Linear Models

170010002 Ameya Vadnere

170010026 Deepak H R

https://colab.research.google.com/drive/1pkBlHJNg-_dBfQRgImLnXwEx9lvwvL3b?authuser=1#scrollTo=V1VHYW8qEygM

Introduction

One of the most basic approaches towards learning a classifier is using Linear Models. Linear models for classification separate the input vectors (data points) into various classes using linear decision boundaries. In this assignment, we study and implement four different types of classifiers based on Linear Models.

POCKET Algorithm (Modified Perceptron Algorithm)

The Perceptron learning algorithm learns a decision boundary given some linearly separable data. In case of non separable data, a modified version of the algorithm is used known as the POCKET Algorithm.

Algorithm: Perceptron Learning Algorithm

$P \leftarrow inputs \quad with \quad label \quad 1$;
$N \leftarrow inputs \quad with \quad label \quad 0$;
Initialize $\mathbf{w}$ randomly;
while $!convergence$ **do**
  Pick random $\mathbf{x} \in P \cup N$;
  if $\mathbf{x} \in P \quad and \quad \mathbf{w}.\mathbf{x} < 0$ **then**
    $\mathbf{w} = \mathbf{w} + \mathbf{x}$;
  end
  if $\mathbf{x} \in N \quad and \quad \mathbf{w}.\mathbf{x} \geq 0$ **then**
    $\mathbf{w} = \mathbf{w} - \mathbf{x}$;
  end
end
//the algorithm converges when all the inputs are classified correctly

In POCKET algorithm, we randomly initialize the separating hyperplane parameters and keep updating them, storing the best parameters found till each iteration in a 'pocket'.

Hypothesis function

$$h_w(x_i) = sgn(w^T x_i + w_0)$$

Loss function

$$J(w) = \sum_{i=1}^{n} max(-y_i h_w(x_i), 0)$$

The class labels are assumed to be {-1, 1} in this case.

Linear Least Squares

In this algorithm, we try to minimize the total squared error. We can augment the input vectors, (i.e append 1 to each input vector) for ease of computation.

Hypothesis function

$$\begin{aligned} h_w(x_i) &= 1 \quad w^T x_i + w_0 > t \\ &= 0 \quad otherwise \end{aligned}$$

Where ***t*** is some threshold value.

Loss function

$$J(w) = \frac{1}{n} \sum_{i=1}^{n} (w^T x_i + w_0 - y_i)^2$$

The solution to **W** is given by:

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

Where both **X** and **W** are augmented.

Shape of **W: (**d+1, 1) and Shape of **X:** (n, d+1)

Logistic Regression

Hypothesis function

$$\begin{aligned} h_w(x_i) &= 1 \quad \sigma(w^T x_i + w_0) > t \\ &= 0 \quad otherwise \end{aligned}$$

where:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Loss function

$$J(w) = -\frac{1}{n} \sum_{i=1}^{n} y_i log(a(x_i)) + (1 - y_i) log(1 - a(x_i))$$

Where:

$$a(x) = \sigma(w^T x + w_0)$$

Q1: Classification on Synthetic dataset

We observed the performance of various algorithms for different prior probabilities. We chose 3 different prior probabilities for class 0: 0.05 (extreme imbalance), 0.3 (moderate imbalance), 0.5 (no imbalance). The metrics we use for comparing the algorithms are accuracy and f1-score.

Terminology:

- acc: test accuracy of our implemented classifier
- Train_f1, Train_acc: train f1 score and train accuracy
- sk_acc: Accuracy of sklearn classifier
- f1 : f1 score of our classifier, sk_f1: f1 score of sklearn classifier

10-dimensional data

We generated 10-dimensional 2000 training examples and 1000 testing examples for both classes. The mean of the data was 0 across all dimensions for class 0 and 1 across all dimensions for class 1. The covariance matrices were identity for both classes.

Performance of each algorithm was as follows:

Bayes Decision Boundary

prior	bayes_acc	bayes_f1	weights
0.05	0.981	0.990047	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, -2.06]
0.30	0.938	0.955903	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, -4.15]
0.50	0.933	0.933860	[1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, -5.0]

POCKET algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.978	0.975	0.969	0.989	0.987	0.984	[0.32, 0.14, 0.24, 0.22, 0.04, 0.46, 0.4, 0.25, 0.27, 0.21, -0.02]	[3.68, 1.03, 3.82, 4.99, 4.63, 1.52, -0.73, 1.06, 3.97, 1.83, -4.0]
0.30	0.952	0.936	0.916	0.967	0.955	0.939	[0.45, 0.6, 0.52, 0.47, 0.22, 0.41, 0.32, 0.48, 0.47, 0.24, -1.5]	[3.94, 5.26, 3.42, 2.8, 4.71, 7.33, 4.28, 3.37, 0.1, 2.5, -18.0]
0.50	0.941	0.916	0.922	0.941	0.917	0.924	[0.44, 0.25, 0.41, 0.52, 0.31, 0.62, 0.25, 0.4, 0.43, 0.22, -1.89]	[3.27, 5.4, 5.19, 1.47, 3.65, 5.01, 2.72, 7.5, 4.43, 6.92, -20.0]

LeastSquares algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.982	0.979	0.954	0.990	0.989	0.976	[0.03, 0.03, 0.03, 0.04, 0.03, 0.04, 0.03, 0.04, 0.04, 0.03, 0.64]	[0.03, 0.03, 0.03, 0.04, 0.03, 0.04, 0.03, 0.04, 0.04, 0.03, 0.64]
0.30	0.957	0.942	0.937	0.969	0.958	0.955	[0.06, 0.07, 0.08, 0.07, 0.06, 0.06, 0.08, 0.07, 0.07, 0.06, 0.22]	[0.06, 0.07, 0.08, 0.07, 0.06, 0.06, 0.08, 0.07, 0.07, 0.06, 0.22]
0.50	0.945	0.935	0.929	0.945	0.935	0.930	[0.09, 0.07, 0.06, 0.06, 0.06, 0.08, 0.06, 0.08, 0.09, 0.07, 0.14]	[0.09, 0.07, 0.06, 0.06, 0.06, 0.08, 0.06, 0.08, 0.09, 0.07, 0.14]

