INDIAN INSTITUTE OF TECHNOLOGY DHARWAD

Lab 4

CS 412: Statistical Pattern Recognition

MEMBERS
GANESH SAMARTH - 170020030
MANDEEP BAWA - 170030038
S V PRAVEEN - 170010025

Link to Code - https://github.com/svp19/linear-classifiers

Solution 1

Comparative study on linear models to classify data points drawn from different gaussian distributions

I. Perceptron algorithm

The Perceptron algorithm is one of the initial algorithms developed to find a separating hyperplane when one exists in the data.

Mathematically, the algorithm tries to find a weight vector w^+ such that,

If y =1, $w^+x_i + w_0 > 0$

If y =-1, $w^+x_i + w_0 < 0$

The objective function the perceptron tries to optimize is :

$$\underset{\mathbf{w}}{\text{argmin}} \left\{ D(\mathbf{w}) = -\sum_{\mathbf{x_i} \in \mathcal{M}} y_i \cdot (\mathbf{w}^\mathsf{T}\mathbf{x}_i) \right\}$$

Differentiating the above function we obtain the weight perceptron rule for the perception algorithm.

We update the weights for all wrongly classified algorithms

$$w = w + y_i * x_i$$

Implementation Details:

Feature Augmentation:

We augment the train examples by one feature, to obtain a weight vector in d+1 dimension space and passes through the origin

Pocket algorithm :

In case of linearly non-separable data, we train the perceptron algorithm for a fixed number of iterations and pick the weight vector which resulted in the least train error

Feature Transformation:

Since most of the cases, the decision boundary is non-linear and the perceptron algorithm can only estimate linear decision boundaries, we enhance the features of the dataset by including polynomial features as well. This helps the linear classifiers to learn non-linear decision boundaries in the linear domain.

II. Linear Regression

Linear Regression algorithm tries to learn a classifier by fitting a line having least square error to the dataset.

Classifier

$$h(x) = W^T x$$

where, the algorithm tries to find a weight vector W^T such that,

$$\text{If } W^T x > 0, \ predict\ 1$$

$$\text{If} W^T x < 0, \ predict\ -1$$

where x is a vector after feature augmentation

Objective Function

We try to minimize the least square error between the classifier prediction and true labels

$$J(W) = ||XW - Y||_2$$

where X is $(x_1^T; x_2^T; ...; x_n^T)$and Y is $(y_1, y_2, ..., y_n)$

Finding W

We find optimal W by equating derivative J(W) w.r.t W to 0

$$W = (X^T X)^{-1} X^T Y$$

III. Logistic Regression

Logistic Regression tries to learn a linear classifier for the log of odds on the posterior probability.

Classifier

$$h(x) = 1/(1 + exp(- W^T x))$$

where, the algorithm tries to find a weight vector W^T such that,

If $W^T x > 0 \; => \; h(x) > 0.5, \; predict \; 1$

If$W^T x < 0, => h(x) < 0.5 \; predict \; 0$

where x is a vector after feature augmentation

Iterative Improvement

$$W^t = W^{(t-1)} - \eta * grad(J(W))$$

$$grad(J(W)) = X^T * h(XW - Y)$$

IV. Fisher Linear Discriminant Analysis

FLDA learns a linear discriminant by maximizing the inter-cluster distance between projected data and minimizing the intra-cluster distance between projected data.

Classifier

$$h(x) = W^T x + b$$

where, the algorithm tries to find a weight vector W^T such that,

If $W^T x + b > 0, \; predict \; 1$

If$W^T x + b < 0, \; predict \; -1$

Objective Function

We try to minimize the following function

$$J(W) = \frac{w^t s_b w}{w^t s_w w}$$

Where,

$$s_b = (M_1 - M_0)(M_1 - M_0)^T$$

$$s_w = \sum_{x_i \in C_0} (x_i - M_0)(x_i - M_0)^T + \sum_{x_i \in C_1} (x_i - M_1)(x_i - M_1)^T$$

$$M_0 = \frac{1}{n_0} \sum_{x_i \in C_0} x_i$$

$$M_0 = \frac{1}{n_1} \sum_{x_i \in C_1} x_i$$

Where, n0 and n1 are the number of points in class 0 and class 1 respectively.

Finding W and b

After finding W, we find b by line search.

We find optimal W by equating derivative J(W) w.r.t W to 0

$$W = s_w^{-1}(M_1 - M_0)$$

1. Synthetic Dataset

Sampled 2000 train points and 1000 test points from normal distribution using following parameters -

Parameters for 10-D

- **multivariate 10-D**
  - Means

    $\mu_1 = 1, \ \mu_1 = 0$

  - Covariances

    Σ = I

  - Priors

    $\lambda_1 = 0.5, \lambda_0 = 0.5$

Results

Sl No.	**Implementation**	**Model**	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.922	0.932	0.9205	0.933
2.	**Ours**	Linear Regression	0.9455	0.9450	0.945	0.943
3.	**Ours**	Logistic Regression	0.945	0.95	0.9452	0.95
4.	**Ours**	FLDA	0.7575	0.742	0.75542	0.73939
5.	Sklearn	Perceptron	0.9208	0.9265	0.913	0.923

6.	Sklearn	Linear Regression	0.9455	0.9450	0.945	0.943
7.	Sklearn	Logistic Regression	0.9465	0.95	0.9465	0.95
8.	Sklearn	LDA	0.947	0.939	0.9469	0.9390

