Gender Detection Problem

By

Muhammmad Sarib Khan s298885

Contents Page

- Project’s goal
- Understanding the dataset
- Analysis of dataset
- Methodology of analysis
- Gaussian classifiers
- Linear Logistic Regression

Contents Page (cont'd)

- Linear SVM
- Quadratic SVM
- Gaussian Mixture Models
- Score Calibration
- Evaluation on test set
- Conclusion

Project’s goal

The objective of this project report is to examine a collection of different low-level images depicting males and females by utilizing diverse Machine Learning (ML) algorithms. It begins with conducting an examination of the dataset's structure, aiming to comprehend its distribution. Subsequently, multiple classifiers will be analyzed in order to develop a system capable of effectively classifying our samples with minimal expenses.

Understanding the dataset

- The training dataset comprises 2400 samples, with 720 males and 1680 females, making it heavily skewed towards females, accounting for near about 70% of the dataset.
- Each sample consists of 12 features and represents a low-dimensional image achieved by mapping the images onto a shared low-dimensional manifold.
- These embeddings lack a tangible interpretation, and the samples are categorized into three distinct age groups, each characterized by a unique distribution of the embeddings. However, no age information is available.
- On the other hand, the test dataset contains 6000 samples, including 4200 males and 1800 females.
- Consequently, the class proportions in the test dataset do not correspond to those in the training dataset, but all other information is maintained.

Analysis of Dataset

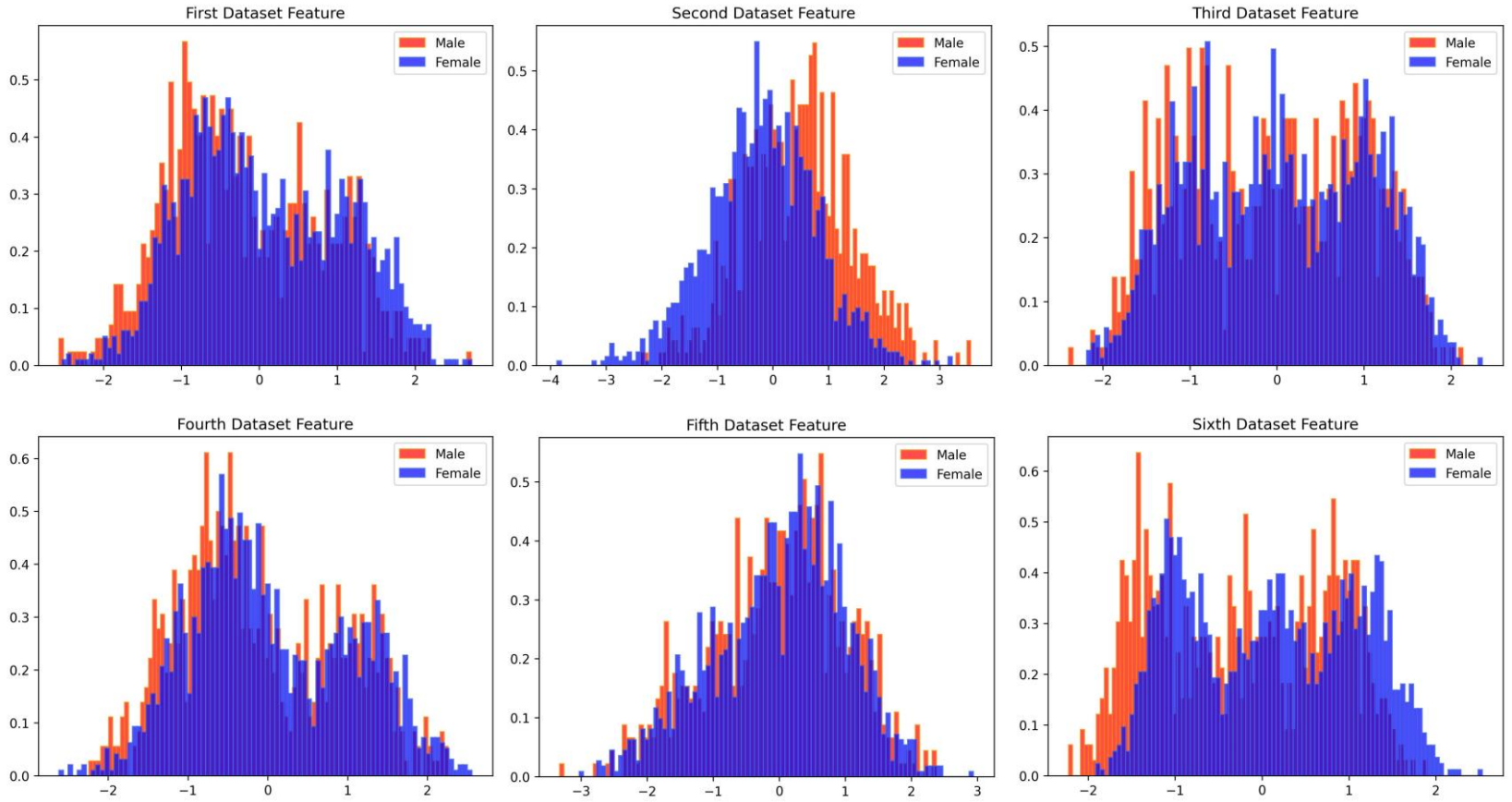
- We will now commence the analysis of the dataset by generating plots to visualize the distribution of our features using histograms.
- To simplify calculations and prevent overflows, the data was preprocessed using Z-Normalization, which involves centering and scaling the data to have unit variance.
- Each candidate, denoted as xi, was transformed using the formula:

$$xi = (xi - \mu) / \sigma$$

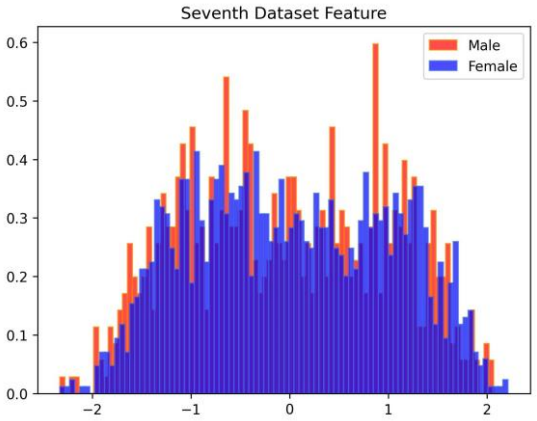
- Here, μ represents the mean of the data and σ represents the standard deviation. This transformation ensures that the data is centered around zero and has a standard deviation of one.

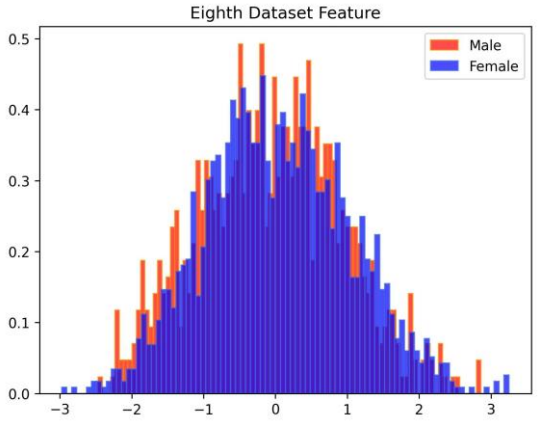
Analysis of Dataset

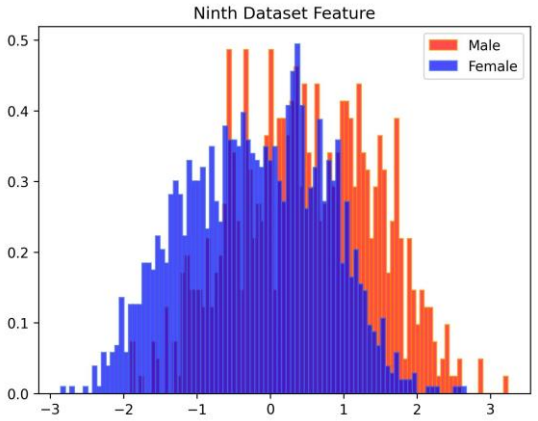
- Below features preprocessed with Z-Normalization have been plotted:

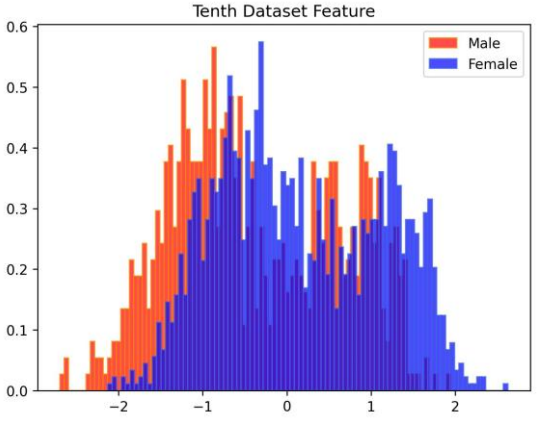


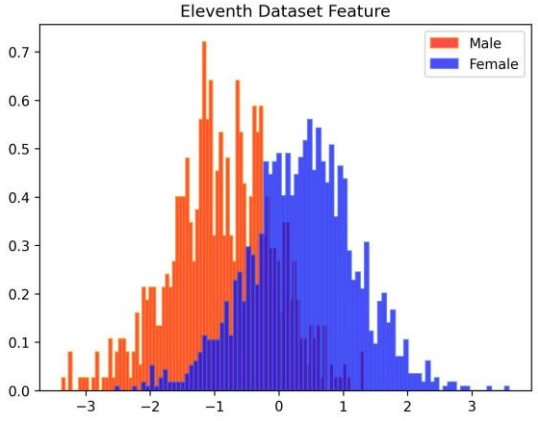
Analysis of Dataset

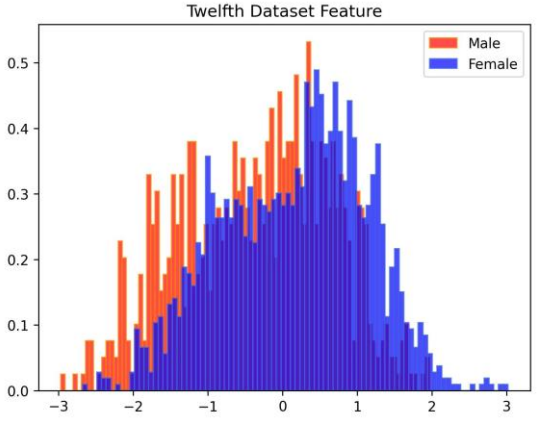










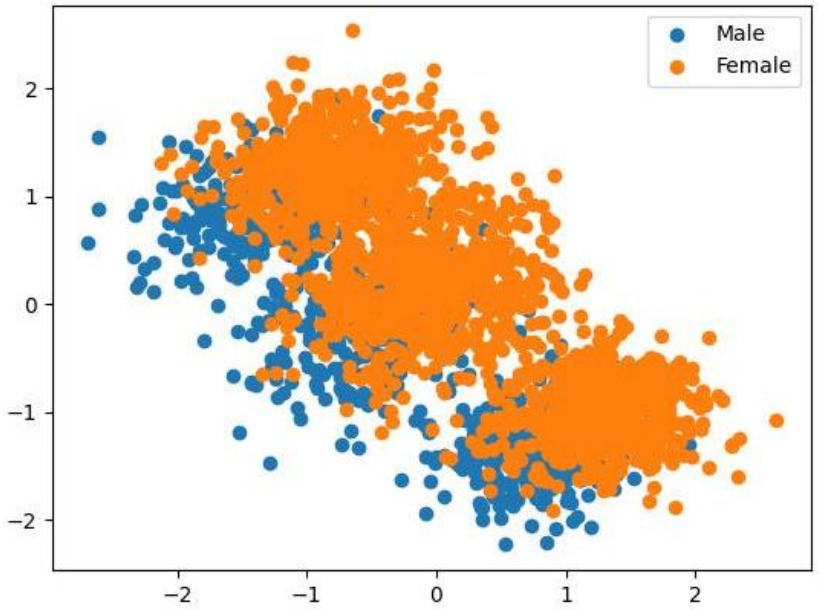


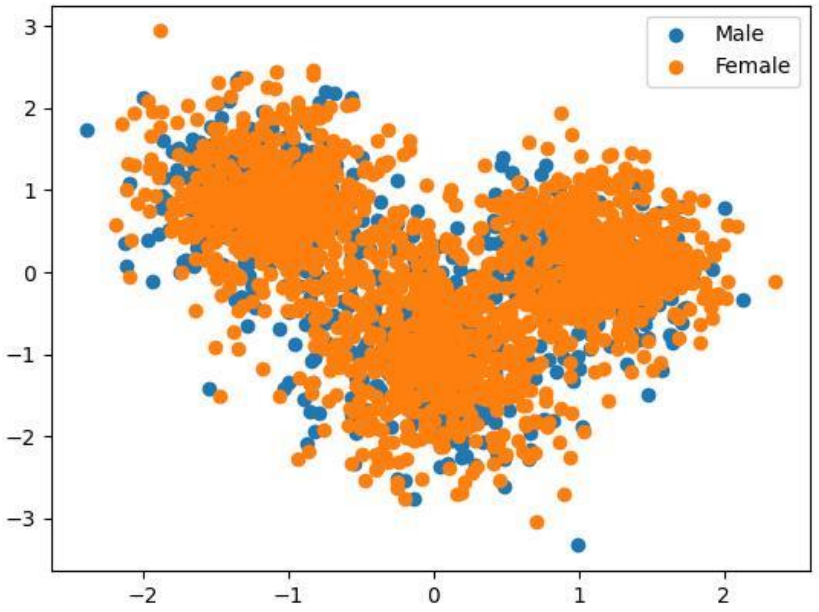
Analysis of Dataset

- Upon observation, it is evident that certain histograms exhibit a resemblance to a Gaussian distribution with three components.
- This pattern aligns with the three age groups mentioned earlier, specifically referring to the histograms of the third, sixth, and, to some extent, seventh feature of the dataset
- The distribution for a single feature is very similar for both classes.
- However, one certain distribution is the most discriminant feature within our dataset. The eleventh feature of our dataset looks the most discriminant

Analysis of Dataset

- To analyze the distribution of the dataset with more clarity, scatterplots were used with different features against each other for both classes
- The first plot is between the ninth feature and the fifth feature
- The second plot is between second and the fourth feature



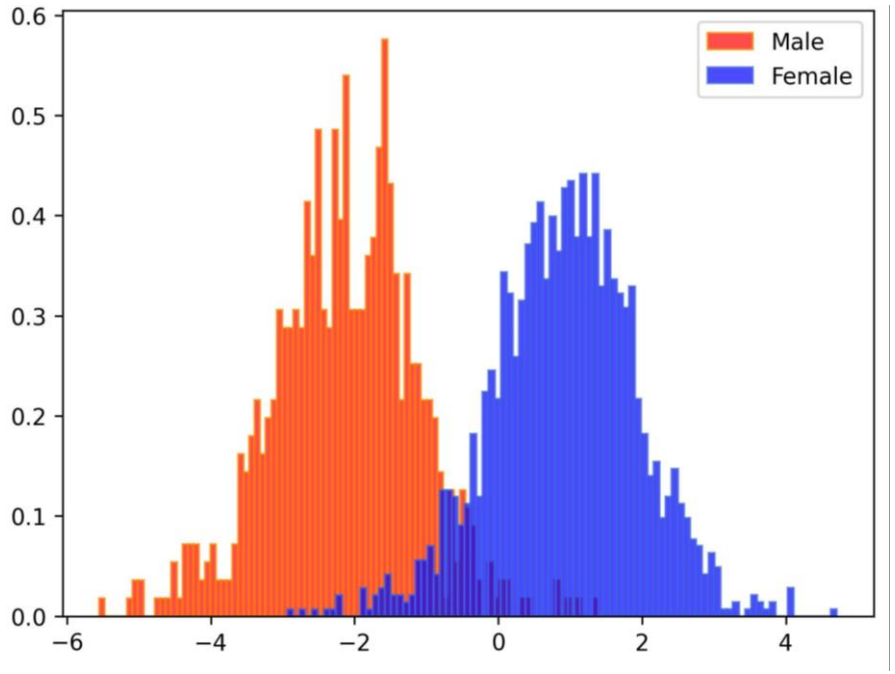


Analysis of Dataset

- The inference that can be made from looking at the scatterplots is that the distribution does appear to be gaussian. Gaussian based models are expected to perform well
- Also, in the both scatterplots, roughly 3 clusters can be identified which is an indicator of 3 different age groups