LogisticRegression algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.982	0.978	0.978	0.991	0.988	0.988	[1.01, 0.63, 0.75, 1.08, 0.9, 0.77, 0.75, 1.2, 1.14, 1.06, -1.93]	[0.96, 0.61, 0.73, 1.03, 0.86, 0.75, 0.73, 1.13, 1.08, 1.0, -1.8]
0.30	0.955	0.936	0.936	0.968	0.955	0.955	[1.06, 1.04, 1.07, 1.12, 0.88, 1.14, 1.06, 1.18, 1.21, 1.01, -4.57]	[1.03, 1.01, 1.04, 1.09, 0.86, 1.1, 1.03, 1.14, 1.17, 0.98, -4.43]
0.50	0.938	0.930	0.931	0.938	0.931	0.931	[1.23, 0.97, 0.97, 0.82, 0.91, 1.15, 0.81, 1.13, 1.26, 1.01, -5.32]	[1.19, 0.95, 0.95, 0.81, 0.89, 1.12, 0.8, 1.11, 1.22, 0.98, -5.19]

FisherLDA algorithm

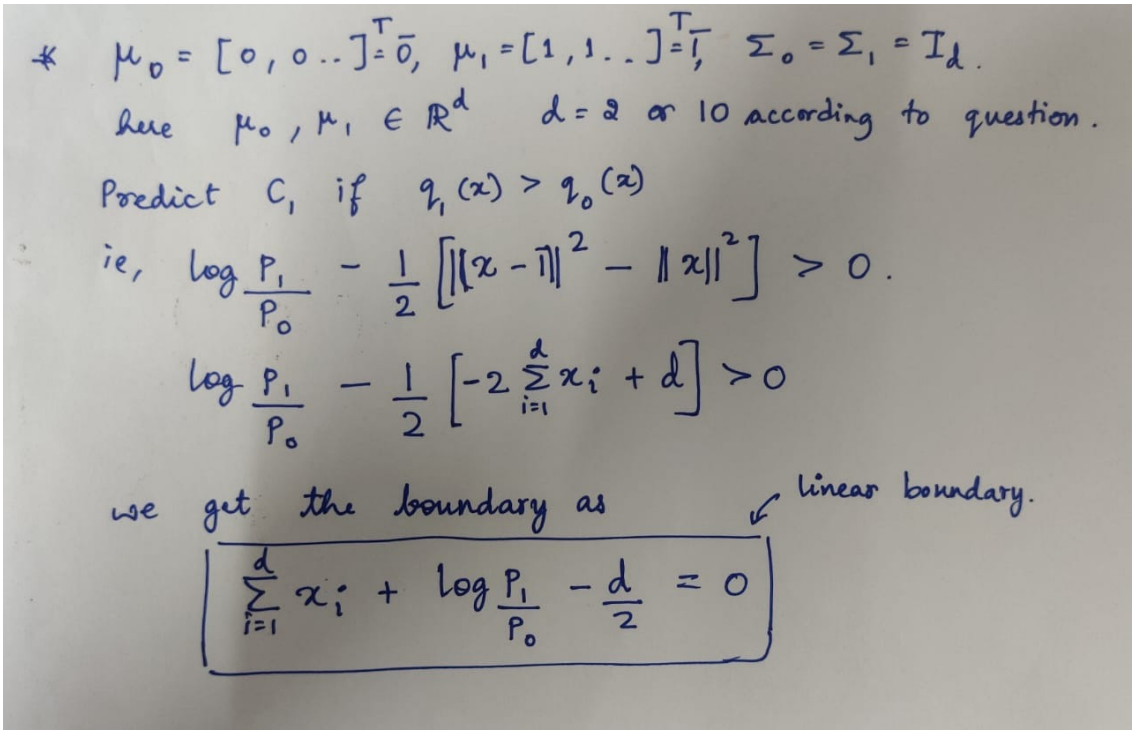
prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.982	0.976	0.979	0.991	0.987	0.989	[0.94, 0.81, 0.8, 1.24, 1.03, 1.11, 0.87, 1.26, 1.14, 0.92, -2.6]	[0.94, 0.81, 0.8, 1.24, 1.03, 1.11, 0.87, 1.26, 1.14, 0.92, -2.18]
0.30	0.958	0.940	0.935	0.969	0.957	0.954	[0.93, 1.06, 1.17, 1.03, 0.95, 0.94, 1.2, 1.13, 1.07, 0.97, -4.86]	[0.93, 1.06, 1.17, 1.03, 0.95, 0.94, 1.2, 1.12, 1.07, 0.97, -4.41]
0.50	0.946	0.928	0.929	0.945	0.927	0.930	[1.22, 1.01, 0.89, 0.91, 0.85, 1.17, 0.87, 1.1, 1.22, 0.95, -5.8]	[1.22, 1.01, 0.89, 0.91, 0.85, 1.17, 0.87, 1.1, 1.22, 0.95, -5.18]

2-dimensional data

Different mean, same identity covariance

In this case, we generate the 2-dimensional data with the same no. of train and test data points with identity covariance matrices.

Bayes classifier



Without feature transformation:

Bayes Decision Boundary

prior	bayes_acc	bayes_f1	weights
0.05	0.954	0.976337	[1.0, 1.0, 1.94]
0.30	0.820	0.877049	[1.0, 1.0, -0.15]
0.50	0.763	0.762763	[1.0, 1.0, -1.0]

POCKET algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.954	0.954	0.951	0.976	0.976	0.974	[0.19, 0.24, 0.39]	[1.45, 0.65, 1.0]
0.30	0.799	0.809	0.317	0.868	0.874	0.047	[0.19, 0.21, 0.05]	[0.55, 0.62, -3.0]
0.50	0.757	0.763	0.711	0.760	0.764	0.762	[0.07, 0.07, -0.07]	[0.86, 0.5, 0.0]

LeastSquares algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.904	0.923	0.950	0.948	0.959	0.974	[0.04, 0.05, 0.87]	[0.04, 0.05, 0.87]
0.30	0.792	0.813	0.811	0.859	0.874	0.873	[0.15, 0.15, 0.5]	[0.15, 0.15, 0.5]
0.50	0.759	0.770	0.767	0.764	0.775	0.764	[0.17, 0.16, 0.32]	[0.17, 0.16, 0.32]