Confusion Matrix for Perceptron

475	43
25	457

Confusion Matrix for Linear Regression

475	25
40	460

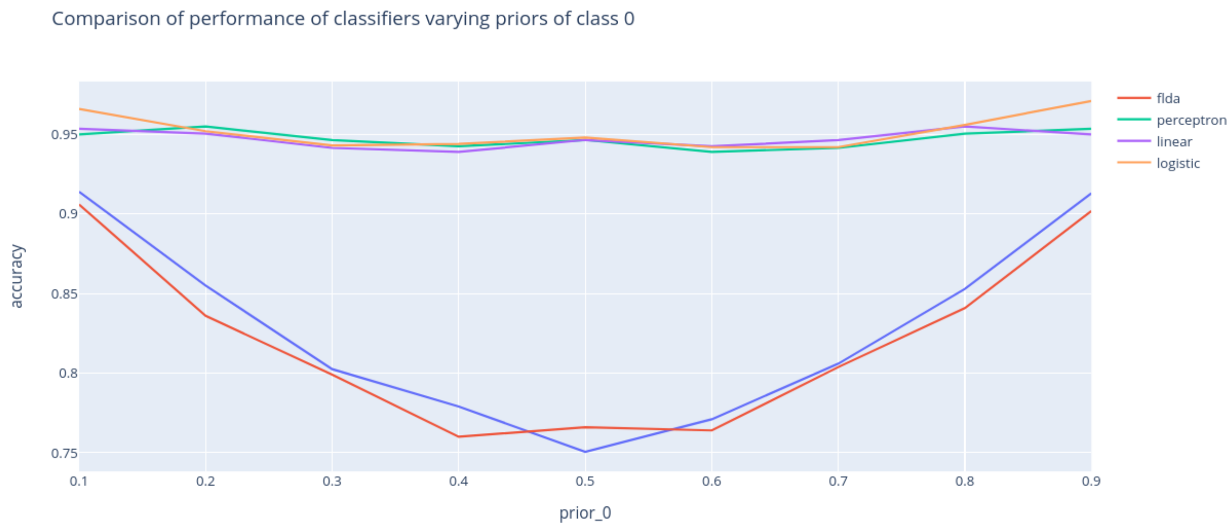
Confusion Matrix for Logistic Regression