Analysis of Dataset

- Likewise, Linear discriminant analysis can be applied to the dataset to predict how linear classifiers will perform
- Below is the output when LDA was performed on the concerned dataset.
- It is to be remembered that since this is a binary classification problem, we will only get 1 direction, since LDA finds no.of classes – 1 directions to plot our features



Analysis of Dataset

- From the plot of LDA, classes look very decently separable and discriminable. However, there is some area where misclassification is possible

Analysis of Dataset

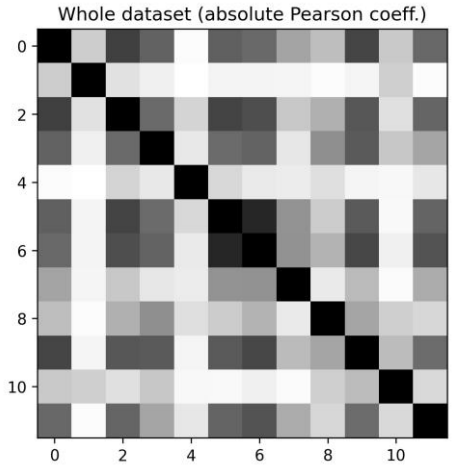
- To examine the correlation between features, heatmaps were generated to depict correlations using the absolute value of the Pearson correlation coefficient formula:

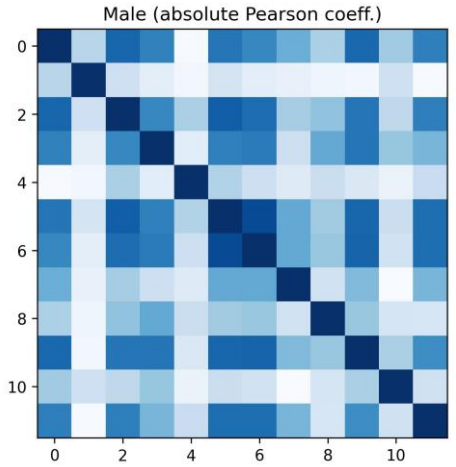
$$corr = Cov(X, Y) / \sqrt{(Var(X) * Var(Y))}$$

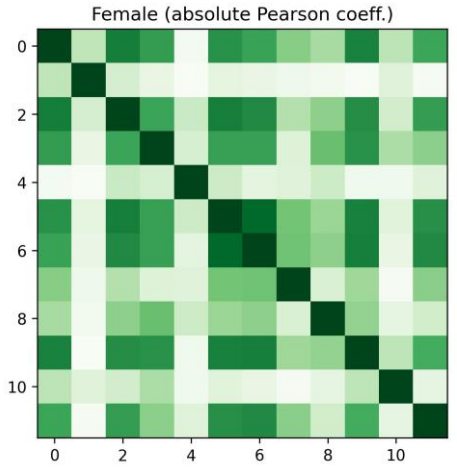
- This approach enables the identification of the correlated features and determine the extent of their correlation.
- It provides valuable insights into the data and offers hints on how Principal Component Analysis (PCA) can be applied to the dataset.

Analysis of Dataset

- Below are three heatmaps showcasing the analyzed features across the entire dataset, the male set, and the female set.
- Darker colors on the heatmaps indicate a stronger correlation with an absolute value of the Pearson coefficient closer to 1.







Analysis of Dataset

- The analysis reveals that specific features, such as 6&9, and 3&6, exhibit a strong correlation, indicating a significant relationship between them.
- On the other hand, some features, like 1&4, show no correlation, implying that they are independent of each other.
- Additionally, most of the features demonstrate a slight correlation with one another, suggesting a weak relationship between them.
- This does suggest that we can apply dimensionality reduction and maintain useful information after applying PCA and reducing the dimension to 11 or 10
- Later it will be seen, that 11-dimensional space was chosen after extensive experimenting with different number of dimensions. The experimentation was also done to demonstrate how useful information is significantly lost if we go down the dimensions a lot

Analysis of Dataset

- Considering the individual heatmaps for the classes, they depict similar behavior as per the whole dataset
- It can be inferred that since there is correlation between features, naïve-bayes classifier(including the tied naïve-bayes) will perform the worst because it assumes independence between features
- It can also be inferred the full-covariance, and tied full-covariance models are expected to perform better since there is considerable correlation between features (some weak and some strong)

Methodology of analysis

- To evaluate which model will perform better and assess the impact of using PCA, we can employ two methodologies:

- Single Split: In this approach, the training set is divided into two subsets: one for training and the other for validation. We typically use a split ratio of 66% for training and 33% for validation. The final classifier is evaluated on the validation set.

- K-Fold cross-validation: Here, the training set is divided into K distinct sets. Each iteration involves using K-1 sets for training and 1 set for validation. The final classifier is obtained by retraining the model on the entire training set.

- Both approaches will be considered for our analysis. However, due to the high imbalance of the dataset, we anticipate more robust results from the K-Fold cross-validation method.

Methodology of analysis

- Our primary application assumes a uniform prior, but we also consider unbalanced applications where the prior is biased towards one of the two classes. The settings for these biased applications are as follows:

  Main application: ($\tilde{\pi}$, Cfp, Cfn) = (0.5, 1, 1)

  Biased application: ($\tilde{\pi}$, Cfp, Cfn) = (0.1, 1, 1)

  Biased application: ($\tilde{\pi}$, Cfp, Cfn) = (0.9, 1, 1)

- The quality of the models will be evaluated based on Minimum Detection Costs. These costs correspond to the expense we would incur if we made optimal decisions using the optimal threshold derived from the training set and applied to the validation subset using the aforementioned techniques.

Methodology of analysis

Through the implementation of these rigorous evaluation techniques and the exploration of various preprocessing methods, our objective is to determine the most effective model. This model should be capable of addressing the challenges posed by the imbalanced dataset and consistently provide accurate results in diverse application scenarios.

Also, all the scores are Z-scores because as mentioned before I standardized the data. I used the word “raw” for portraying that the “Raw” data in this documentation means without any PCA

Gaussian Classifiers

- We will now commence our analysis on Machine Learning models, specifically focusing on Gaussian classifiers such as MVG (Full covariance) and Naive Bayes (Diagonal covariance).
- Additionally, we will consider the Tied assumption for these classifiers, where each class has its own mean (μc), but the covariance matrix (Σ) is computed over the entire dataset.

- Gaussian classifiers operate based on the assumption that the data follows a Gaussian distribution, represented as:

$$(Xi \mid Ci = c, \Theta) \sim (X \mid C = c, \Theta) \sim N(\mu c, \Sigma c)$$

Gaussian Classifiers

- The Naive Bayes version of the classifier, with its assumption of feature independence, is expected to yield poorer results. This assumption implies that the covariance matrix's off-diagonal elements are zero, indicating no correlation among features. However, since some of our features display strong correlations, we anticipate suboptimal outcomes from this classifier.

- Conversely, the Tied version of the MVG classifier should provide better results due to its ability to capture correlations and the larger number of available samples. Below are the obtained results from our analysis.