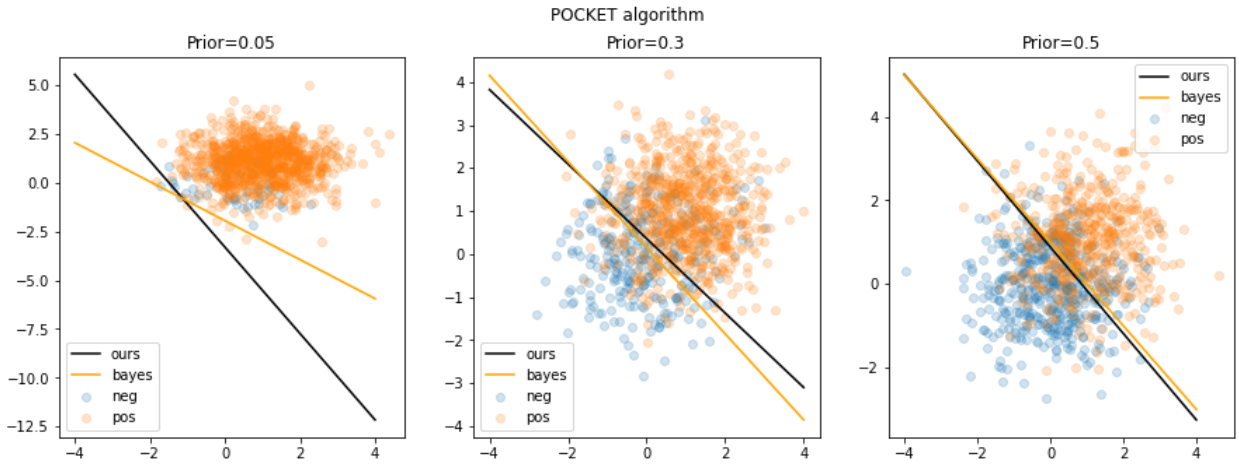
LogisticRegression algorithm

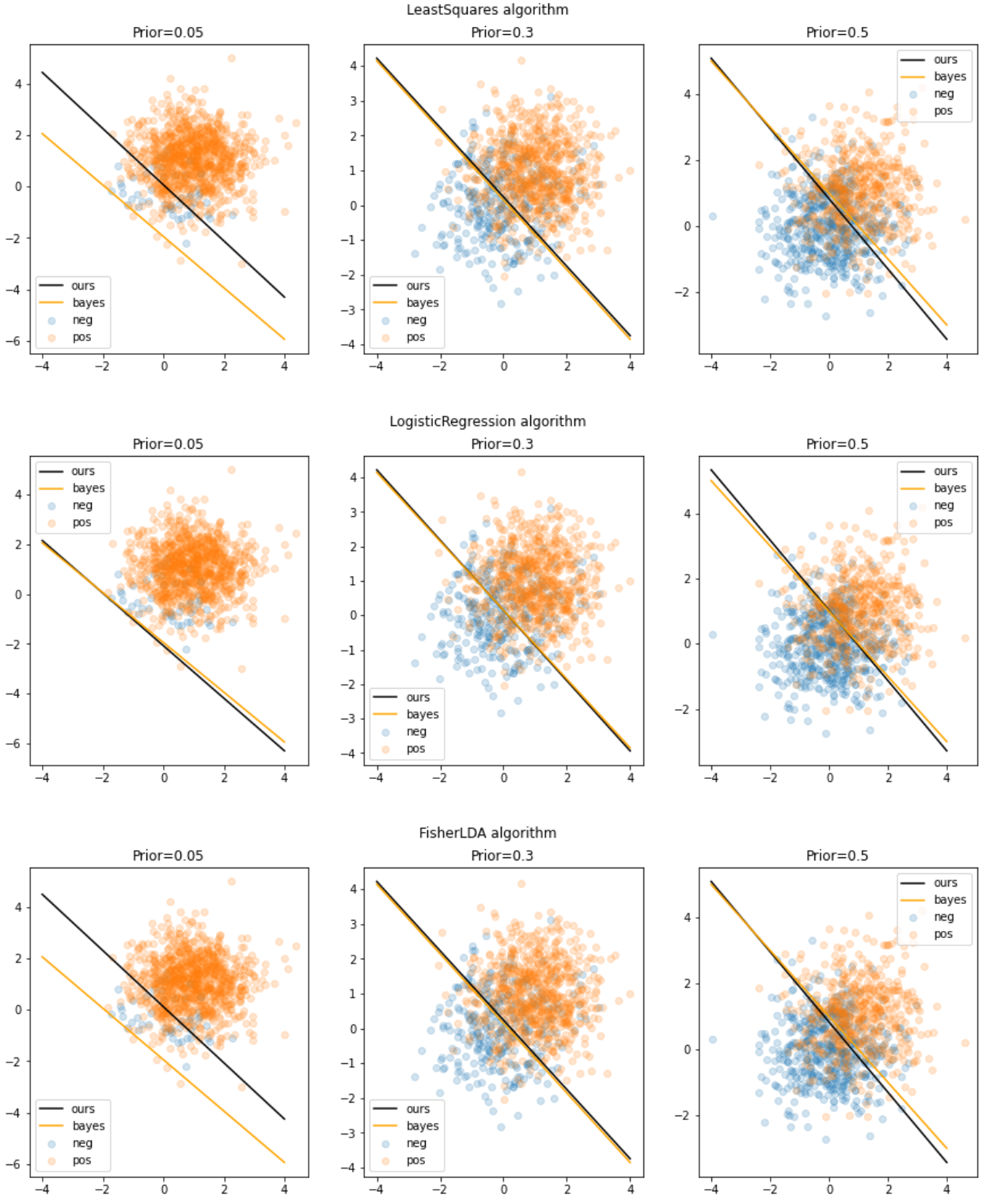
prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.952	0.954	0.954	0.975	0.976	0.976	[1.01, 1.05, 2.0]	[1.0, 1.04, 2.0]
0.30	0.789	0.818	0.817	0.857	0.877	0.876	[0.99, 0.98, -0.11]	[0.98, 0.97, -0.11]
0.50	0.755	0.770	0.770	0.753	0.768	0.768	[1.05, 0.98, -1.06]	[1.04, 0.97, -1.05]

FisherLDA algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.909	0.922	0.954	0.951	0.958	0.976	[0.99, 1.06, 0.23]	[0.99, 1.06, 1.99]
0.30	0.792	0.813	0.818	0.859	0.874	0.876	[0.98, 0.99, -0.05]	[0.98, 0.99, -0.12]
0.50	0.760	0.770	0.767	0.765	0.775	0.764	[1.04, 0.97, -0.94]	[1.04, 0.97, -1.06]

We plotted the decision boundaries for the algorithms:




LeastSquares algorithm
Prior=0.05
Prior=0.3
Prior=0.5
LogisticRegression algorithm
Prior=0.05
Prior=0.3
Prior=0.5
FisherLDA algorithm
Prior=0.05
Prior=0.3
Prior=0.5
ours
bayes
neg
pos

Applying Feature Transformation

We transformed the input data features into a polynomial with degree=2, hoping to improve the performance. That is, initially we had (x_1, x_2). Now, after feature transformation, we have non-linear features: (1, x_1, x_2, x_1x_2, x_1^2, x_2^2).