474	26
24	476

Confusion Matrix for FLDA

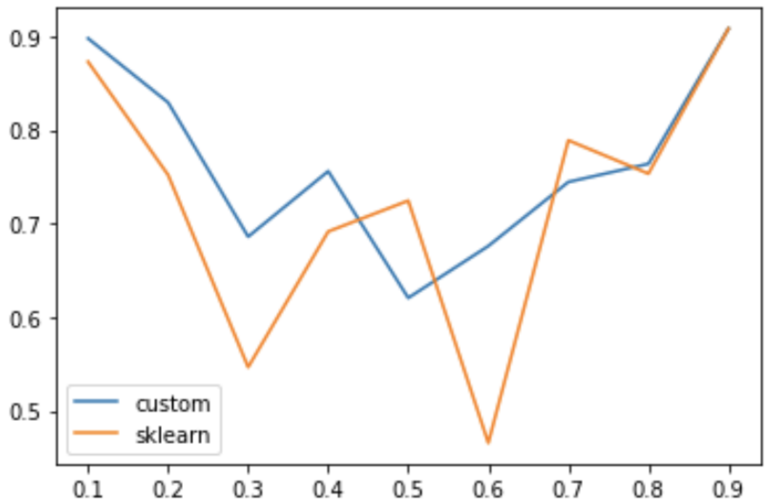
376	124
134	366

Varying Priors of class 0

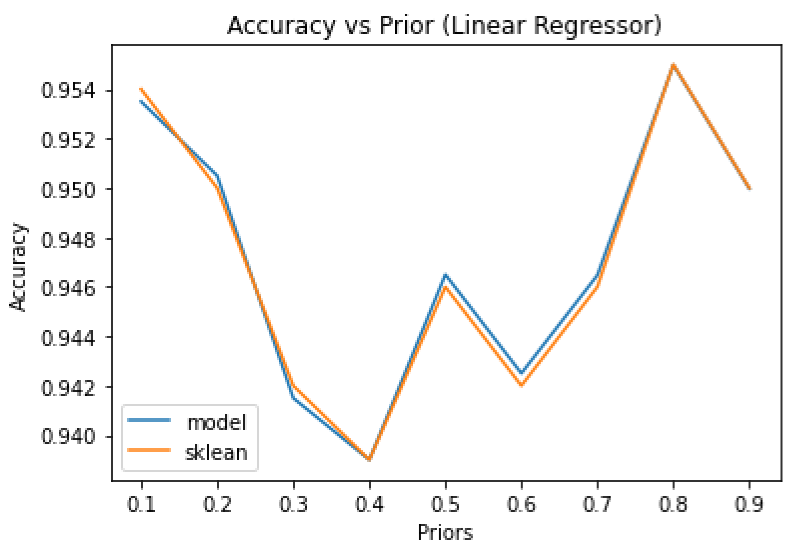


Varying Priors with sklearn

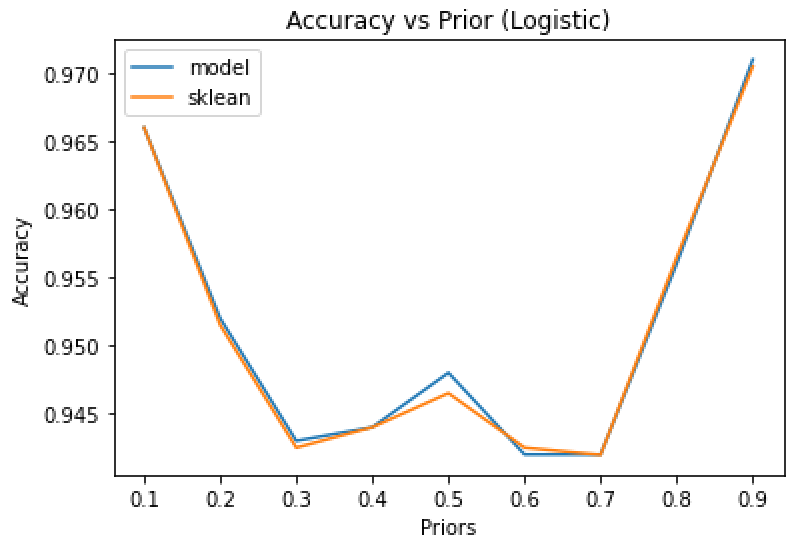
I. Perceptron



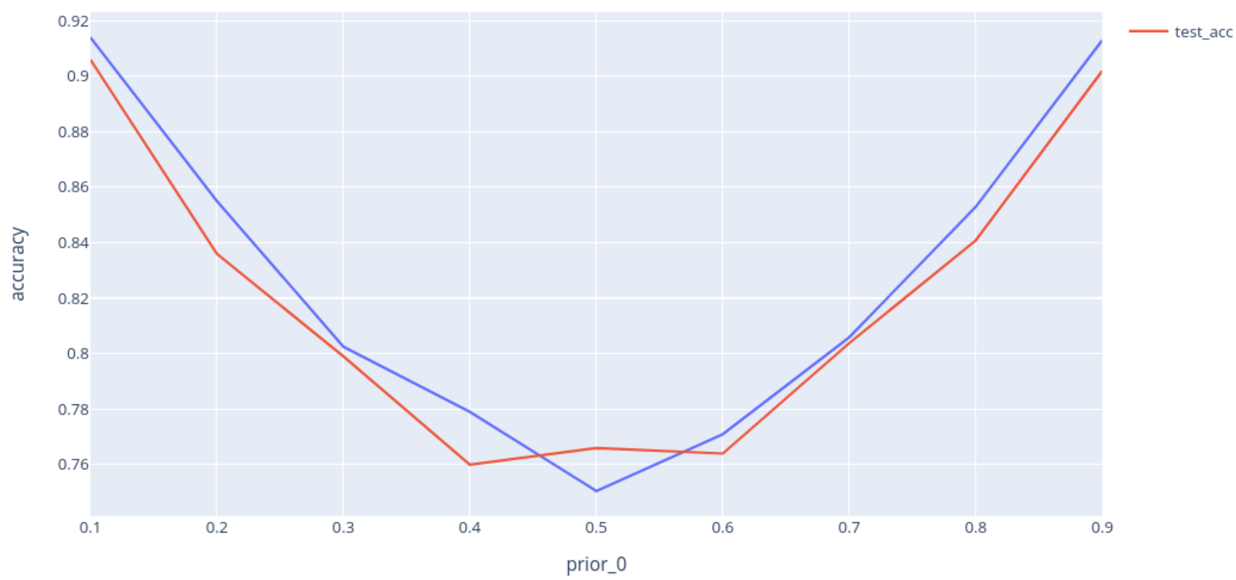
II. Linear



III. Logistic



IV. FLDA



Parameters for Different Means, Same Covariances

- **multivariate 2-D**
  - Means

    $\mu_1 = 1, \ \mu_1 = 0$

  - Covariances

$$\mathbf{\Sigma = I}$$

- Priors

$$\lambda_1 = 0.5,\ \lambda_0 = 0.5$$

Results

Sl No.	Implementation	Model	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.7375	0.75	0.71	0.735
2.	**Ours**	Linear Regression	0.763	0.788	0.762	0.79
3.	**Ours**	Logistic Regression	0.7635	0.759	0.762	0.758
4.	**Ours**	FLDA	0.787	0.775	0.82626	0.81542