Gaussian classifiers(K-folds=5)

π	0.5	0.1	0.9
Full-Covariance	0.113	0.297	0.350
Diagonal-Covariance	0.463	0.771	0.777
Tied Full-Covariance	0.109	0.299	0.342
Tied Diagonal-Covariance	0.457	0.770	0.781

No PCA

π	0.5	0.1	0.9
Full-Covariance	0.122	0.312	0.358
Diagonal-Covariance	0.124	0.312	0.349
Tied Full-Covariance	0.118	0.299	0.354
Tied Diagonal-Covariance	0.123	0.294	0.355

PCA m=11

Gaussian classifiers(K-folds=5)

π	0.5	0.1	0.9
Full-Covariance	0.187	0.407	0.538
Diagonal-Covariance	0.184	0.435	0.546
Tied Full-Covariance	0.183	0.428	0.535
Tied Diagonal-Covariance	0.179	0.421	0.543

PCA m=10

π	0.5	0.1	0.9
Full-Covariance	0.220	0.500	0.578
Diagonal-Covariance	0.208	0.486	0.597
Tied Full-Covariance	0.212	0.476	0.577
Tied Diagonal-Covariance	0.210	0.482	0.589

PCA m=9

Gaussian classifiers(Single-Split)

π	0.5	0.1	0.9
Full-Covariance	0.263	0.494	0.631
Diagonal-Covariance	0.251	0.501	0.596
Tied Full-Covariance	0.257	0.461	0.584
Tied Diagonal-Covariance	0.257	0.433	0.614

No PCA

π	0.5	0.1	0.9
Full-Covariance	0.115	0.223	0.309
Diagonal-Covariance	0.119	0.211	0.246
Tied Full-Covariance	0.114	0.217	0.317
Tied Diagonal-Covariance	0.115	0.175	0.291

PCA m=11

Gaussian classifiers(Single-split)

π	0.5	0.1	0.9
Full-Covariance	0.200	0.386	0.552
Diagonal-Covariance	0.176	0.368	0.575
Tied Full-Covariance	0.182	0.369	0.561
Tied Diagonal-Covariance	0.174	0.348	0.573

PCA m = 10

π	0.5	0.1	0.9
Full-Covariance	0.223	0.406	0.615
Diagonal-Covariance	0.205	0.439	0.612
Tied Full-Covariance	0.197	0.421	0.607
Tied Diagonal-Covariance	0.196	0.365	0.603

PCA m=9

Gaussian Classifiers

- Following inferences were made from the results:
  - The results are mainly in line with our expectations. It was expected that naïve-bayes will perform poorly and it did
  - As we go down in dimensionality, the performance decreased. Even by going to 11 dimensions, in some cases naïve-bayes outperformed the full-covariance classifier.
  - Naïve-bayes did perform very well when dimensions were further reduced. PCA provides us with orthogonal directions to visualize our data. The orthogonality of these directions implies independence. Therefore, the resulting projected data will be independent which leads to better performance
  - In most of the cases, tied full-covariance model performed the best which was expected
  - As far as single-split is concerned, the results are highly inconsistent which was expected due to the imbalance of classes. Hence, it will be dropped in further analysis
  - Also, dimensionality reduction hasn't proved to be that fruitful. For the sake of comparison, I kept going for PCA m=11 but better and more robust results are expected from raw data without any dimensionality reduction
  - Since, Tied Full-covariance model outperformed full covariance model it could be that linear classifiers might outperform the quadratic ones later

Gaussian Classifiers

- Following inferences were made from the results:
  - The reason behind the similarity in performances between MVG (Full covariance) and TMVG (Tied covariance) is that PCA retains the principal discriminant information in the leading directions. As a result, both classifiers, MVG and TMVG, do not lose this crucial information. Consequently, their performances remain comparable and similar.

Logistic Regression

- Based on logistic(sigmoid function) mapping any real number between 0 and 1
- Assumption: linear relationship between predictor and log of odds of binary response variable
- We expect this linear classifier to work good since we obtained good results by tied-covariance gaussian models.

Logistic regression

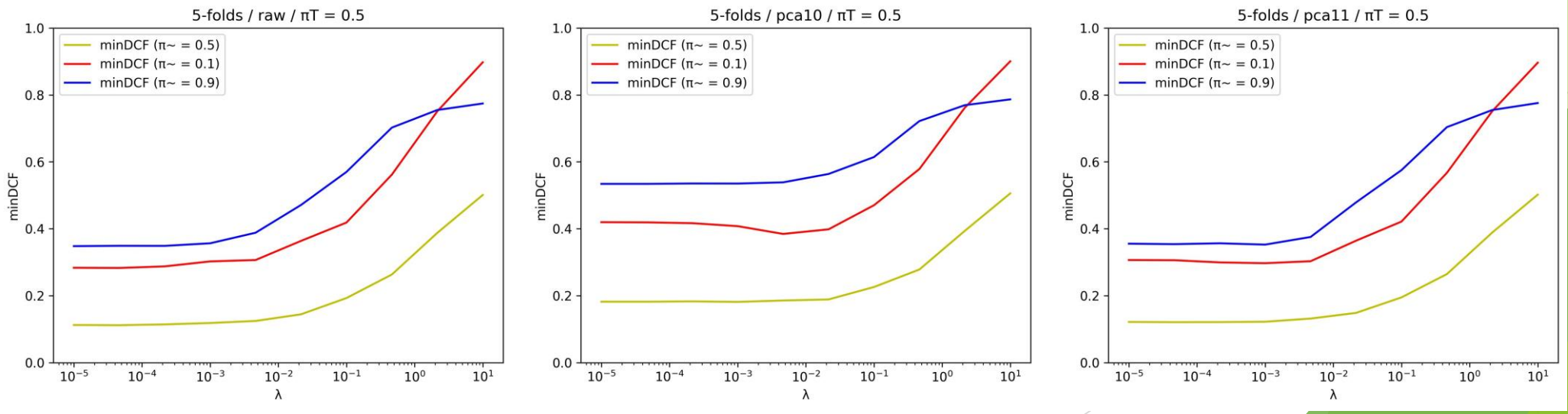
- A regularized version of objective function is used because classes are unbalanced:

$$J(\mathbf{w},b)=\frac{\lambda}{2}\|\mathbf{w}\|^2+\frac{\pi_T}{n_T}\sum_{i=1|c_i=1}^{n}\log\left(1+e^{-z_i(\mathbf{w}^T\mathbf{x}_i+b)}\right)+\frac{1-\pi_T}{n_F}\sum_{i=1|c_i=0}^{n}\log\left(1+e^{-z_i(\mathbf{w}^T\mathbf{x}_i+b)}\right)$$

- Λ is our hyperparameter called regularization coefficient. It controls the amount of shrinkage of weights in training process (regularize the norm of w). If it is too big then more shrinkage, which means a simpler model, but we risk underfitting. Hence poor separation of classes
- If regularization coefficient is almost equal to zero, then the classes are well separated but generalization on unseen data is subpar. It can lead to overfitting
- In order to find the perfect tradeoff for our hyperparameter, we will set πT(threshold probability in logistic regression) to 0.5(since this is our main application) and plot minDCF against regularization coefficient

Logistic Regression

- From the plots it can be observed that 10^{-5} can prove to be a decent value for regularization coefficient. By selecting this value of λ, we aim to strike a balance between minimizing overfitting and achieving comparable performance to other small λ values and enhances its capability to generalize effectively to new and unseen data.
- It can also be observed that both pca and no pca, the results are not too different. But best results are provided without the pca and only those will be reported in the following slide



Logistic regression(5-folds)

π	0.5	0.1	0.9
LogReg($\lambda = 10^{-5}$, $\pi T = 0.5$)	0.112	0.283	0.348
LogReg($\lambda = 10^{-5}$, $\pi T = 0.1$)	0.121	0.296	0.377
LogReg($\lambda = 10^{-5}$, $\pi T = 0.9$)	0.110	0.315	0.343

No PCA

Logistic Regression

- When experimenting with different values for πT, we observe that it does not significantly enhance the model's performance.
- Although there is a slight improvement when using πT = 0.9, it can be attributed to the imbalanced nature of the dataset, where the female class (class 1) dominates.
- In general, varying the value of πT does not yield substantial improvements in the model's overall performance.
- Ultimately, we did see good performance from logistic regression model. Very close to our previous best (TMVG classifier). This strengthens our belief in linear classifiers

Support Vector Machines

- Next, we will shift our focus to a non-probabilistic models which is Support Vector Machine models (SVMs).
- It is referred to as non-probabilistic because the scores it produces does not have a probabilistic interpretation. We will begin by considering the linear SVM and, like our previous analysis, we will examine the model and its underlying assumptions.

