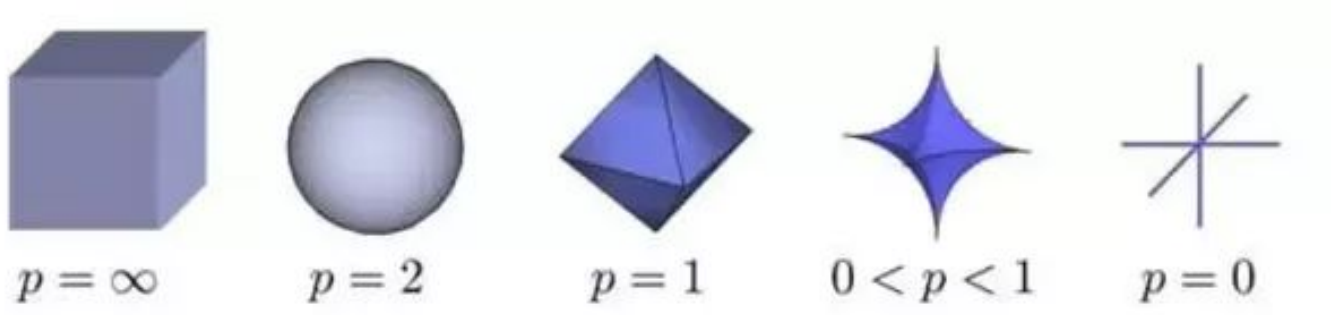
Question 4

- For each part, the images display the predicted y values (in blue) and the actual y values (in black). All these plots are with respect to the standard deviation of x component values.
- The values in x components were pre-processed so as to ensure 0 mean and unit variance across each dimension. This was done by computing the numerical mean and standard deviation across every dimension independently, and then normalizing each feature in every x independently. This was done only while printing the plots.
- The images on the left display the values predicted by regression, while the images on the right display the normalized values (set 1 if predicted value was greater than 0.5, else set to 0).

Regularization

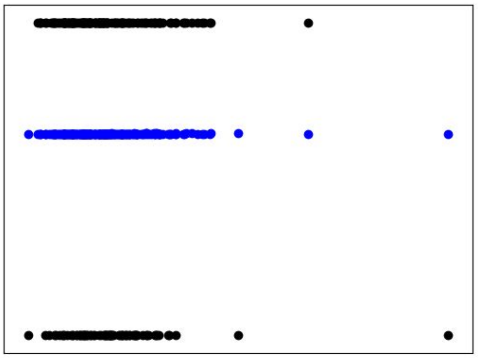
Regularization is a technique which introduces additional information to prevent overfitting of data. Therefore, it can be said that Regularization enforces that the output values are more “general” / evenly spread.
There are many ways of doing so, and we see two major ways in this question.
For any p >= 1, the Lp norm is defined as:

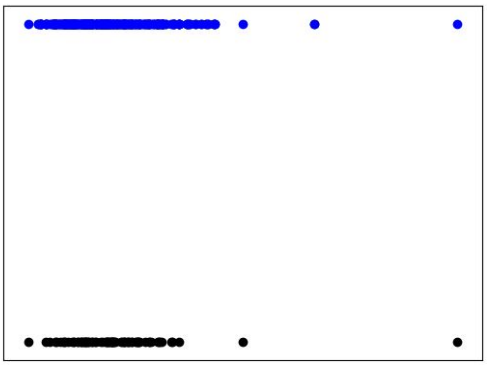
$$\|x\|_p = \left(|x_1|^p + |x_2|^p + \cdots + |x_n|^p\right)^{\frac{1}{p}}$$



The above image shows various lp balls. As the value of p decreases, the size of corresponding lp space also decreases.
L1 norm corresponds to p = 1, and similarly, L2 norm corresponds to p=2.

Part 1 - Lasso (L1) Regularization

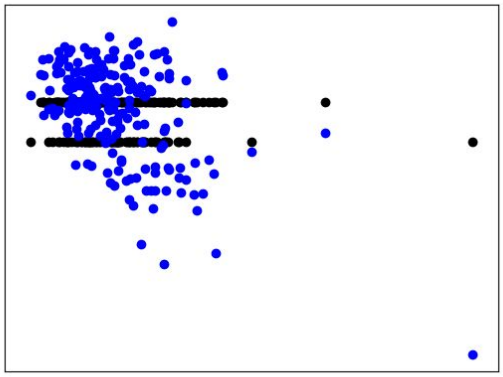


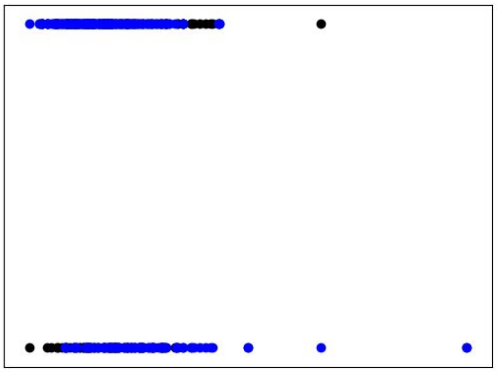


Lasso forces the sum of the absolute value of the regression coefficients to be less than a fixed value, which forces certain coefficients to be set to zero, effectively choosing a simpler model that does not include those coefficients. In other words, if there is a group of highly correlated variables, then lasso tends to select one variable from a group and ignore the others.