Confusion Matrix for Perceptron

402	152
98	348

Confusion Matrix for Linear Regression

389	111
101	399

Confusion Matrix for Logistic Regression

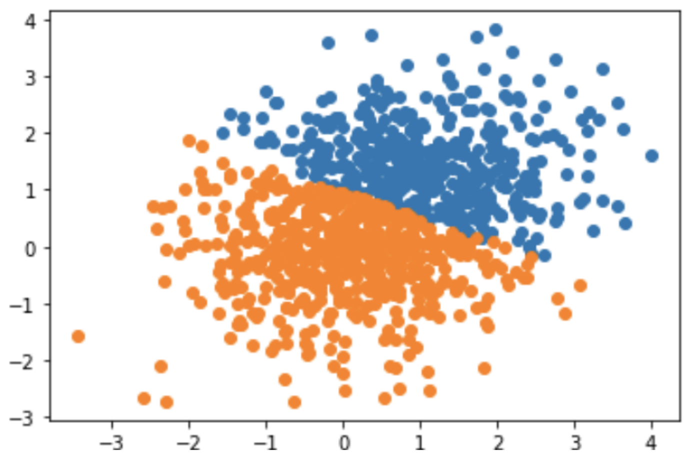
380	120
121	379

Confusion Matrix for FLDA

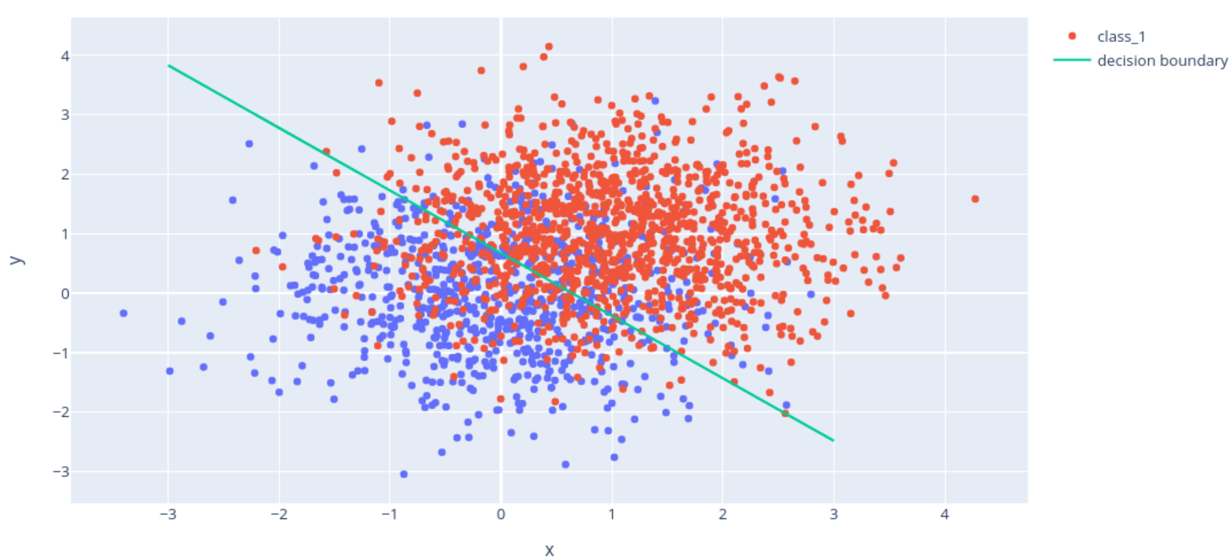
278	122
103	497

Decision Boundaries

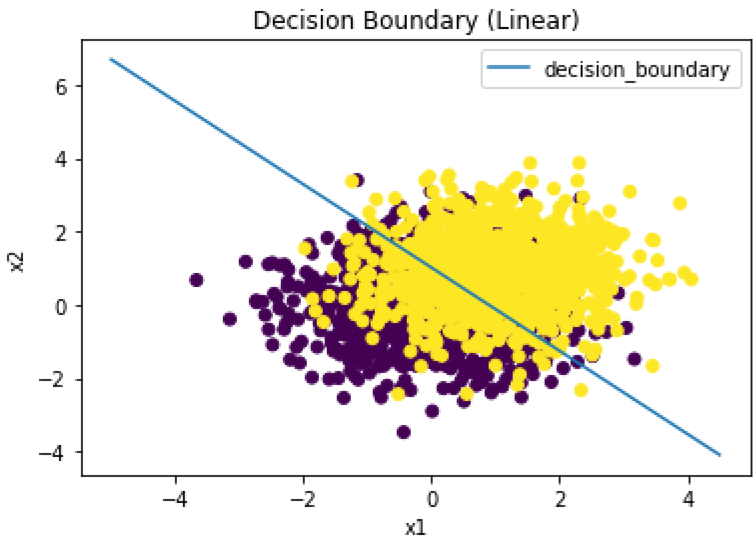
Perceptron



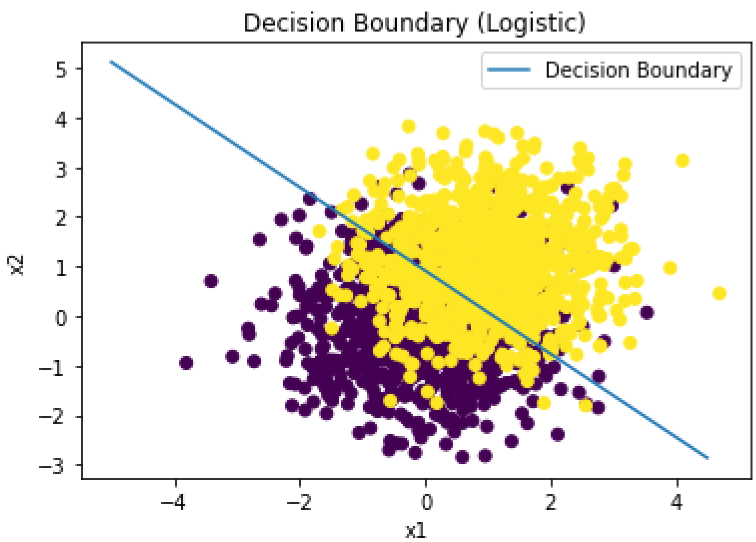
FLDA



Linear



Logistic



Parameters for Same Means, Different Covariances

- **multivariate 2-D**
  - Means

    $\mu_1 = 0, \ \mu_1 = 0$

  - Covariances

    Σ_0 = I, Σ_1 = [[1, 0.9], [0.9, 1]]

