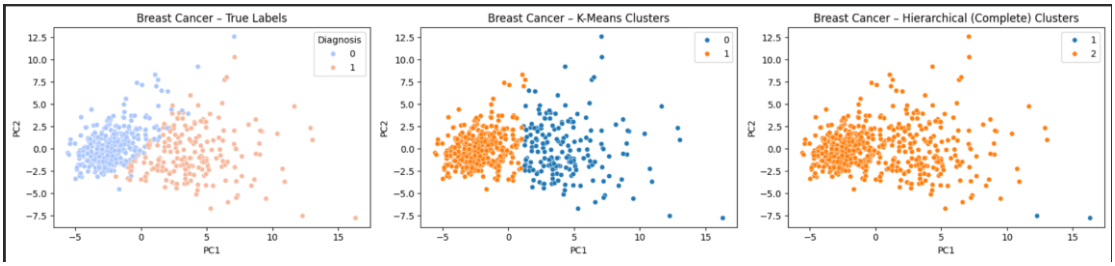
When I used single linkage, I noticed that the tree came out much simpler than the complete linkage and didn't have many branches. The reason is that single linkage merges clusters based only on the closest two points, so merging usually happens quickly and doesn't create deep levels in the tree. The

shape looked like a chain forming step by step without high levels.

As for complete linkage, when I used it, I noticed that the plot became bigger and more filled with branches. This method merges clusters based on the farthest two points, and for that reason, merging doesn't happen until the distance between the two points increases, which makes the branches clearer and more numerous.

Overall, I think complete linkage gives a clearer picture of how samples might actually group together, while single linkage gives a more simplified view but can be overly affected by nearby points.

PCA Visualization:



Here about the PCA or the (Principal Component Analysis) it's giving me better understanding and easy visually how well clustering methods separate the clases, and to decide which method is the better based on reduce the dataset to tow dimensions and shows the clustering results.

The plot of true labels showed that the data is naturally well separated,, which gives a reason for why models like SVM and K Means performed strongly. The K Means clusters closely matched the true labels confirming that the algorithm captured the underlying structure.

In contrast, the hierarchical clustering plot showed more overlap between classes which aligns with its lower accuracy. Overall, PCA helped me visually confirm the strengths and weaknesses of each clustering method

Once I projected the data into 2D I could see that the true labels were naturally well separated. And that made sense especially considering how strong the performance was from models like SVM and K Means.

By looking at the K Means plot the clusters lined up very well with the actual labels.

The boundaries were clear, and there wasn't much overlap which showed that K-Means did a good job capturing the real structure of the data.

On the other hand the hierarchical clustering plot (using complete linkage) looked more mixed. Some points from different classes were grouped close together, and that matched the lower accuracy I saw earlier. It was clear that this method struggled more to separate the two groups.

In the end, PCA didn't just help me reduce the dimension but it gave me a way to see how each clustering method behaved.

Adult Income Dataset:

Now by comparing between the Cancer dataset and the Adult dataset, the Adult dataset is much heavier and more complicated. It has over 32,000 records and 15 columns, mixing numbers with categories. The goal is to predict whether someone earns more than 50K a year, but the path to that answer isn't straightforward.

I started by loading the Adult dataset from the UCI repository using the **ucimlrepo** library.

This dataset is much larger than the Breast Cancer one and contains over 32,000 samples. It's commonly used for income prediction tasks, where the goal is to classify whether a person earns more than 50K or not based on demographic and work-related features. And after that I separated the features and target.

The matadata showed me The dataset includes 15 columns: 14 features like age, education, occupation, hours perweek, and so on.

1 target column: income (binary)

The features are a mix of:

- Integer values (age, capital gain, hours per week)
- Categorical values (workclass, education, marital status)
- Binary values (sex, income)

Preprocessing:

I started by combining the features and target into one dataframe. Then I replaced all ? entries with NAN and dropped any rows with missing values to keep things clean.

Next, I stripped spaces and periods from the income labels and encoded them as binary:

<=50K → 0

>50K → 1

After that, I applied one-hot encoding to all categorical features cause the data includes categorical features like (workclass, education..) and used StandardScaler to scale the entire feature set because the dataset mixes values with very different ranges so without scaling, features like capital-gain would dominate smaller ones like age or binary categories. Also, many models are sensitive to outliers and scale differences since they rely on distance calculations. Scaling ensures all features are on the same level.

Finally, I split the data into training and testing sets using an 80/20 ratio.