The results are as follows:

POCKET algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.953	0.951	0.950	0.976	0.975	0.974	[0.73, 0.43, 0.54, 0.56, -0.2, 0.11, -0.19]	[1.0, 1.83, 5.83, 2.99, -3.37, 11.0, 1.0]
0.30	0.792	0.801	0.518	0.857	0.865	0.536	[-0.17, 0.45, 0.43, 0.01, -0.06, -0.09, 0.1]	[-1.0, -2.89, 3.34, -4.79, 3.52, 1.62, -1.0]
0.50	0.778	0.759	0.756	0.780	0.761	0.752	[-0.25, 0.34, 0.43, 0.13, -0.0, -0.04, -0.17]	[-2.0, 2.56, 4.69, 1.41, 3.83, -0.2, -2.0]

LeastSquares algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.936	0.940	0.953	0.966	0.968	0.976	[0.42, 0.12, 0.11, -0.02, -0.04, -0.02, 0.42]	[0.0, 0.12, 0.11, -0.02, -0.04, -0.02, 0.85]
0.30	0.800	0.798	0.800	0.862	0.862	0.867	[0.25, 0.21, 0.18, -0.01, -0.04, -0.01, 0.25]	[0.0, 0.21, 0.18, -0.01, -0.04, -0.01, 0.49]
0.50	0.781	0.764	0.763	0.778	0.761	0.762	[0.16, 0.17, 0.17, -0.0, -0.0, 0.01, 0.16]	[0.0, 0.17, 0.17, -0.0, -0.0, 0.01, 0.33]

LogisticRegression algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.953	0.952	0.952	0.976	0.975	0.975	[1.02, 1.12, 1.12, -0.14, -0.04, 0.06, 1.02]	[-0.0, 1.1, 1.1, -0.14, -0.03, 0.07, 2.04]
0.30	0.798	0.798	0.798	0.860	0.862	0.862	[-0.15, 1.01, 0.9, 0.08, 0.04, 0.11, -0.15]	[-0.0, 1.01, 0.89, 0.08, 0.05, 0.11, -0.3]
0.50	0.777	0.763	0.763	0.777	0.763	0.763	[-0.55, 1.11, 1.08, -0.01, -0.05, 0.04, -0.55]	[-0.0, 1.1, 1.07, -0.0, -0.04, 0.04, -1.09]

FisherLDA algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.943	0.946	0.946	0.970	0.972	0.972	[0.0, 2.94, 2.75, -0.56, -0.93, -0.42, 2.3]	[0.0, 2.95, 2.76, -0.56, -0.93, -0.43, 2.31]
0.30	0.800	0.798	0.797	0.862	0.862	0.864	[0.0, 1.42, 1.23, -0.1, -0.28, -0.04, -0.38]	[0.0, 1.42, 1.23, -0.1, -0.28, -0.04, -0.16]
0.50	0.782	0.758	0.763	0.777	0.754	0.762	[0.0, 1.05, 1.06, -0.0, -0.02, 0.03, -1.17]	[0.0, 1.05, 1.06, -0.0, -0.02, 0.03, -1.07]

As can be observed from the tables, applying feature transformation did not improve test accuracy/f1 score much. This is because the Bayes decision boundary itself is linear.

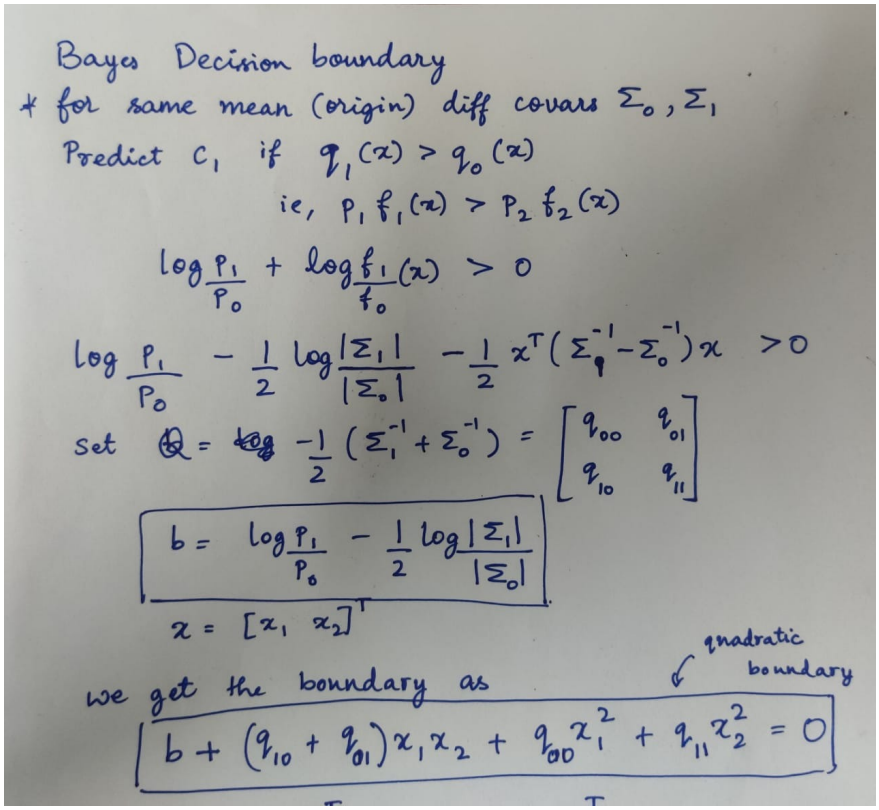
Also, adding complexity to the model has increased train_acc and train_f1, which is an example of a small amount of overfitting.

Same means, different covariance1

Here, we used mu_0 = mu_1 = $[0, 0]^T$, sigma_0 = I and sigma_1= [(1, .9), (.9, 1)] to generate data

Bayes classifier

We notice that the bayes classifier has quadratic terms in its decision boundary.



Without Feature transformation

POCKET algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.960	0.961	0.947	0.979	0.980	0.973	[0.2, -0.21, 0.3]	[0.39, 0.01, 1.0]
0.30	0.766	0.750	0.712	0.856	0.846	0.829	[0.19, -0.19, 0.2]	[0.35, -0.46, 1.0]
0.50	0.642	0.610	0.505	0.727	0.707	0.024	[-0.12, 0.12, 0.08]	[0.59, 0.36, -2.0]