- Priors

$$\lambda_1 = 0.5,\ \lambda_0 = 0.5$$

Results

Sl No.	Implementation	Model	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.498	0.537	0.44	0.48
2.	**Ours**	Linear Regression	0.535	0.514	0.542	0.514
3.	**Ours**	Logistic Regression	0.508	0.513	0.513	0.523
4.	**Ours**	FLDA	0.6395	0.613	0.72596	0.70615

Confusion Matrix for Perceptron

322	285
178	215

Confusion Matrix for Linear Regression

256	244
242	258

Confusion Matrix for Logistic Regression

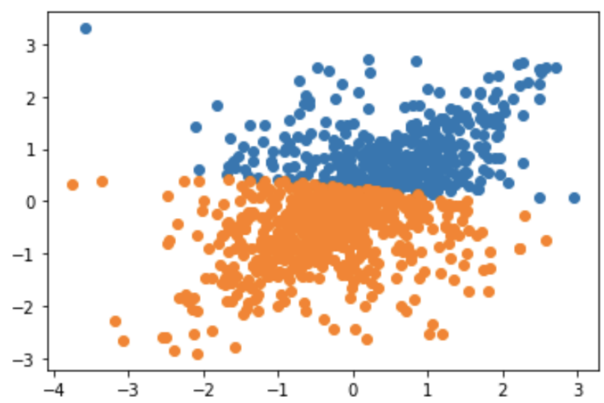
246	254
233	267

Confusion Matrix

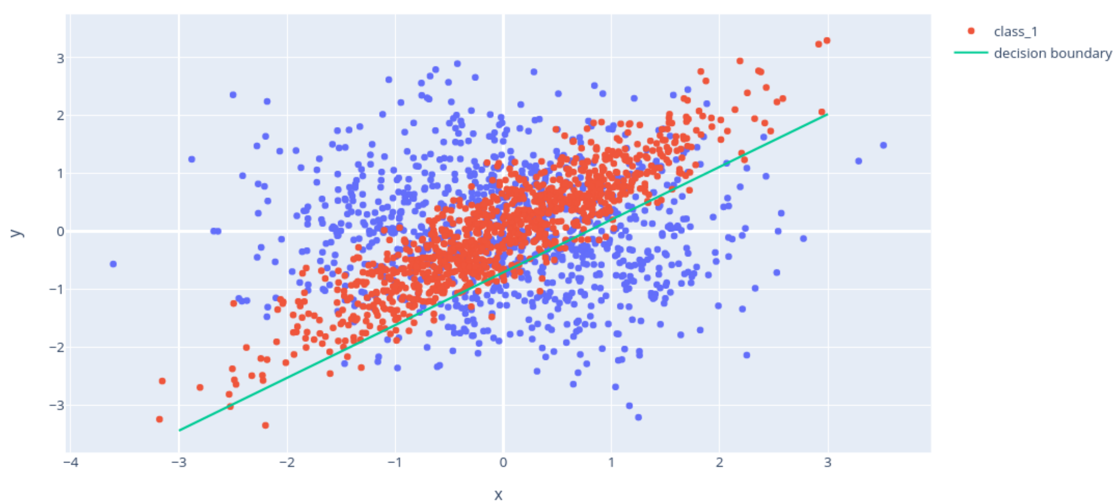
148	352
35	465

Decision Boundaries

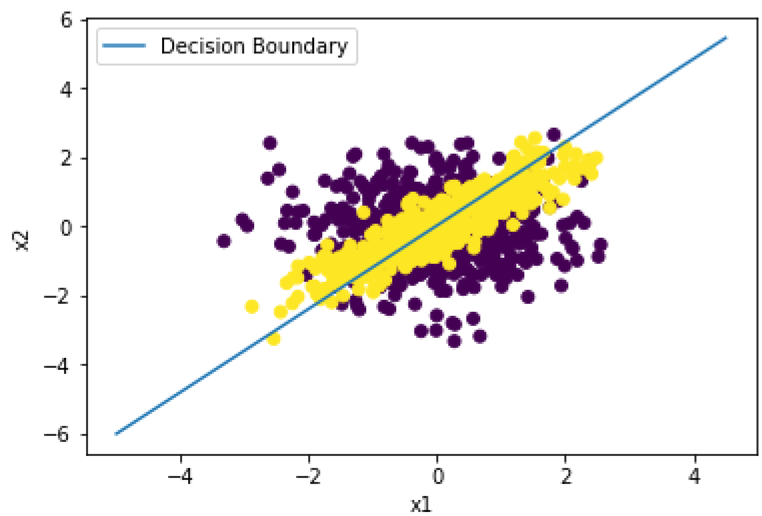
Perceptron



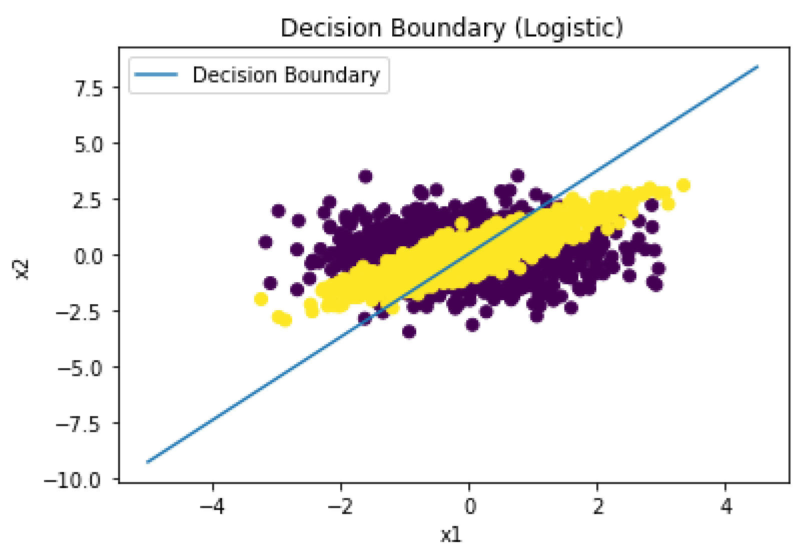
FLDA



Linear



Logistic



Using a polynomial transformation of degree=2

Results

Sl No.	Implementation	Model	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)

