1)While it is hard to write the explicit formula for the bias and variance of using LASSO, we can quantify the expected general trend. Make sure you justify the answers to the following questions for full points:

(a) (Written) What is the general trend of the bias as λ increases?

As λ increases, the bias of LASSO increases. This is because stronger regularization constrains the model more, forcing coefficients to shrink toward zero and introducing more bias.

(b) (Written) What about the general trend of the variance as λ increases?

As λ increases, the variance of the model typically decreases. Strong regularization reduces the model's complexity and makes it less sensitive to variations in the training data, which reduces overfitting.

(c) (Written) What is the bias at $\lambda = 0$?

When $\lambda = 0$, there is no regularization. In this case, LASSO reduces to ordinary least squares regression, which is an unbiased estimator. So the bias at $\lambda = 0$ is low.

(d) (Written) What about the variance at $\lambda = \infty$?

At $\lambda = \infty$ in the context of LASSO (Least Absolute Shrinkage and Selection Operator), the regularization term becomes extremely strong, which in turn converts all coefficients to exactly zero. This results in an overly simple model that fails to capture important relationships in the data. So the variance at $\lambda = \infty$ is low.

2c) Fit a Naive Bayes model to each of the four preprocessing steps above using the code in 2b. Each preprocessing should be performed independently (i.e., use each of the functions you created in 2a on the original dataset). Report the accuracy rate and AUC on the training and test sets across the 4 preprocessing steps in a table

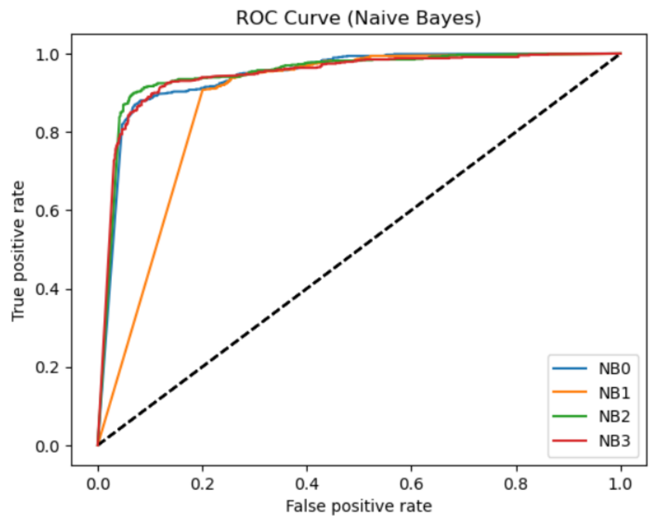
	do_nothing()	do_std()	do_log()	do_bin()
train_acc	0.82533333333333	0.816	0.82366666666666	0.79866666666666
train_auc	0.94672776024229	0.89066645574017	0.95434411562026	0.94941143997701
test_acc	0.81698938163647	0.81011867582760	0.81511555277951	0.80012492192379
test_auc	0.94227082789660	0.87526297645857	0.94812235614019	0.94473064444938

2e) Fit a standard (no regularization) logistic regression model to each of the four preprocessing steps above using the code in 2d. Report the accuracy rate and AUC on the training and test sets for the 4 preprocessing steps in a table.

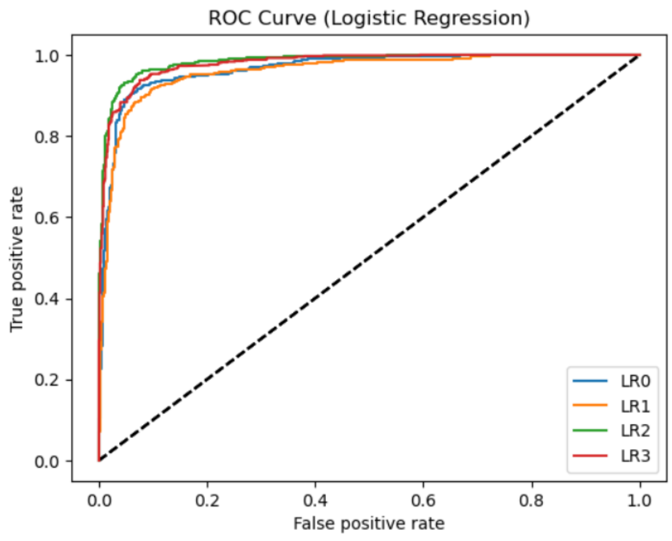
	do_nothing()	do_std()	do_log()	do_bin()
train_acc	0.92466666666666	0.901	0.94866666666666	0.93733333333333
train_auc	0.96778246468545	0.96443737079187	0.98518296152062	0.98055001117221
test_acc	0.92317301686445	0.89131792629606	0.93816364772017	0.92192379762648
test_auc	0.96514696074925	0.95891768071677	0.98359155769051	0.97915599457455

2f)Plot the receiver operating characteristic (ROC) curves for the test data. You should generate 3 plots:

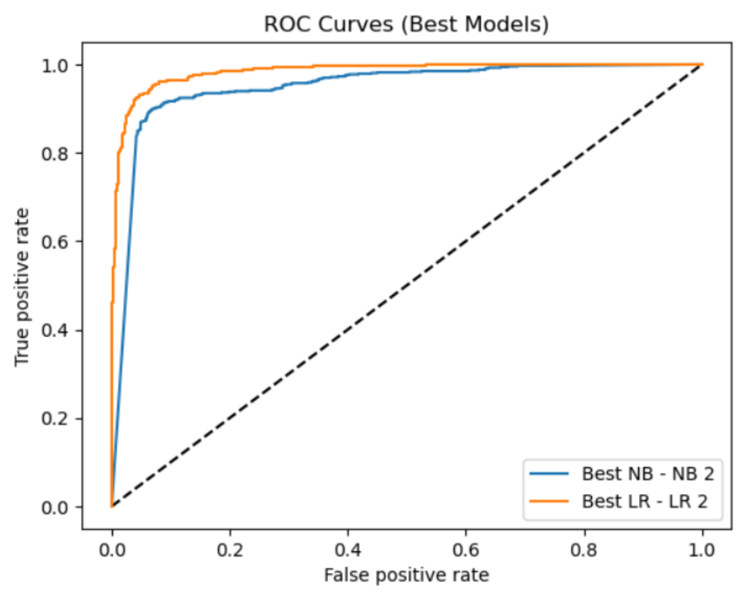
• One plot containing the 4 Naive Bayes model curves representing each of the preprocessing steps.



• One plot containing the 4 logistic regression model curves representing each of the preprocessing steps.



• One plot containing the best Naive Bayes model and the best logistic regression model curve



Where 0 1 2 3 stands for the preprocessing done, i.e do_nothing, do_std, do_log, do_bin on the data. I've determined the best curve using AUC values, calculated using metrics.auc(fpr, tpr).

2g) Given your results in 2c, 2e, and 2f, comment on how the preprocessing affects the models (logistic and Naive Bayes) with regards to ROC, AUC, and accuracy. Also, comment on how Naive Bayes compares with logistic regression.

Based on the results from parts 2c, 2e, and 2f, we can infer :

- Preprocessing slightly decreased the test accuracy of Naive Bayes (Gaussian Model) in all the cases. Whereas the test AUC increased in Naive Bayes when we preprocess it with do_log. Overall AUC was best in the case where we use do_log to the Naive Bayes Model.
- Preprocessing slightly decreased the test accuracy of Logistic Regression in standardization and binarization, where it increased significantly in do_log preprocessing. Whereas the test AUC increased in Naive Bayes when we preprocess it with do_log or do_bin. Overall AUC was best in the case where we implement do_log to the Logistic Regression Model.
- Overall, Logistic Regression outperformed Naive Bayes in terms of AUC and accuracy across all preprocessing techniques.

3. (a) (Written) How did you preprocess the data for this method? Why?

I preprocessed the data using do_log() for this question. Because I got the best accuracy values in the logistic regression (2e) for both test and train data.

3e) What is your regularization parameter search space for ridge and LASSO- regularized logistic regression?
For Ridge, my regularization parameter search space would be np.linspace(0.01, 10, 100) - Searches small alpha values from 0.01 to 10 on a linear scale with 100 values.

For Lasso, my regularization parameter search space would be np.linspace(0.01, 1, 10) - Searches small alpha values from 0.01 to 1 on a linear scale with 10 values.

3 g) For your regularization parameter search space specified in 3f, fit ridge and LASSO using 3d by searching over a variety of split ratios. For each unique split ratio, report the validation metrics (AUC and accuracy). What is the best 'parameter' for ridge and LASSO based on these metrics? Note ridge and LASSO may have different optimal 'parameters' .

I have specified the best accuracy for each split and reported that parameter below. From that, we can pick the best parameter and the split ratio.

For Ridge :

Split Ratio	Regularization Parameter (For best Val_acc in this split)	Validation acc	Validation AUC
0.1	5.56	0.96	0.9895133228466562
0.2	0.01	0.955	0.9935881481481482
0.3	2.23	0.9511	0.9861966314140227
0.4	8.89	0.9525	0.9819919001131862
0.5	7.78	0.946	0.9829905550086142
0.6	8.89	0.9427	0.9845594913714805
0.7	2.23	0.9466	0.9831546270757623
0.8	10.0	0.937	0.9774889835487662
0.9	2.23	0.9377	0.9792022377189227

For Ridge, the optimal parameter is 5.56 when the split ratio is 0.1 based on the validation metrics.