LeastSquares algorithm

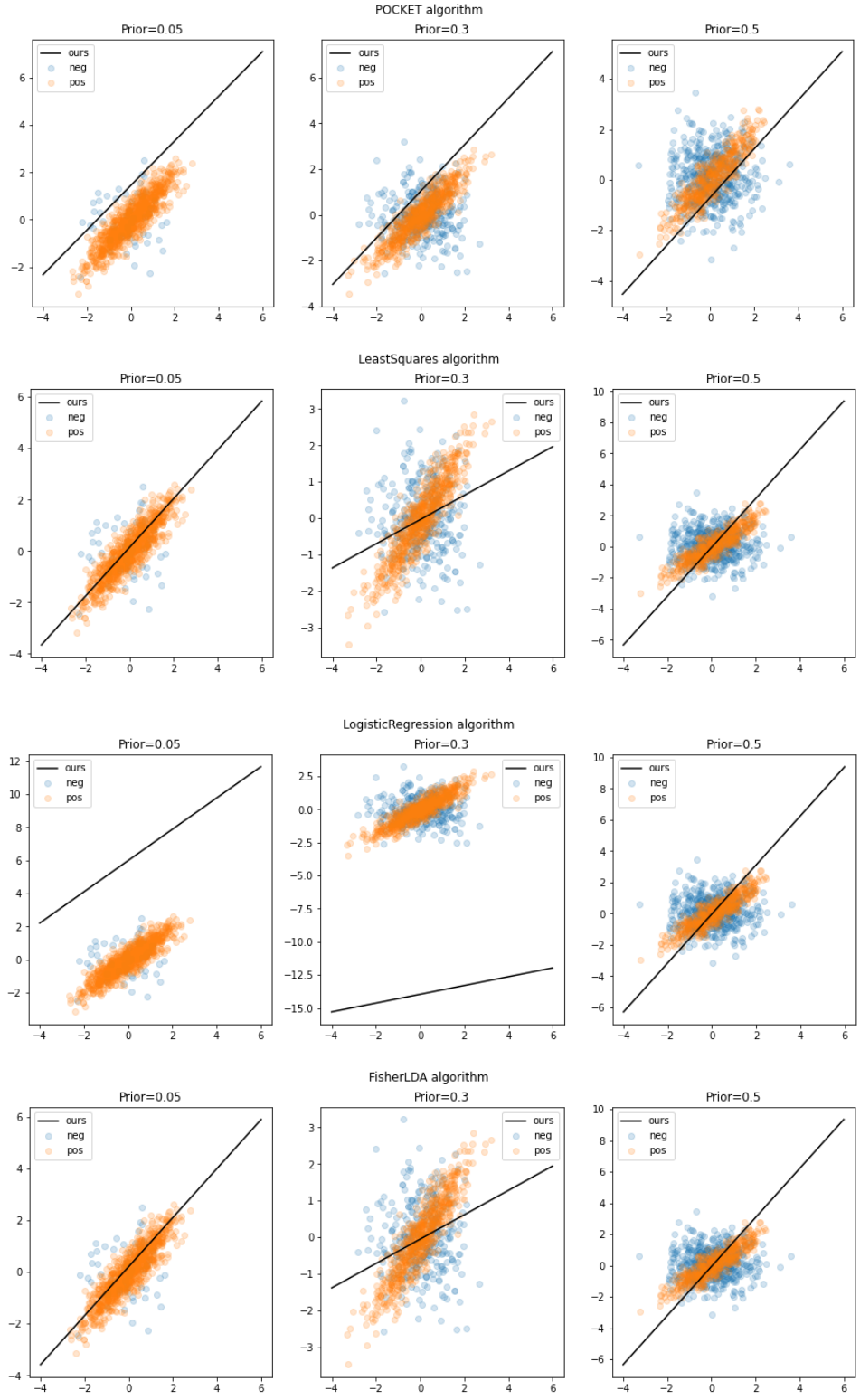
prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.626	0.629	0.950	0.762	0.765	0.974	[0.02, -0.03, 0.95]	[0.02, -0.03, 0.95]
0.30	0.520	0.524	0.700	0.607	0.608	0.824	[-0.0, 0.01, 0.7]	[-0.0, 0.01, 0.7]
0.50	0.532	0.477	0.473	0.539	0.481	0.470	[-0.01, 0.01, 0.5]	[-0.01, 0.01, 0.5]

LogisticRegression algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.950	0.950	0.950	0.974	0.974	0.974	[0.47, -0.5, 2.98]	[0.44, -0.47, 2.97]
0.30	0.700	0.700	0.700	0.824	0.824	0.824	[-0.02, 0.06, 0.85]	[-0.02, 0.06, 0.85]
0.50	0.524	0.473	0.474	0.525	0.470	0.472	[-0.06, 0.04, 0.0]	[-0.06, 0.04, 0.0]

FisherLDA algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.687	0.686	0.950	0.809	0.808	0.974	[0.51, -0.53, 0.11]	[0.51, -0.53, 2.98]
0.30	0.520	0.527	0.700	0.611	0.612	0.824	[-0.02, 0.06, 0.0]	[-0.02, 0.06, 0.85]
0.50	0.535	0.481	0.473	0.545	0.489	0.470	[-0.06, 0.04, 0.0]	[-0.06, 0.04, 0.0]


POCKET algorithm
Prior=0.05
Prior=0.3
Prior=0.5
LeastSquares algorithm
Prior=0.05
Prior=0.3
Prior=0.5
LogisticRegression algorithm
Prior=0.05
Prior=0.3
Prior=0.5
FisherLDA algorithm
Prior=0.05
Prior=0.3
Prior=0.5
ours
neg
pos

Applying feature transformation degree = 2

A function was written to calculate the numerical values of the coefficients from the above image. Using the same decision boundary, we get the following performance metrics

Bayes Decision Boundary

prior	bayes_acc	bayes_f1	weights
0.05	0.969	0.983929	[3.77, 0.0, 0.0, -2.13, 4.74, -2.13]
0.30	0.815	0.878847	[1.68, 0.0, 0.0, -2.13, 4.74, -2.13]
0.50	0.735	0.768559	[0.83, 0.0, 0.0, -2.13, 4.74, -2.13]

POCKET algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.967	0.967	0.963	0.983	0.983	0.981	[0.38, 0.12, -0.01, -0.18, 0.6, -0.13, 0.0]	[2.0, 2.65, 2.13, 1.07, 14.16, -0.99, 2.0]
0.30	0.834	0.807	0.793	0.888	0.871	0.861	[0.75, 0.14, -0.13, -0.42, 1.16, -0.57, -0.46]	[1.0, 2.78, 0.4, -2.36, 16.92, -5.94, 1.0]
0.50	0.771	0.729	0.624	0.803	0.770	0.493	[0.33, -0.13, 0.16, -0.61, 1.42, -0.6, -0.03]	[0.0, -0.07, -1.44, -7.32, 15.59, -7.06, 0.0]