1.	**Ours**	Perceptron	0.706	0.71	0.733	0.737
2.	**Ours**	Linear Regression	0.741	0.743	0.785	0.787
3.	**Ours**	Logistic Regression	0.757	0.787	0.787	0.809
4.	**Ours**	FLDA	0.7395	0.744	0.77933	0.77854

Confusion Matrix for Perceptron

298	90
202	410

Confusion Matrix for Linear Regression

267	233
24	476

Confusion Matrix for Logistic Regression

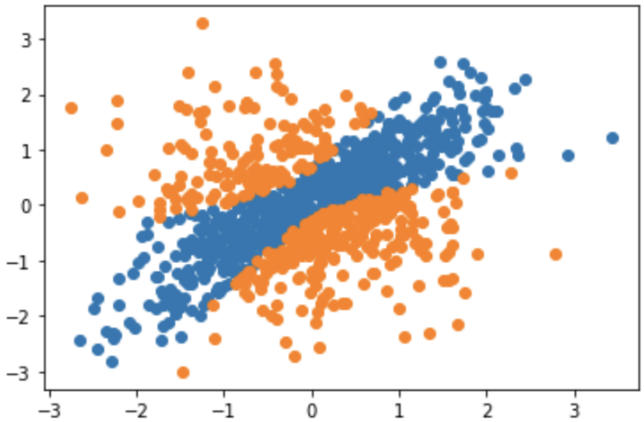
335	165
48	452

Confusion Matrix for FLDA

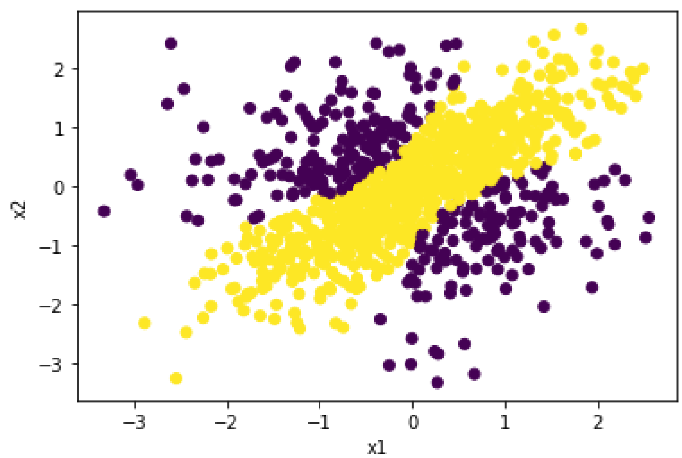
294	206
50	450

Decision Boundaries

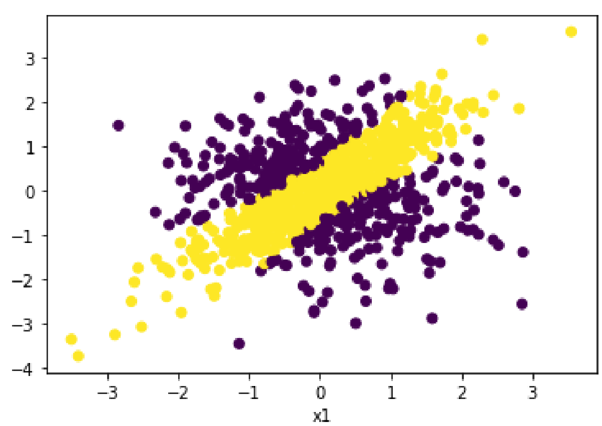
Perceptron



Linear



Logistic



Parameters for Different Means, Different Covariances

Using 4000 train points and 2000 test points

- **multivariate 2-D**
  - Means

    $\mu_1 = [3, 6], \ \mu_1 = [3, -2]$

  - Covariances

    Σ_0 = [[0.5, 0], [0, 2]]

    Σ_1 = [[2, 0], [0, 2]]