Training SVM models:

After preprocessing the dataset, I trained the three Support Vector Machine (SVM) models to classify income levels like before

Each model was trained on the scaled training data (X_train_adult, y_train_adult) and then used to predict income on the test set (X_test_adult).

And there are reasonse about tried different models, cause I didn’t want to assume the data was linearly separable, so that’s way I tested multiple kernels to see which one fits best.

Linear is fast and simple, good for clean separation.

RBF is flexible and handles curves well.

Polynomial can model more complex boundaries, but might overfit if the degree is too high.

By comparing their predictions, I could evaluate which kernel captured the structure of the data most effectively.

Evaluating Models:

After training the three SVM models (Linear, RBF, Polynomial), I evaluated their performance using four key metrics:

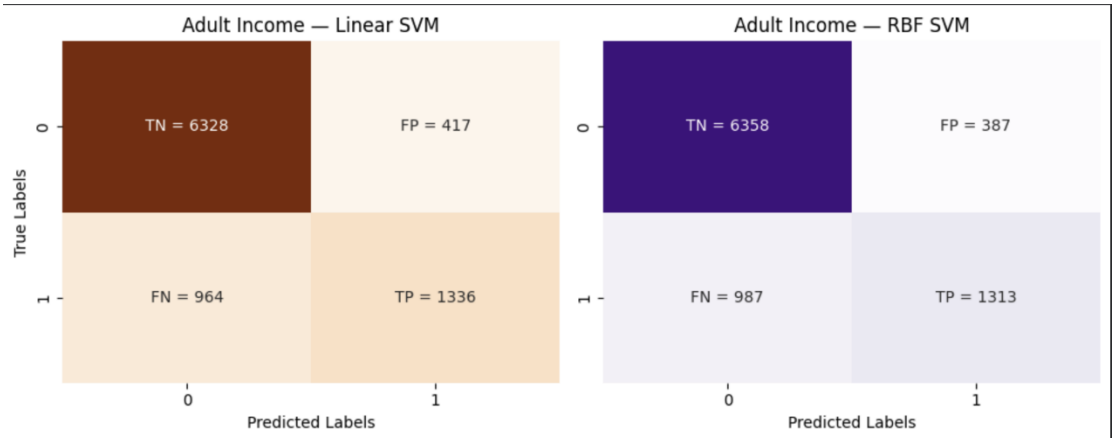
Model	Accuracy	Recall	Precision	F1 Score
Linear SVM	84.73%	58.1%	76.2%	65.9%
RBF SVM	84.81%	57.1%	77.2%	65.7%
Poly SVM	82.679	49.0%	74.6%	59.2%

Linear and RBF SVM gave very similar results. Both had high accuracy and decent precision, but recall was lower, meaning they missed a lot of >50K cases.

Polynomial SVM performed worse overall. It had the lowest recall and F1 score, so this is means it struggled to generalize and may have overfitted.

RBF kernel had slightly better precision than the linear one, but the difference was small.

Confusion Matrix:

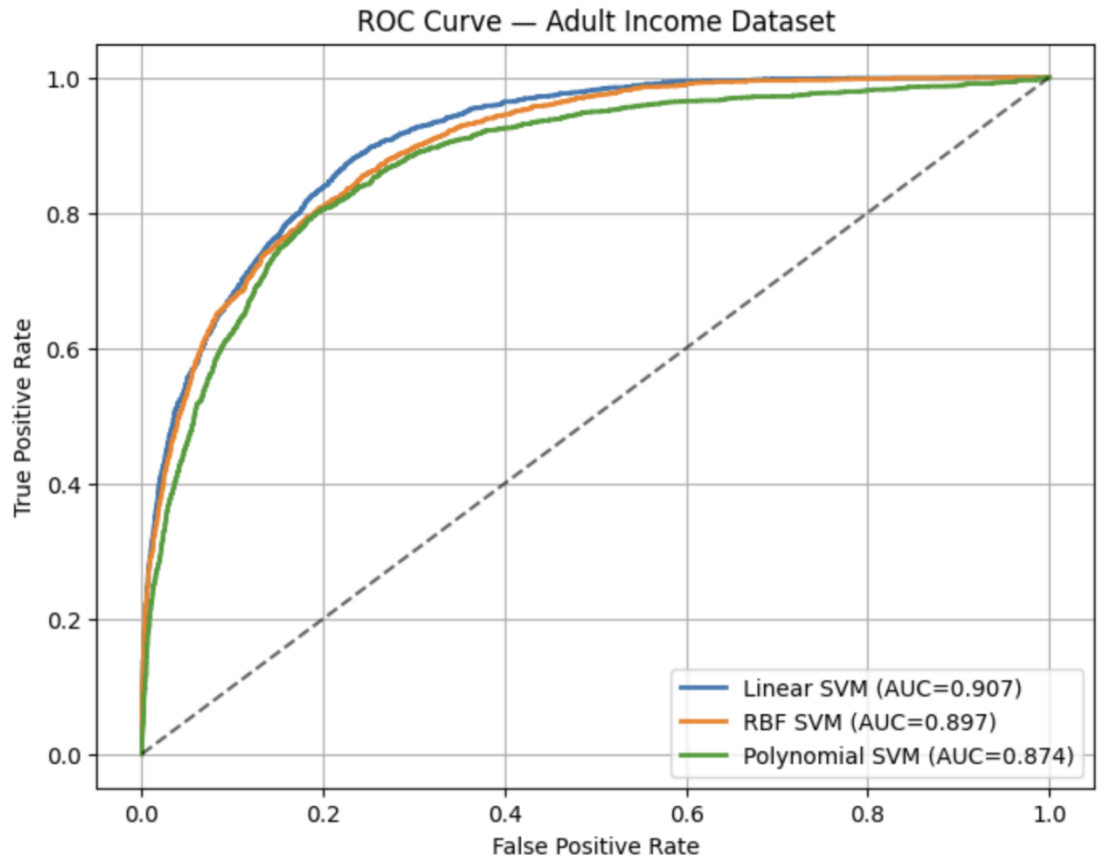


Both models had a strong ability to detect the <=50K class (high TN), but struggled more with the >50K class (lower TP and higher FN).

RBF SVM had slightly fewer false positives than Linear SVM, meaning it was a bit more cautious when predicting >50K.

However, it also missed more actual >50K cases (higher FN), which explains the slightly lower recall.

The differences were small, but they helped confirm what I saw in the precision and recall scores earlier.



When I plotted the ROC curves for the three SVM models I wanted to see how well each one could separate the two income classes across different thresholds, not just at a single decision point.

The results were clear: the Linear stood out with the highest AUC of 0.907, showing that it consistently distinguished between <=50K and >50K incomes. The RBF SVM followed closely with an AUC of 0.897, which means it was also strong but slightly less stable across thresholds. The Polynomial SVM, on the other hand, lagged behind with an AUC of 0.874, reinforcing the weaker performance I had already seen in its recall and F1 score. What this told me is that the Linear kernel wasn't just accurate—it was

also the most reliable when tested across the full range of possible decision boundaries.

The ROC curve gave me a deeper confirmation of the earlier metrics, showing visually and numerically why Linear and RBF kernels are better suited for this dataset, while the Polynomial kernel struggled to generalize.

Hyperparameatr Tuning:

After running the manual tuning for the Adult dataset, I started to see clear differences between how each SVM kernel behaves.
For the Linear kernel, lower regularization (C = 0.1) worked best, which made sense since the data seems close to linearly separable.
For the RBF kernel, the best balance came from C = 1 with gamma = 'scale', which gave stable results without overfitting.
The Polynomial kernel was the most sensitive. Even though C = 10, degree = 2 performed the best for this model, its accuracy was still slightly below the simpler kernels, which matched what I noticed earlier: the polynomial kernel tends to overfit much faster.

Overall this tuning step helped me see which kernels generalize better on this dataset and why the simpler Linear and RBF models consistently performed more reliably.

K Mean:

I wanted to see how well unsupervised learning could separate income levels in the Adult dataset, so I applied KMeans clustering with k = 2, since the target labels are binary.

The model doesn't use the actual labels during training, so I had to compare both possible label assignments to find the best match.

The final accuracy came out to about 75.78%, which at first glance seemed promising But when I looked deeper specifically at the confusion matrix I realized the model was heavily biased toward the majority class.

It correctly identified most of the <=50K cases but missed a huge portion of the >50K ones. Only 399 samples were correctly labeled as >50K, while over 10,000 were misclassified.

That told me the clustering wasn't really capturing the structure of the minority class at all. What stood out to me is how misleading accuracy can be in imbalanced datasets. Without the confusion matrix, I might've thought the model was doing well But seeing how it handled each class separately helped me understand the limitations of K Means in this context. It's not just about getting a good score it's about knowing what that score actually means.

Hierarchical Clustering:

To explore how hierarchical clustering performs on the Adult dataset, I applied both single and complete linkage methods using a binary cluster setup.

When I first tried hierarchical clustering on the full Adult dataset, the complete linkage method crashed my session after more than an hour of execution that was a clear sign that this algorithm, while conceptually elegant, isn't always practical for large datasets especially ones with high dimensionality like this one after one hot encoding and scaling.