For Lasso :

Split Ratio	Regularization Parameter (For best Val_acc in this split)	Validation acc	Validation AUC

0.1	0.3	0.9633	0.9858429858429859
0.2	0.3	0.95	0.9855023923444975
0.3	0.7	0.9533	0.9880578382365838
0.4	0.1	0.945	0.9822248927339071
0.5	0.9	0.9446	0.9829102458264538
0.6	1.0	0.95	0.9829355442506482
0.7	0.7	0.9404	0.9759764789872334
0.8	0.9	0.9366	0.9780397405012221
0.9	0.8	0.9351851851851852	0.9772725029050338

For Lasso, the optimal parameter is 0.3, when the split ratio is 0.1 based on the validation metrics.

3 h) For your regularization parameter search space specified in 3f, fit ridge and LASSO using the k-fold cross-validation approach by searching over k = 2 , 5 , 10. For each value of k , specify the best 'parameter' based on the metrics

For Ridge :

k	Regularization Parameter (For best Val_acc in this split)	Validation acc	Validation AUC
2	3.34	0.946	0.982739996471611
5	7.78	0.945	0.9816094170502181
10	8.89	0.945	0.9826252063222369

The best Validation metrics are when K = 2 and Regularization parameter is 3.34, for Ridge.

For LASSO :

k	Regularization Parameter (For best Val_acc in this split)	Validation acc	Validation AUC
2	0.4	0.94166	0.9803724328099968
5	0.7	0.9426666666666665	0.9816667449527005

10	0.2	0.9426666666666668	0.9816855009670599

The best Validation metrics are when K = 10 and Regularization parameter is 0.2, for LASSO.

3j) Fit ridge and LASSO using the Monte Carlo Cross-validation approach with s = 5 , 10 samples and different split ratios. What is the best 'parameter' based on the splits and the samples?

For Ridge :

Sample Size	Split Ratio	Regularization Parameter (For best Val_acc in this split)	Validation acc	Validation AUC
5	0.1	8.89	0.9573	0.984238051673273
5	0.2	8.89	0.952	0.982608242847217
5	0.3	5.56	0.9455	0.984420976339683
5	0.4	5.56	0.9456	0.982328459132125
5	0.5	4.45	0.9446	0.982461644279225
10	0.1	8.89	0.9476	0.983075590986816
10	0.2	5.56	0.948	0.982980375361839
10	0.3	6.67	0.9464	0.983062516889105
10	0.4	10.0	0.9463	0.983102039359522
10	0.5	8.89	0.9439	0.981578310935673

For Monte Carlo CV, The best parameter of Ridge is 8.89 when the sample size is 5 and split ratio is 0.1, based on validation metrics.

For Lasso :

Sample Size	Split Ratio	Regularization Parameter (For best Val_acc in this split)	Validation acc	Validation AUC
5	0.1	0.8	0.9506	0.986700453848005
5	0.2	0.8	0.9473	0.983031969003881

5	0.3	0.7	0.9446	0.982380061926162
5	0.4	0.4	0.9448	0.982589358834878
5	0.5	0.9	0.9409	0.980376300754859
10	0.1	0.4	0.9473	0.985396800346311
10	0.2	0.7	0.9448	0.984251963679539
10	0.3	0.4	0.9460	0.983361724609121
10	0.4	0.1	0.9411	0.981824446853851
10	0.5	0.8	0.9408	0.979935548943412

For Monte Carlo CV, The best parameter of Lasso is 0.8 when the sample size is 5 and split ratio is 0.1, based on validation metrics.

3k) Using the optimal parameters identified in 3g, 3h, and 3j, re-train the regularized logistic regression model using all the training data and report the performance on the test set in terms of AUC and accuracy in a table. Note that this means you are likely training at least 6 different regularized models and reporting the performance for each of the models.

For Ridge,

Optimal Parameter	Test - Acc	Test- AUC
5.56 (using Eval_holdout)	0.9212991880074953	0.9648457258472434
3.36(using Eval_kfold)	0.9131792629606496	0.9650350044621427
8.89(Using MonteCarlo_cv)	0.9194253591505309	0.9650309772575704

For Lasso,

Optimal Parameter	Test - Acc	Test- AUC
0.3 (using Eval_holdout)	0.9206745783885072	0.9721655728779048
0.2(using Eval_kfold)	0.9219237976264835	0.97181117887554
0.8(Using MonteCarlo_cv)	0.9219237976264835	0.9719158861944205

3(l) (Written) Comment on how the different model selection techniques compare with one another with regard to AUC and accuracy, the robustness of the validation estimate, and the computational complexities of the three different hold-out techniques.

AUC and Accuracy:

- Holdout provides a single train/test split, so accuracy can vary a lot depending on the split. But the best accuracy is obtained in this.
- K-fold produces more robust estimates by averaging over multiple splits.
- Monte Carlo also produces more robust estimates by averaging over multiple splits.

Robustness of validation estimate:

- K-fold reduces variance by averaging k folds.
- Monte Carlo reduces variance by averaging multiple splits.
- The holdout validation set estimate has low variance by itself. But the overall holdout model evaluation has high variance because it depends on the particular split.

Computational Complexity :

- Hold-out is fastest while K-Fold and Monte Carlo take more compute time.
- Monte Carlo CV is similar to k-fold CV in compute time.