  - Priors

    $\lambda_1 = 0.5, \lambda_0 = 0.5$

Results

Sl No.	**Implementation**	**Model**	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.945	0.995	0.9945	0.995
2.	**Ours**	Linear Regression	0.9962	0.9960	0.996	0.995
3.	**Ours**	Logistic Regression	0.9962	0.9965	0.9962	0.9964
4.	**Ours**	FLDA	0.9985	0.995	0.99849	0.99849

Confusion Matrix for Perceptron

496	1
4	499

Confusion Matrix for Linear Regression

997	3
5	995

Confusion Matrix for Logistic Regression

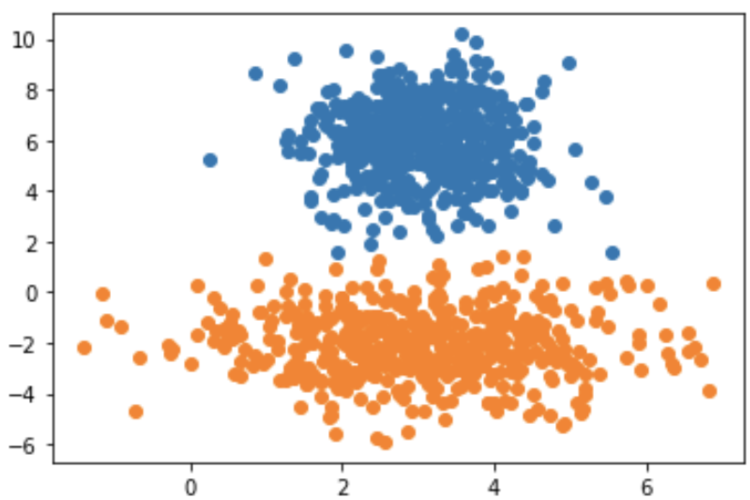
998	2
5	995

Confusion Matrix for FLDA

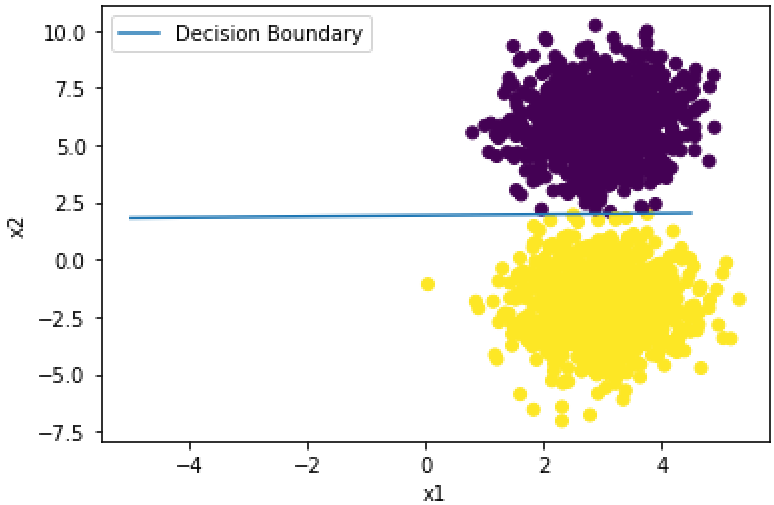
998	2
8	992

Decision Boundaries

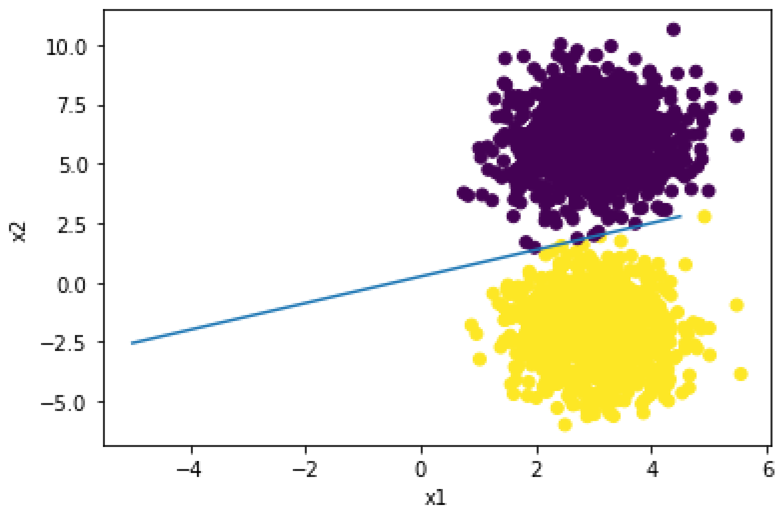
Perceptron



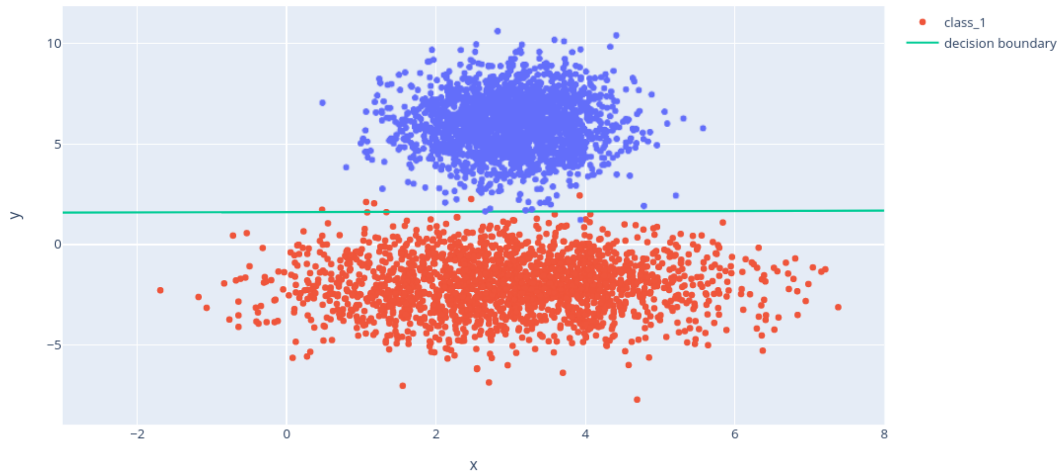
Linear



Logistic



FLDA


class_1
decision boundary
10
5
0
−5
−2
0
2
4
6
8
x
y

2. German Credit Data Set

Results for 80:20 Split

Sl No.	Implementation	Model	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.74	0.73	0.82	0.793
2.	**Ours**	Linear Regression	0.788	0.770	0.588	0.566
3.	**Ours**	Logistic Regression	0.7	0.7	0.822	0.822
4.	**Ours**	FLDA	0.7825	0.765	0.85402	0.83392

Confusion Matrix for Perceptron

42	26
28	104

Confusion Matrix for Linear Regression

124	16
30	30

Confusion Matrix for Logistic Regression

140	0
60	0

Confusion Matrix for FLDA

35	33
14	118

Results for 70:30 Split

Sl No.	**Implementation**	**Model**	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.72	0.74	0.79	0.799
2.	**Ours**	Linear Regression	0.781	0.766	0.553	0.562
3.	**Ours**	Logistic Regression	0.705	0.686	0.824	0.814
4.	**Ours**	FLDA	0.8942	0.6966	0.92494	0.77193

Confusion Matrix for Perceptron

67	52
26	155

Confusion Matrix for Linear Regression

185	17
53	45

Confusion Matrix for Logistic Regression

206	0
94	0

Confusion Matrix for FLDA

55	37
54	154

3. Porto Seguro's Safe Driver Prediction

There is a huge imbalance in the dataset with 573,518 examples belonging to class 0 and 21,694 examples in class 1. We undersample the data from the leading class and sample 21,694 examples from each class to obtain a dataset of ((43388, 57)) feature vectors and (43388) labels.

Results for 80:20 Split

Sl No.	Implementation	Model	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.556	0.56	0.6468	0.64
2.	**Ours**	Linear Regression	0.5737	0.5733	0.5737	0.5733