To work around this, I decided to reduce the dataset to just the first 300 samples, but the execution still take a very long time, therefore I added the sample code to specify 10% of data for training. This allowed me to rerun both single and complete linkage methods without overwhelming the system.

After I corrected the sampling, the accuracy changed noticeably compared to my earlier attempt

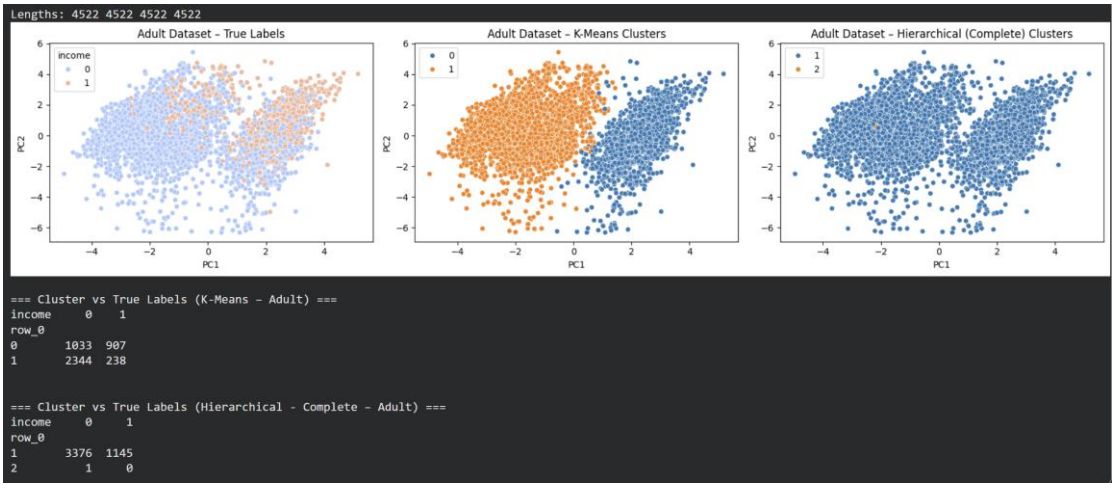
This shift made it clear to me that hierarchical clustering is very sensitive to the sampling ratio and small changes in the portion of data used can lead to big differences in performance.

What this really highlighted is the importance of carefully choosing how much data to include

especially when working with large and imbalanced datasets.

If the sampling isn't handled properly, the model can easily become biased toward the majority class, which weakens its ability to capture meaningful patterns. This experience showed me that with hierarchical clustering, the sampling strategy isn't just a technical detail—it directly shapes the quality of the results.

PCA:



To make sense of how the clustering algorithms were grouping the data, I used PCA to reduce the Adult dataset to two dimensions and visualize the results. The first plot, showing the true income labels, immediately revealed that the classes weren't clearly separated there was significant overlap, which hinted at the complexity of the dataset. When I looked at the K Means clustering results, I noticed that although the

clusters didn’t perfectly match the true labels, they still reflected some underlying structure. The confusion matrix supported this: K Means formed two distinct groups, even if many samples were misclassified. In contrast, the hierarchical clustering with complete linkage produced a much weaker result.

The PCA plot showed that nearly all points were assigned to a single cluster, and the confusion matrix confirmed that the second cluster contained only one sample. This visual comparison helped me see beyond the accuracy scores. It showed that KMeans was more responsive to the data’s distribution, while hierarchical clustering failed to capture meaningful patterns. PCA didn’t just simplify the data it gave me a clearer window into how each algorithm interpreted the structure, and why some methods worked better than others.

Comparative Analysis:

1. How does SVM performance compare to K-Means and Hierarchical on each dataset?

On the Cancer dataset, the difference between supervised and unsupervised approaches is that the SVM models performed almost flawlessly with the **Linear** kernel reaching near perfect accuracy (AUC close to 1.0) and both **RBF** and **Polynomial** kernels also achieving extremely strong results. **K Means** clustering did reasonably well at around 91%, but it still fell short compared to the precision of the SVM models. **Hierarchical** clustering struggled the most, with accuracy dropping to about 63% and a lot of overlap between clusters, which made separation weak and unreliable.

The Adult dataset showed a similar pattern.

Once again, SVM outperformed the clustering methods, with the tuned **Linear** and **RBF** models achieving the best accuracies

in the range of [84-85%].

K **Means** and **Hierarchical** clustering had difficulty handling the complexity and overlap in the data, which led to poor separation between income groups.