We get a training accuracy of 62.77% and a testing accuracy of 63.94% with l1 norm.
This was for parameter values alpha=1.0, fit_intercept=True, normalize=False, precompute=False, copy_X=True, max_iter=10000, tol=0.0001, warm_start=False, positive=False, random_state=None, selection='cyclic'

Part 2 - Ridge (L2) Regularization



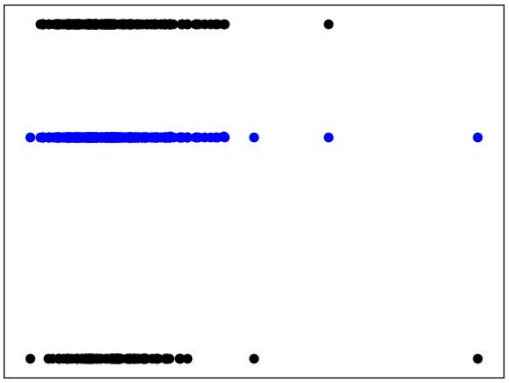


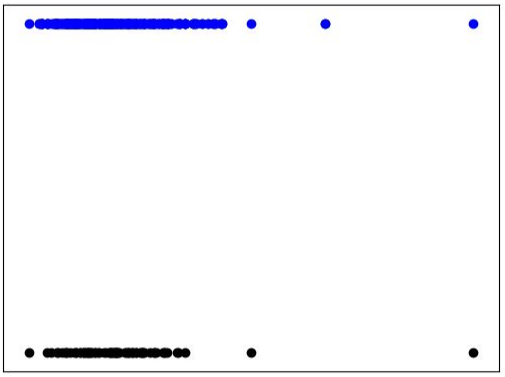
Ridge Regularization, as opposed to Lasso Regularization, gives preference to solutions with smaller norms, and does not force them to be set to 0.
All possible vectors of some L2-norm, say 1/2, form a unit hypersphere. Putting an L2-norm of parameters in the objective function constraints the optimised theta to a specific limited length.
That is the reason, when in ridge linear regression with polynomial features of a high degree, the coefficients do not blow up, because the *feasible set* of solutions for theta is restricted to a ball.

Since L2 norm allows for a bit more “freedom” than L1 norm, we see that the values obtained here are a bit more dispersed than what is obtained in L1 norm. Typically ridge penalties are much better for minimizing prediction error rather than l1 penalties. The reason for this is that when two predictors are highly correlated, l1 regularizer will simply pick one of the two predictors. In contrast, the l2 regularizer will keep both of them and jointly shrink the corresponding coefficients a little bit.

We get a training accuracy of 74.68% and a testing accuracy of 70.19%, which is better than what we had obtained with l1 norm.
This was for parameter values alpha=1.0, fit_intercept=True, normalize=False, copy_X=True, max_iter=None, tol=0.001, solver=’auto’, random_state=None

Part 3 - ElasticNet (L1 + L2) Regularization



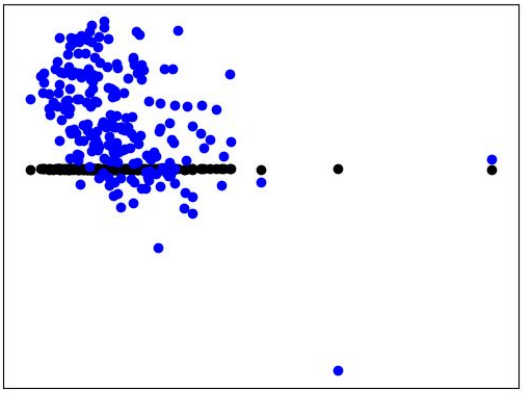


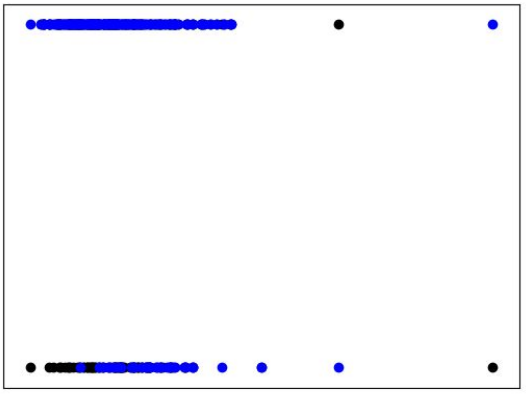
ElasticNet linearly combines l1 norm and l2 norm. In doing so, it tries to work upon the limitations of l1 and l2 as each offers something that the other method lacks.

We get a training accuracy of 62.77% and a testing accuracy of 63.94% with l1 + l2 norm and l1 ratio > 0.1. Surprisingly, this is exactly the same as what we get with l1 norm.
For l1 ratio 0.1, we get a training accuracy of 64.29% and a testing accuracy of 63.94%, while with l1 ratio of 0.05, we get a training accuracy of 64.07% and a testing accuracy of 64.90%.

This was for parameter values alpha=1.0, l1_ratio=(as specified above), fit_intercept=True, normalize=False, precompute=False, max_iter=1000, copy_X=True, tol=0.0001, warm_start=False, positive=False, random_state=None, selection=’cyclic’.

Part 4 - No Regularization





We get a training accuracy of 75.32% and a testing accuracy of 73.08% without regularization. The training accuracy is higher than with regularization as the model proceeds to overfit the data. This may reflect in real life with extremely low test time accuracies, even though what we get here is reasonably high.