3.	**Ours**	Logistic Regression	0.574	0.582	0.538	0.55
4.	**Ours**	FLDA	0.59034	0.59899	0.60294	0.60757
5.	**Sklearn**	SVM	0.70803	0.58527	0.70183	0.57574
6.	**Sklearn**	MLPClassifier	0.73298	0.55623	0.73624	0.55862

Confusion Matrix for Perceptron

1335	851
2965	3527

Confusion Matrix for Linear Regression

2734	1586
1969	2389

Confusion Matrix for Logistic Regression

2834	1461
2162	2221

Confusion Matrix for FLDA

2504	1871
1609	2694

Results for 70:30 Split

Sl No.	**Implementation**	**Model**	Train Accuracy	Test Accuracy	F-1 Score (train)	F-1 Score (test)
1.	**Ours**	Perceptron	0.512	0.51	0.667	0.66
2.	**Ours**	Linear Regression	0.591	0.588	0.58	0.571
3.	**Ours**	Logistic Regression	0.427	0.438	0.477	0.485
4.	**Ours**	FLDA	0.59224	0.58661	0.5746	0.57024
5.	**Sklearn**	SVM	0.71463	0.584466	0.70333	0.57197
6.	**Sklearn**	MLPClassifier	0.74406	0.5525	0.74318	0.55456

Confusion Matrix for Perceptron

414	297
6080	6226

Confusion Matrix for Linear Regression

4082	2502
2860	3573

Confusion Matrix for Logistic Regression

2250	4358
2954	3455

Confusion Matrix for FLDA

4066	2416
2965	3570

Conclusion

In this report, we explored four classifiers - Perceptron (Pocket Algorithm), Linear Regression, Logistic Regression and Fisher's Linear Discriminant Analysis. Linear regression has less time complexity and is easy to implement followed by FLDA. Perceptron and Logistic Regression are iterative algorithms and hence, comparatively slower.

All the models are able to learn reasonably good decision boundaries. In terms of consistency, Logistic Regression is the most consistent across datasets followed by linear regression, perceptron and FLDA in the order respectively.

While varying prior probabilities, we see that linear regression, logistic regression and perceptron perform well, however, FLDA is the most sensitive and performs poorly when the priors are equal. In contrast, in the case of same means and different covariances, we see that FLDA is able to outperform all other classifiers. By transforming the features, we enable our models to learn non-linear boundaries while classifying the data and can see up to a ~9% accuracy boost with this polynomial transformation.

Finally, in case of linearly separable data, all algorithms perform equally well to learn a decision boundary to separate both classes.