LeastSquares algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.970	0.969	0.962	0.984	0.984	0.980	[0.49, 0.0, -0.0, -0.14, 0.29, -0.14, 0.49]	[0.0, 0.0, -0.0, -0.14, 0.29, -0.14, 0.99]
0.30	0.834	0.811	0.778	0.889	0.874	0.863	[-763326545046.74, 0.03, -0.02, -0.1, 0.26, -0.11, 763326545047.48]	[0.0, 0.03, -0.02, -0.1, 0.26, -0.11, 0.74]
0.50	0.769	0.726	0.722	0.801	0.768	0.774	[2880729406525.78, -0.01, 0.01, -0.06, 0.22, -0.07, -2880729406525.23]	[0.0, -0.01, 0.01, -0.06, 0.22, -0.07, 0.54]

LogisticRegression algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.968	0.969	0.969	0.983	0.984	0.984	[2.01, -0.12, 0.05, -2.43, 5.06, -2.4, 2.01]	[-0.0, -0.13, 0.06, -1.93, 4.07, -1.91, 3.77]
0.30	0.837	0.818	0.813	0.892	0.880	0.878	[0.84, 0.17, -0.16, -2.26, 5.12, -2.3, 0.84]	[-0.0, 0.15, -0.14, -2.0, 4.57, -2.03, 1.59]
0.50	0.774	0.737	0.734	0.798	0.768	0.768	[0.44, -0.11, 0.1, -2.35, 5.41, -2.44, 0.44]	[-0.0, -0.11, 0.1, -2.06, 4.76, -2.12, 0.8]

FisherLDA algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.970	0.969	0.971	0.984	0.984	0.985	[0.0, 0.04, -0.07, -4.01, 8.36, -4.18, 6.4]	[0.0, 0.04, -0.07, -4.04, 8.43, -4.21, 7.7]
0.30	0.834	0.813	0.783	0.889	0.876	0.866	[0.0, 0.21, -0.13, -0.59, 1.6, -0.67, 0.37]	[0.0, 0.21, -0.13, -0.59, 1.61, -0.67, 1.38]
0.50	0.770	0.727	0.722	0.802	0.769	0.774	[0.0, -0.03, 0.05, -0.33, 1.13, -0.37, 0.13]	[0.0, -0.03, 0.05, -0.33, 1.13, -0.37, 0.22]

Different means, different covariances

Data was drawn from 2 normal distributions with `mu0 = (3, 6), mu1 = (3, -2), cov0 = [[.5, 0],[0, 2]], cov1 = [[2,0], [0,2]]`

Without feature transformation

POCKET algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.999	0.997	0.999	0.999	0.999	0.999	[0.42, -0.61, 0.6]	[5.27, -11.57, 16.0]
0.30	0.998	0.997	0.996	0.999	0.998	0.997	[0.07, -0.97, 2.01]	[0.5, -10.82, 17.0]
0.50	0.996	0.996	0.994	0.996	0.996	0.994	[0.17, -1.01, 1.15]	[0.91, -13.95, 15.0]

LeastSquares algorithm

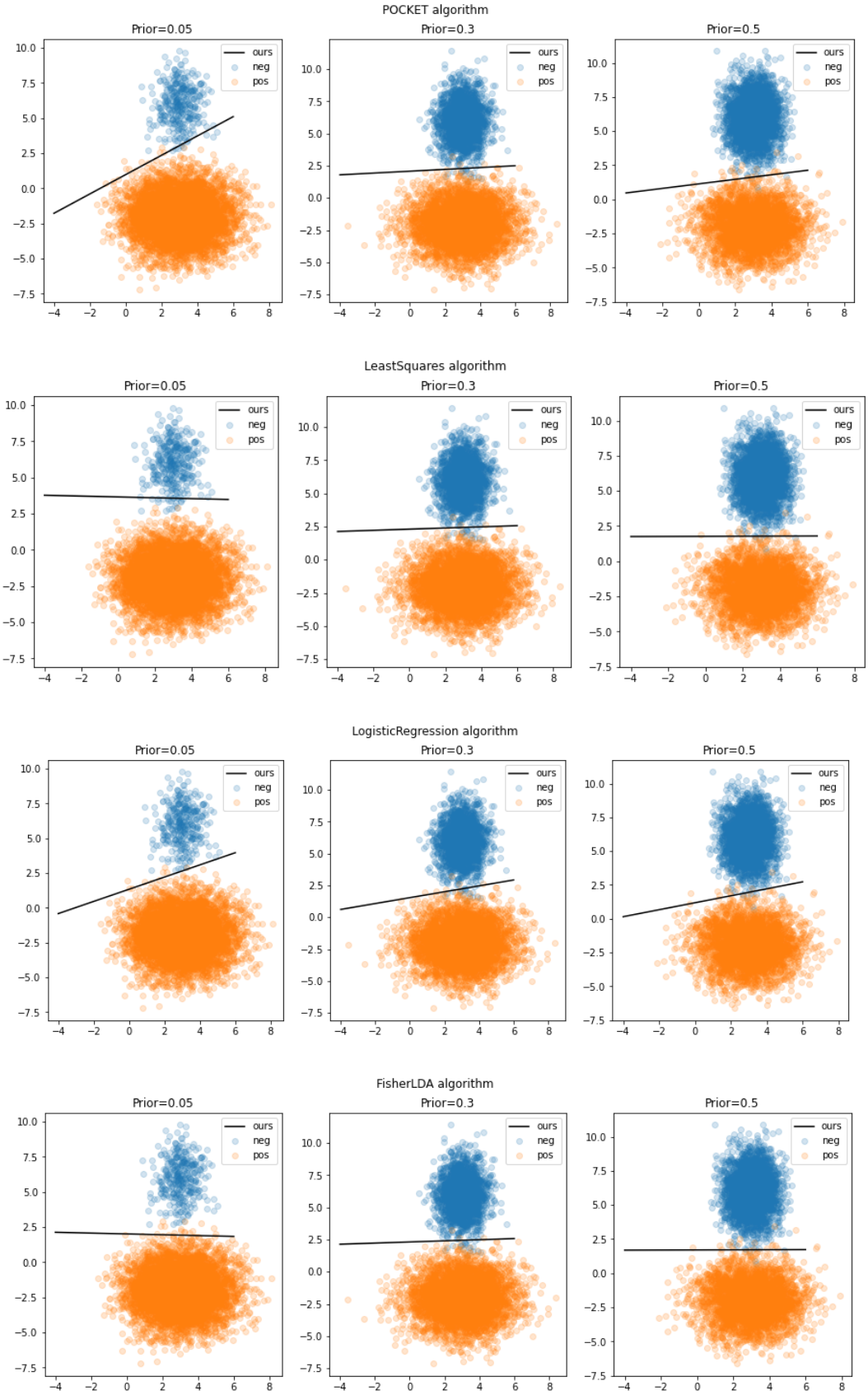
prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	1.000	0.998	0.994	1.000	0.999	0.997	[-0.0, -0.08, 0.83]	[-0.0, -0.08, 0.83]
0.30	0.999	0.997	0.997	0.999	0.998	0.998	[0.0, -0.11, 0.73]	[0.0, -0.11, 0.73]
0.50	0.997	0.997	0.997	0.997	0.997	0.997	[0.0, -0.11, 0.72]	[0.0, -0.11, 0.72]