Support Vector Machines

- The aim is to find hyperplane that maximizes the margin
- Margin is the distance between the hyperplane and nearest data point of each class
- points closest to hyperplane are vectors, they determine position and orientation of hyperplane
- Advantage of SVM: can handle non-linear data using kernel trick. Mapping data into higher dimensional space so that it becomes linearly separable. Hence, non-linear relationships can be captured

Support Vector Machines

In our analysis, we will consider the prior-weighted version of the model for SVM.

This means that the objective function, or the dual problem for SVM, which I aim to maximize will be formulated as follows:

$$J^D(\boldsymbol{\alpha}) = -\frac{1}{2}\boldsymbol{\alpha}^T \boldsymbol{H}\boldsymbol{\alpha} + \boldsymbol{\alpha}^T \mathbf{1} \qquad with\ 0 \leq \alpha_i \leq C, \forall i \in \{1, ..., n\}$$

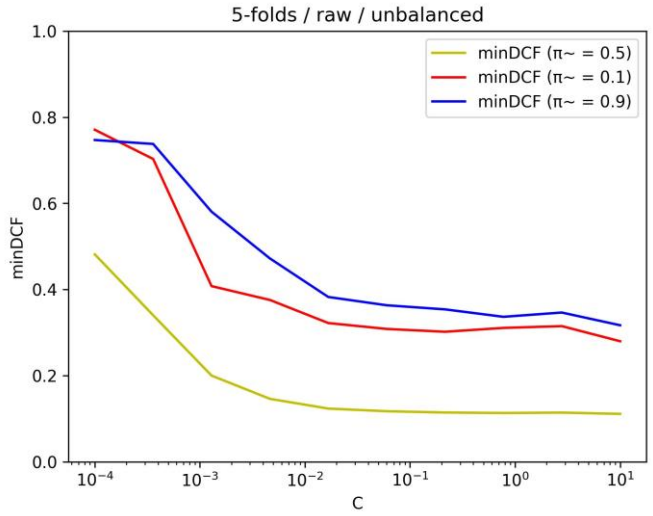
$$where\ C_i = \begin{cases} C\frac{\pi_T}{\pi_T^{emp}} & \text{if } i \in Class_1 \\ C\frac{\pi_F}{\pi_F^{emp}} & \text{if } i \in Class_0 \end{cases}$$

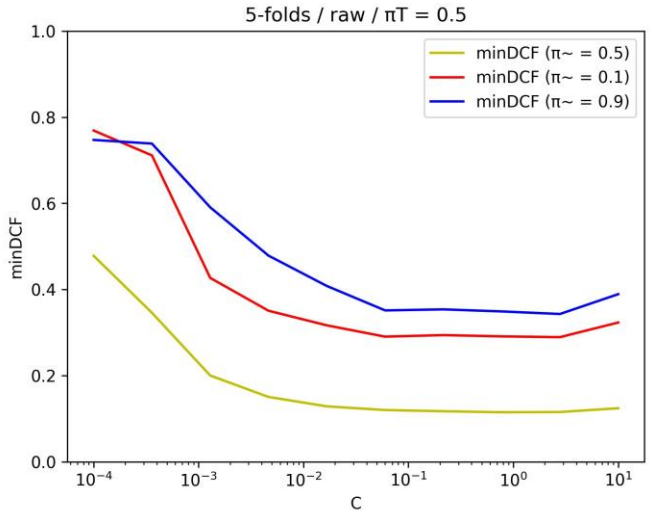
It is to be noted that Π^{emp} is empirical prior is proportions of sample in training set corresponding to each class (Depending on t or F; t means class 1; f means class 0

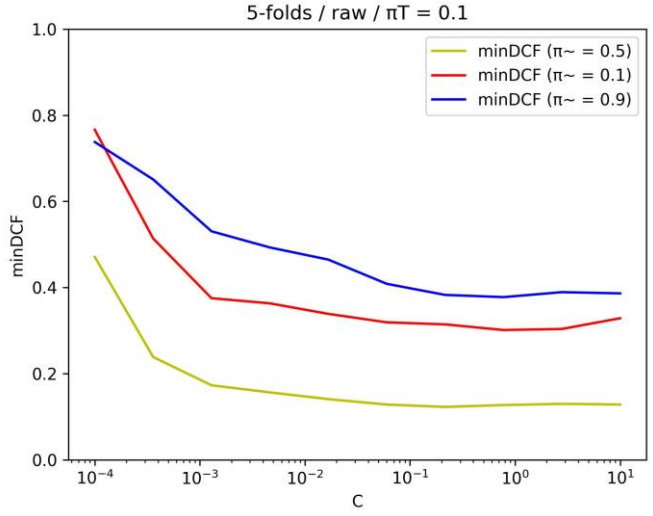
Support vector machines

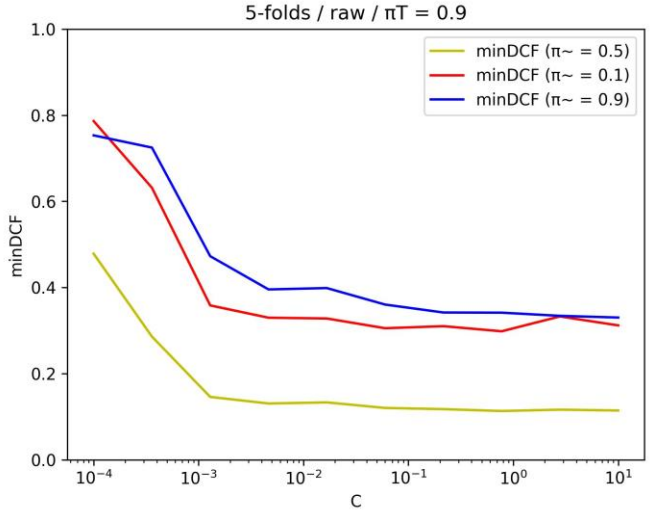
- Considering linear svm, tuning of hyperparameter c is required to balance the classes. Different values of c are considered.
- C is the regularization parameter that controls the trade-off between margin maximization and misclassification minimization
- In order to select our best value of c, minDCF graphs are plotted for both balanced and unbalanced SVM
- Ideally, increasing C for minority class so penalty increases to misclassify them, and decreasing C for majority class so penalty to misclassify them decreases is the norm that is followed when tuning the SVM model

Support Vector Machines









Support vector machines

- If C is to approach infinity, we will have a low error on training, but overfitting will emerge as an issue. Similarly, if C is too small, we risk underfitting as now we are going for a big margin with little care for training data
- We will be choosing our c as 10 because it gives the most optimized results in all the applications considered
- Only unbalanced application will be considered since rebalancing is not making a lot of difference.

Linear SVM(5-folds)

RAW data

π	0.5	0.1	0.9
Linear SVM(C = 10 , unbalanced)	0.111	0.280	0.317
Linear SVM(C = 10 , πT = 0.5)	0.124	0.323	0.389
Linear SVM(C = 10 , πT =0.1)	0.128	0.329	0.386
Linear SVM(C = 10 , πT =0.9)	0.114	0.312	0.330

Support Vector Machines

- The results look good and are in line with our expectation that linear classifier will perform well.
- We also got good results when $\pi T = 0.9$ because of the imbalance in classes
- Nevertheless, balancing did not give very significant improvements. The best application is still the unbalanced one. So in final evaluation only that will be considered

Quadratic Support Vector Machines

- It is an extension of SVM
- Using kernel functions, input features are mapped in higher-dimensional space
- The hyperplane is then figured out in the transformed space

Quadratic Support Vector Machines

- The dual SVM formulation relies on dot products between samples, represented as:

$$H_{ij} = z_i z_j x^T_i \, x_j$$

- This property eliminates the need for explicit feature expansion to compute scores. It is sufficient to compute scalar products between training and test samples.

- By efficiently calculating dot products in the expanded space, using a kernel function denoted as: $k(x_i, x_j) = \Phi(x_i)^T \, \Phi(x_j)$, we can perform both training and scoring.