Taken together, these results highlight why SVM consistently came out on top: as a supervised method, it learns directly from the labels and builds

precise decision boundaries, while clustering methods rely only on the structure of the data. This makes SVM far more reliable for datasets where clear separation is needed, whereas unsupervised approaches often fall short when faced with imbalance or overlapping classes.

2.Which method seems to separate the classes better, and why?

On the Cancer dataset the results were straightforward.. The SVM models, especially the **Linear** kernel, achieved almost perfect separation between the two classes. This matched what I saw in the PCA visualization, where the true labels were naturally well separated, making a linear boundary highly effective. K Means was able to capture this natural structure fairly well, though not with the same precision as SVM, while **Hierarchical** clustering struggled showing significant overlap between clusters and weaker performance overall.

And on the Adult dataset presented a more difficult challenge. None of the methods produced a clean visual separation, which reflects the dataset's complexity and high dimensionality. Even so, SVM again stood out, with both the **Linear** and **RBF** kernels achieving the strongest results by learning flexible decision boundaries that adapted to the overlap in the data.

In contrast, the clustering methods **KMeans** and **Hierarchical** mixed the classes heavily, failing to capture meaningful structure.

Taken together, these findings highlight why SVM consistently performed better across both datasets. As a supervised method, it learns directly from the labels and optimizes margin based boundaries, which allows it to separate classes with precision. Clustering methods, by contrast, rely only on distances and the inherent structure of the data, which makes them far less effective when the classes are overlapping or imbalanced.

2. Which method seems to separate the classes better, and why?

On the Cancer dataset, the results were very clear. The SVM models, particularly the **Linear** kernel, achieved the strongest separation between the two classes. This matched what I saw in the PCA visualization, where the true labels were naturally well separated, making a linear boundary highly effective. **K Means** was able to capture this natural structure fairly well, though not with the same precision as SVM.

Hierarchical clustering, however, struggled the most showing significant overlap between clusters and weaker performance overall.

The Adult dataset presented a more complex challenge. None of the methods produced a clean visual separation, which reflects the dataset's high dimensionality and overlapping class structure. Even so, SVM repeatedly stood out, with both the **Linear** and **RBF** kernels achieving the best results by learning flexible decision boundaries that adapted to the complexity of the data. In contrast, the clustering methods **K Means** and **Hierarchical** mixed the classes heavily, failing to uncover meaningful separation.

Taken together, these findings highlight why SVM consistently outperformed the clustering approaches across both datasets. As a supervised method, SVM learns directly from the labels and optimizes margin based boundaries, which allows it to separate classes with precision. Clustering methods, by contrast, rely only on distances and the inherent structure of the data, making them far less effective when the classes are overlapping or imbalanced.

3. How did preprocessing (encoding, scaling) help your models?

Preprocessing turned out to be a critical step in both datasets. Scaling had a major impact on SVM performance, especially for the RBF and Polynomial kernels, since SVM is highly sensitive to feature magnitude. This is something I had already noticed

earlier when comparing kernel performance—without scaling, the results were far weaker. K-Means also benefited from scaling, because its distance-based nature can easily become biased when features are measured on different scales.

For the Adult dataset, encoding was absolutely essential. Converting categorical values into numeric ones was the only way to make the data usable for SVM and clustering algorithms. Without this step, the models would either perform poorly or fail altogether.

In the end, preprocessing made the datasets comparable, reduced bias between features, and allowed the models to learn meaningful patterns. As I've mentioned before, these steps aren't just technical details—they directly shape the quality of the results and explain why some models succeed while others struggle.

4. For each dataset, which method would you recommend, and why?

Breast Cancer Dataset

The SVM models, particularly the Linear and RBF kernels, delivered the highest accuracy and AUC. Since the PCA visualization showed that the classes were naturally well separated, a linear boundary worked almost perfectly This makes SVM especially

reliable for binary classification tasks in medical style datasets where clear separation is critical.

Adult Dataset

On the Adult dataset, SVM again achieved the best performance, with both the RBF and Linear kernels outperforming all other methods. Its ability to handle high-dimensional data allowed it to adapt to the complexity of the dataset. In contrast, clustering methods such as K Means and Hierarchical struggled, producing poor separation and failing to capture the underlying structure.

Overall Recommendation

Across both datasets, SVM consistently proved to be the most effective method. It achieved the highest accuracy, produced clearer boundaries, and benefited significantly from preprocessing steps like scaling and encoding. These results reinforce that supervised approaches like SVM are far more reliable than clustering methods when precise class separation is required.