LogisticRegression algorithm

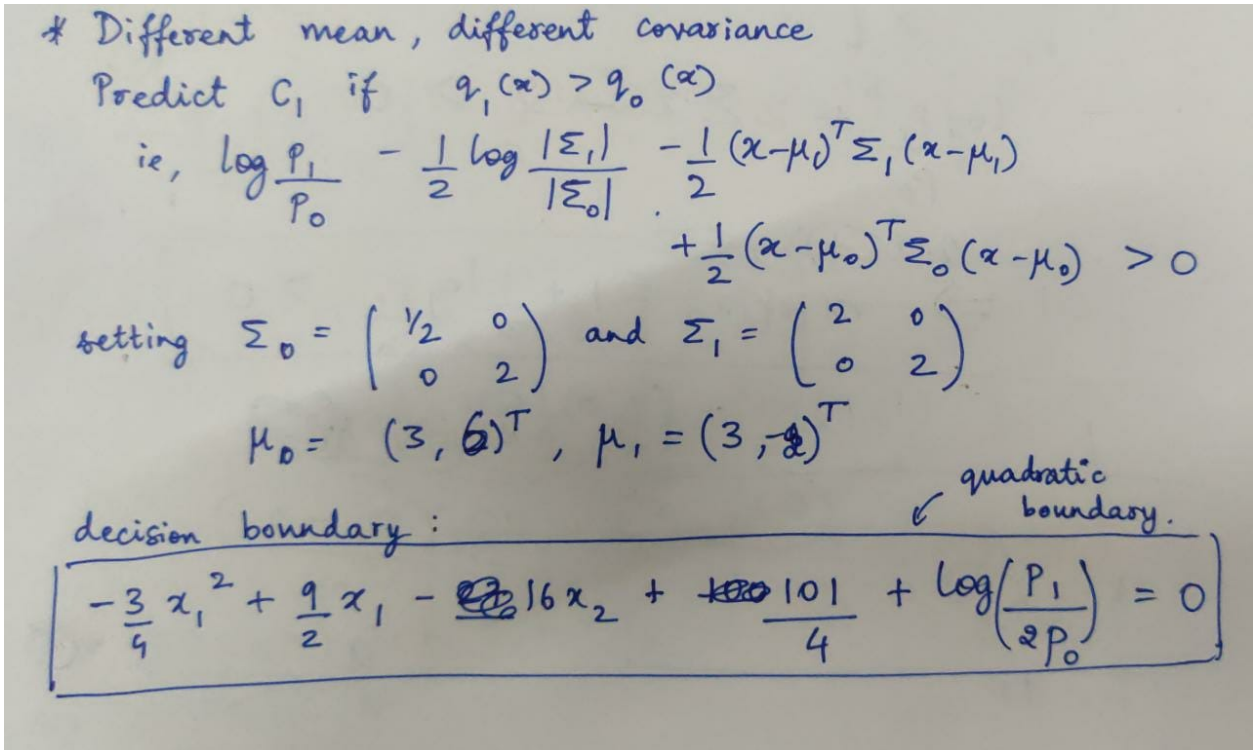
prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.999	0.999	0.999	0.999	0.999	1.000	[1.34, -3.06, 4.09]	[-0.18, -2.53, 7.76]
0.30	0.997	0.997	0.997	0.998	0.998	0.998	[0.64, -2.75, 4.25]	[-0.07, -2.75, 6.53]
0.50	0.996	0.997	0.997	0.995	0.997	0.997	[0.79, -3.07, 3.62]	[0.18, -2.92, 5.16]

FisherLDA algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	1.000	0.998	0.999	1.000	0.999	1.000	[-0.12, -4.17, 8.4]	[-0.12, -4.17, 11.61]
0.30	0.999	0.997	0.997	0.999	0.998	0.998	[0.17, -3.98, 9.23]	[0.17, -3.98, 8.3]
0.50	0.997	0.996	0.997	0.997	0.996	0.997	[0.02, -3.88, 6.61]	[0.02, -3.88, 7.63]


POCKET algorithm
Prior=0.05
Prior=0.3
Prior=0.5
LeastSquares algorithm
Prior=0.05
Prior=0.3
Prior=0.5
LogisticRegression algorithm
Prior=0.05
Prior=0.3
Prior=0.5
FisherLDA algorithm
Prior=0.05
Prior=0.3
Prior=0.5
ours
neg
pos

Bayes Classifier



Applying feature transformation degree=2

Bayes Decision Boundary

prior	bayes_acc	bayes_f1	weights
0.05	0.998143	0.999022	[27.5, 4.5, -16.0, -0.75, 0.0, 0.0]
0.30	0.996857	0.997755	[25.4, 4.5, -16.0, -0.75, 0.0, 0.0]
0.50	0.996571	0.996568	[24.56, 4.5, -16.0, -0.75, 0.0, 0.0]

POCKET algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	0.998	0.997	0.998	0.999	0.998	0.999	[3.21, 2.02, -4.39, 5.8, -7.01, 1.57, 2.6]	[54.0, 32.04, -41.3, 26.34, -59.13, 6.21, 54.0]
0.30	0.998	0.997	0.997	0.999	0.998	0.998	[11.26, 8.84, -9.71, 0.88, -5.22, 0.1, 11.59]	[94.0, 67.46, -98.18, 14.09, -45.3, 4.2, 94.0]
0.50	0.996	0.997	0.995	0.996	0.997	0.995	[7.14, 1.88, -9.05, 3.04, -6.01, -0.07, 6.0]	[79.0, 27.25, -102.85, 45.91, -67.21, 8.2, 79.0]

LeastSquares algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	1.000	0.997	0.996	1.000	0.998	0.998	[0.48, -0.01, -0.06, 0.0, -0.0, -0.01, 0.48]	[0.0, -0.01, -0.06, 0.0, -0.0, -0.01, 0.95]
0.30	0.998	0.997	0.995	0.999	0.998	0.996	[0.43, -0.03, -0.09, 0.01, 0.0, -0.01, 0.43]	[0.0, -0.03, -0.09, 0.01, 0.0, -0.01, 0.86]
0.50	0.997	0.998	0.998	0.997	0.998	0.998	[0.4, -0.05, -0.11, 0.01, 0.0, -0.0, 0.4]	[0.0, -0.05, -0.11, 0.01, 0.0, -0.0, 0.79]