Quadratic Support Vector Machines

- The kernel function enables the computation of a linear separation surface in the expanded space, which corresponds to a non-linear separation surface in the original feature space.
- We can utilize two different types of kernel functions:

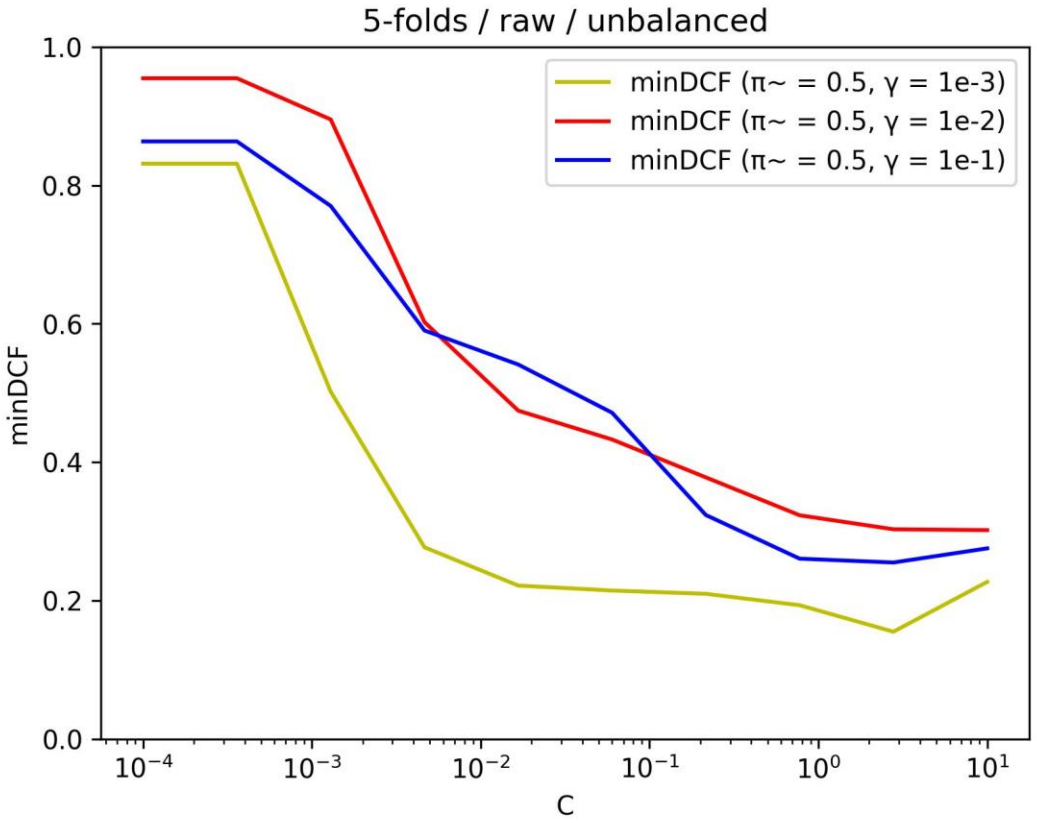
  Radial Basis Function (RBF): $e^{(-\gamma||x_i-x_j||^2)}$

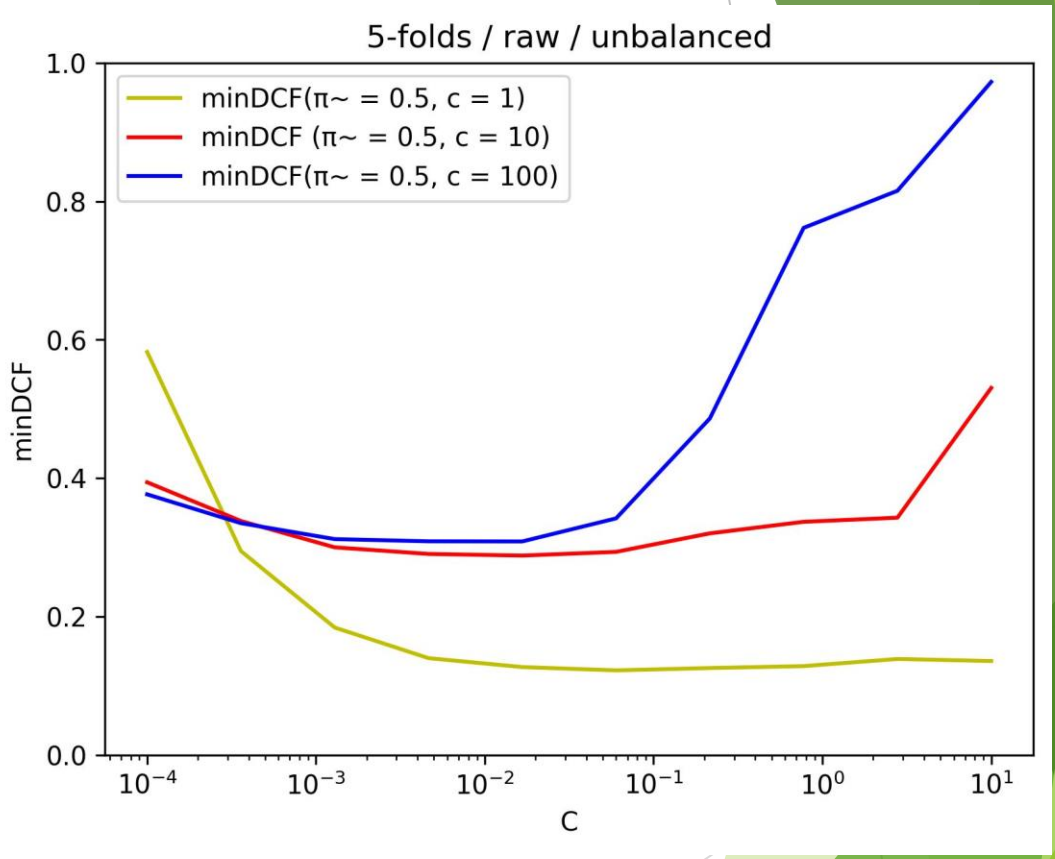
  Polynomial: $(x_i^T x_j + c)^d$

Quadratic Support Vector Machines

- We will investigate two versions of SVM: the quadratic SVM using a polynomial kernel with a degree of 2 and the radial basis SVM.
- Both models depend on the hyperparameter C, while the radial version additionally relies on another parameter y(gamma). We have fine-tuned the value of gamma, considering three options: 10^{-1}, 10^{-2}, and 10^{-3}

Quadratic Support Vector Machines





Quadratic SVm(5-folds)

No PCA

π	0.5	0.1	0.9
RBF SVM(C = 1 , $\gamma = 10^{-3}$)	0.210	0.320	0.392
Poly SVM(C = $1e^{-1}$, c = 1, d = 2)	0.129	0.352	0.345

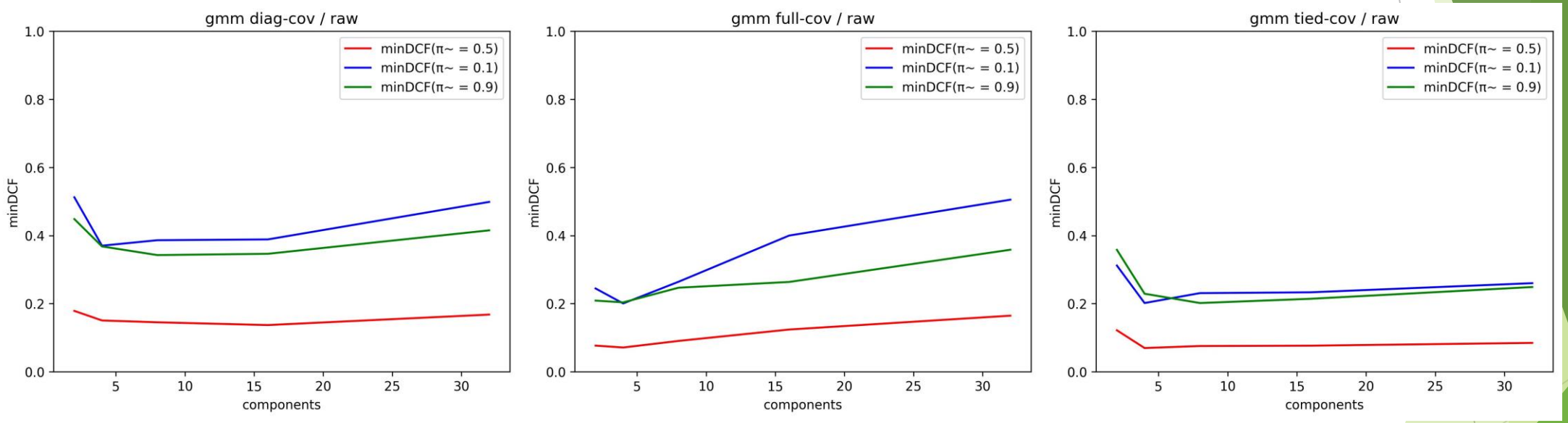
Quadratic Support Vector Machines

- Here we can see the quadratic SVM performed poorly than its linear counterpart
- Polynomial model (C = 1e-1, c = 1, d = 2) is the better of the two and can be used as a benchmark for comparison if more quadratic models were to be trained, for example, quadratic LR. But due to time constraints, I opted against that
- So far, linear classifiers are outperforming the non-linear classifiers
- Henceforth, there is no good reason to consider this in final evaluation

Gaussian Mixture Models

- In concluding our classification task, we will discuss Gaussian Mixture Models (GMMs).
- GMMs operate on the assumption that each sample is generated from a mixture of a finite number of Gaussian distributions with unknown parameters.
- The number of these distributions is a hyperparameter that needs to be determined. To estimate the correct number of components, we will plot different types of GMMs, including Full Covariance, Diagonal Covariance, and Tied Covariance.
- These plots will aid us in identifying the appropriate number of components for our GMM model.
- We anticipate achieving the best results with Gaussian Mixture Models that have 2 to 4 components. This is because our training set resembles a Gaussian distribution with 3 components or clusters.

Gaussian Mixture Models



GMM

- GMM Tied covariance with 4 components performs the best here. It will be taken into consideration as a possible candidates along with tied full-cov, linear svm and logistic regression. It was anticipated that a 4-component model will perform better due to then nature of our data. Both full-covariance and tied-covariance performed well

π	0.5	0.1	0.9
GMM Full Cov (4 components)	0.071	0.201	0.204
GMM Diagonal Cov (16 components)	0.197	0.501	0.474
GMM Tied Cov (4 components)	0.068	0.237	0.222

No PCA

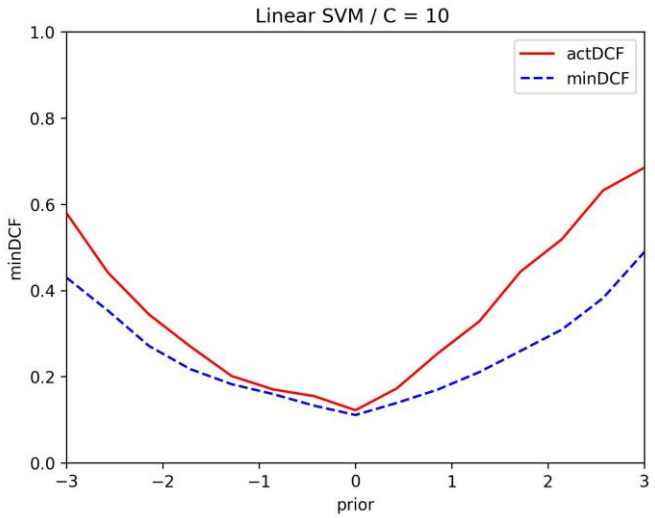
Score Calibration

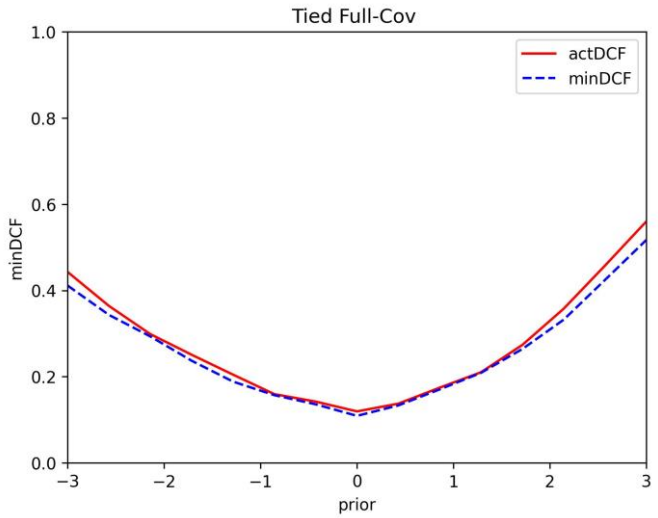
- Now, let's focus on calibrating the best performing models, which include:
  - MVG with Tied Covariance
  - Logistic Regression ($\lambda = 10^{-5}$, $\pi T = 0.9$)
  - Linear SVM (C = 10, unbalanced)
  - GMM with Tied-Covariance (4 components)

- Previously, we have considered minDCF as the evaluation metric. However, the actual cost we would incur depends on the accuracy of the threshold used for class assignment. To account for this, we introduce the concept of actual DCF.

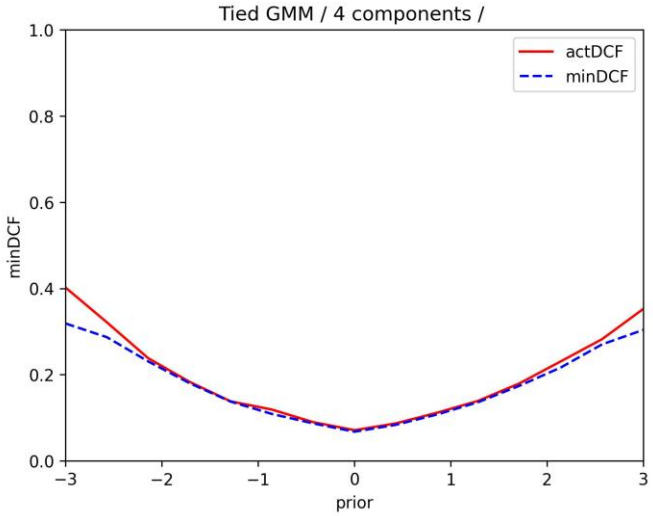
Score Calibration

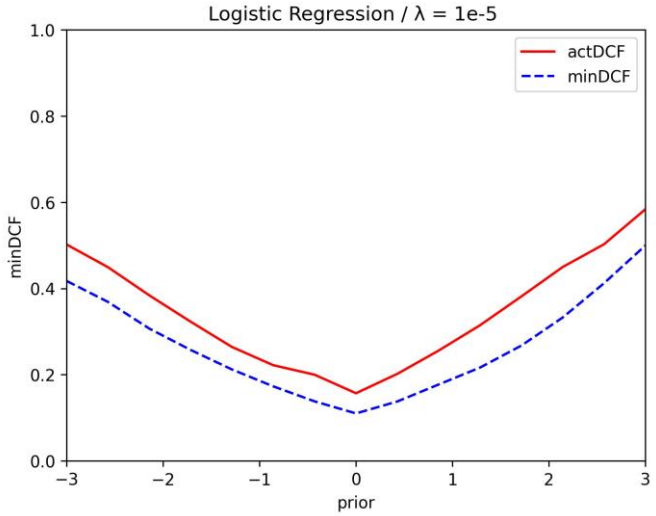
- Unlike minDCF, actual DCF uses the threshold corresponding to ˜π (the prior probability) for classification instead of testing all possible thresholds and selecting the lowest DCF.
- The Bayes Error Plots that follow highlight the differences between minDCF and actDCF for the selected models. The gap indicates losses due to miscalibration of scores





Score Calibration





Scores Calibration

From the plots, it is evident that minDCF differs from actDCF by a significant amount in many cases.

So, we must use a technique of score calibration in order to compute transformation function to map uncalibrated score into calibrated ones.

$f(s) = \alpha s + \beta$ => Here we are assuming that the function is linear in s.