FisherLDA algorithm

prior	train_acc	acc	sk_acc	train_f1	f1	sk_f1	weights	sk_weights
0.05	1.000	0.998	0.997	1.000	0.999	0.998	[0.0, -1.34, -8.92, 0.15, -0.16, -1.69, 27.14]	[0.0, -1.34, -8.98, 0.15, -0.16, -1.71, 61.86]
0.30	0.998	0.997	0.995	0.999	0.998	0.996	[0.0, -1.42, -3.94, 0.27, 0.04, -0.24, 11.29]	[0.0, -1.42, -3.94, 0.27, 0.04, -0.24, 15.05]
0.50	0.997	0.997	0.998	0.997	0.997	0.998	[0.0, -1.74, -3.98, 0.3, 0.06, -0.03, 8.8]	[0.0, -1.74, -3.98, 0.3, 0.06, -0.03, 10.24]

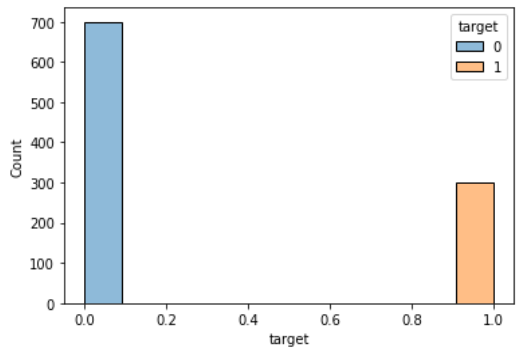
Q2: German Credit Data

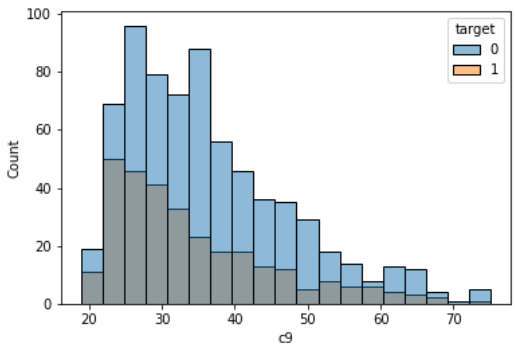
This dataset contains attributes of different people (numerical and categorical) and a target (binary) denoting good/bad credit risk.

EDA

For the remaining graphs, please refer to the colab notebook

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14	c15	c16	c17	c18	c19	c20	c21	c22	c23	target
0	1	6	4	12	5	5	3	4	1	67	3	2	1	2	1	0	0	1	0	0	1	0	0	1	0
1	2	48	2	60	1	3	2	2	1	22	3	1	1	1	1	0	0	1	0	0	1	0	0	1	1
2	4	12	4	21	1	4	3	3	1	49	3	1	2	1	1	0	0	1	0	0	1	0	1	0	0
3	1	42	2	79	1	4	3	4	2	45	3	1	2	1	1	0	0	0	0	0	0	0	0	1	0
4	1	24	3	49	1	3	3	4	4	53	3	2	2	1	1	1	0	1	0	0	0	0	0	1	1





Without transformation

Without transforming
POCKET algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	0.772	0.740	0.690	0.500	0.447	0.597
0.3	0.776	0.687	0.687	0.529	0.338	0.113

LeastSquares algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	0.795	0.765	0.765	0.627	0.569	0.535
0.3	0.796	0.733	0.733	0.619	0.500	0.500

LogisticRegression algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	0.780	0.760	0.77	0.562	0.529	0.540
0.3	0.786	0.717	0.73	0.595	0.459	0.484

FisherLDA algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	0.786	0.755	0.770	0.647	0.574	0.566
0.3	0.793	0.700	0.723	0.660	0.536	0.497

Transformation degree=4

POCKET algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	0.781	0.770	0.57	0.615	0.596	0.552
0.3	0.793	0.707	0.75	0.625	0.494	0.586

LeastSquares algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	1.0	0.460	0.515	1.0	0.341	0.340
0.3	1.0	0.503	0.553	1.0	0.433	0.407

LogisticRegression algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	0.742	0.750	0.775	0.376	0.444	0.587
0.3	0.760	0.717	0.737	0.458	0.351	0.533

FisherLDA algorithm

test_size	train_acc	acc	sk_acc	train_f1	f1	sk_f1
0.2	0.3	0.350	0.625	0.462	0.444	0.490
0.3	0.3	0.437	0.557	0.462	0.470	0.453

XGboost classifier

```
Test size: 0.2
  F1  score: 1.000(train), 0.544(test)
  Acc score: 1.000(train), 0.765(test)
Test size: 0.3
  F1  score: 1.000(train), 0.522(test)
  Acc score: 1.000(train), 0.743(test)
```

Q3: Porto Seguro's Safe Driver

Kaggle competition: use features in the data to predict whether an auto insurance policy holder files a claim.

The dataset has 58 features, the suffixes of the name denote if the feature is categorical or not. This indicator has been used to encode those columns into one-hot vectors.

It was noted that there was very high class imbalance - only 3% of the examples belonged to the positive class. It is important to note that a model predicting zero all the time would get 97% accuracy.

Here, we have set out to maximize the `f1_score` as our performance metric, which is independent of the class imbalance.

We first divide the whole dataset into training and the test set. To tackle the class imbalance problem, we resample the training set to contain an equal number of positive and negative examples. This under-sampled training set is fed to all of the training algorithms below.

POCKET algorithm

test_size	acc	f1
0.2	0.823	0.086
0.3	0.921	0.079

LeastSquares algorithm

test_size	acc	f1
0.2	0.716	0.093
0.3	0.708	0.100

LogisticRegression algorithm

test_size	acc	f1
0.2	0.035	0.068
0.3	0.964	0.000

FisherLDA algorithm

test_size	acc	f1
0.2	0.763	0.098
0.3	0.737	0.101

Xgboost classifier

We make an ensemble model as follows: repeatedly sample training set as mentioned above, train a model on it. Average the test predictions over all the models to measure performance over the test set.

```
Test size: 0.2
Training:
  Fold0: Acc: 0.781, f1:0.763
  Fold1: Acc: 0.789, f1:0.777
  Fold2: Acc: 0.774, f1:0.762
  Fold3: Acc: 0.775, f1:0.763
  Fold4: Acc: 0.777, f1:0.760
Testing:
  Acc: 0.703, f1:0.099
Test size: 0.3
Training:
  Fold0: Acc: 0.794, f1:0.786
  Fold1: Acc: 0.801, f1:0.795
  Fold2: Acc: 0.796, f1:0.794
  Fold3: Acc: 0.796, f1:0.790
  Fold4: Acc: 0.785, f1:0.778
Testing:
  Acc: 0.687, f1:0.101
```