The function f(s) can be interpreted as the log-likelihood ratio between the two classes, providing well-calibrated scores. The class posterior probability for a given prior $\tilde{\pi}$ can be expressed as:

$$\log(P(C = Class1|s) / P(C = Class0|s)) = \alpha s + \beta + \log(\tilde{\pi} / (1 - \tilde{\pi}))$$

Scores Calibration(cont'd)

Here, α and β are coefficients, and $\tilde{\pi}$ represents the prior probability. This equation captures the relationship between the log-likelihood ratio, the prior probability, and the class posterior probabilities.

By interpreting s as features and rewriting the equation, we can derive the same expression for the log posterior ratio of Logistic Regression. If we define $\beta_0 = \beta + \log(\tilde{\pi} / (1 - \tilde{\pi}))$, then the model needs to be trained to learn the parameters α and β_0.

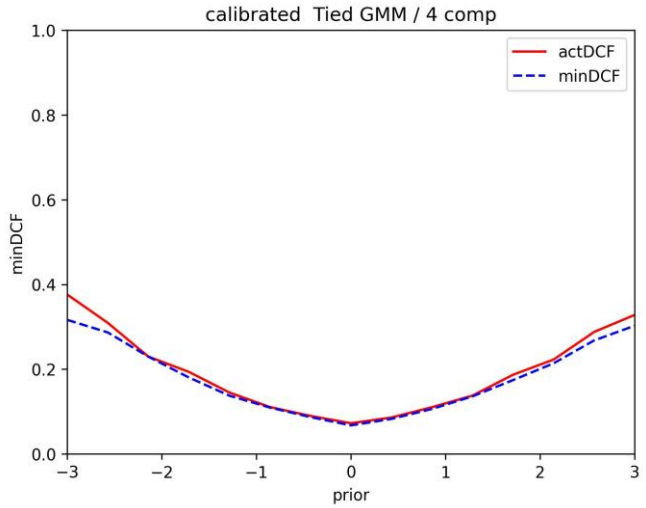
The equation for recovering calibrated scores is given by:

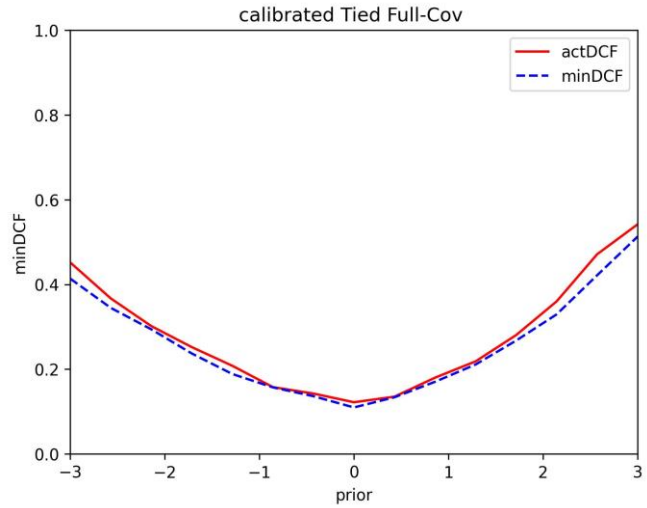
$$f(s) = \alpha s + \beta_0 - \log(\tilde{\pi} / (1 - \tilde{\pi}))$$

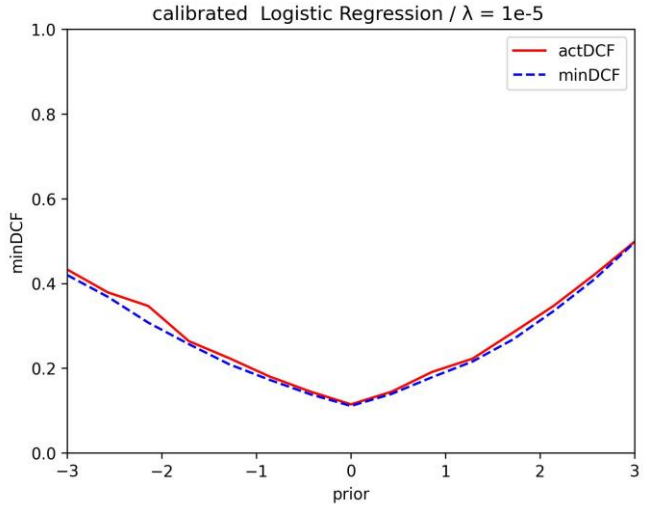
In our case, we choose $\tilde{\pi} = 0.5$ as the prior probability to optimize our application for this specific scenario. Additionally, we try a hyperparameter value of $\lambda = 10^{(-5)}$ for the calibration of Logistic Regression.

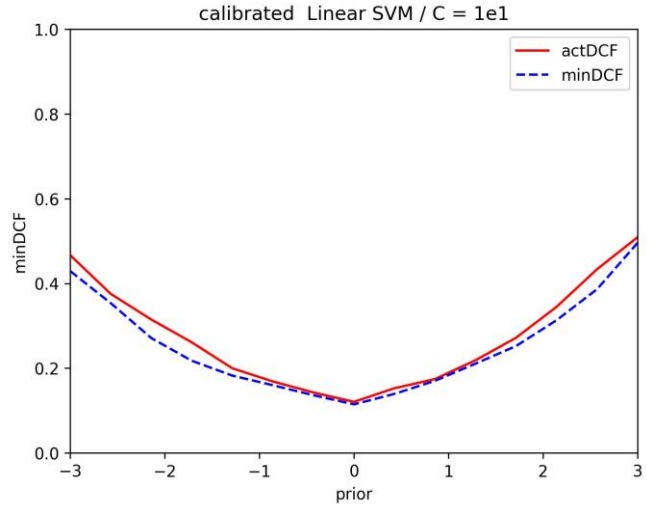
.

Score Calibration (Calibrated plots)









Score Calibration

- The utilization of score calibration allowed us to effectively transform mis calibrated scores into well-calibrated ones across a diverse range of applications. This process ensured that the scores produced by the models were reliable and accurately reflected the underlying probabilities, enhancing the overall performance of the models in various contexts.

SCORE CALIBRATION

π	0.5	0.1	0.9
Tied Full-Covariance	0.110	0.299	0.339
LogisticReg(10^{-5} , 0.9)	0.111	0.315	0.345
SVM(C = 10)	0.115	0.280	0.321
GMM Tied Covariance(4)	0.067	0.236	0.220

minDCF

π	0.5	0.1	0.9
Tied Full-Covariance	0.122	0.301	0.372
LogisticReg(10^{-5} , 0.9)	0.115	0.346	0.356
SVM(C = 10)	0.121	0.312	0.359
GMM Tied Covariance(4)	0.072	0.236	0.224

Actual DCF

Evaluation with test set

- Now, let's proceed with the analysis of the selected models' performances on the test set. Since even without dimensionality reduction, we got better results. So, we will stick to the approach and employ our best models.
- Once again, we will evaluate the models' performances in terms of minDCFs and actDCFs, providing a comprehensive understanding of their effectiveness.

Evaluation

minDCF

π	0.5	0.1	0.9
Tied Full-Covariance	0.119	0.303	0.331
LogisticReg(10^{-5} , 0.9)	0.123	0.340	0.298
SVM(C = 10)	0.130	0.361	0.313
GMM Tied Full Covariance(4)	0.062	0.188	0.197

actDCF

π	0.5	0.1	0.9
Tied Full-Covariance	0.123	0.307	0.344
LogisticReg(10^{-5} , 0.9)	0.125	0.348	0.314
SVM(C = 10)	0.130	0.366	0.320
GMM Tied Full Covariance(4)	0.063	0.197	0.212

Evaluation

- The results obtained on the test set are consistent with the findings from the previous analysis on the training set. In this case, GMM Tied-Full covariance (4-components) emerges as the best-performing model
- although the differences between the other models are minimal. GMM shows a significantly better performance which was anticipated during the analysis.
- To further compare the models, we can utilize ROC (Receiver Operating Characteristic) curves and DET curves. The models with the highest Area Under the Curve (AUC), for ROC, will be considered the best performers.

ROC and DET

- Although the curve looks somewhat identical for all classifiers, GMM is decently distinct and covering the most area, hence the best.
- The high slope on the left (For ROC) implies that models can correctly classify among the two classes