Exploring Naive Bayes Under Adverse Conditions

DS 5220 Spring 2019 Project Report

Andrew Reichel (MSDS 2020), Smruthi Ramesh (MSDS 2020), Ross Marino (MSDS 2020), Adam Ribaudo (MSDS 2020)

Introduction

The Naive Bayes classification method is a generative model known to perform well even when its underlying assumption of conditional independence between predictors given a particular class is broken. That being the case, conditions exist where the underlying data breaks this assumption to such an extent that class assignments change and predictions are no longer useful. This project explores these and other thresholds that outline when Naive Bayes performs worse than other models that make fewer or different assumptions about the data.

The following adverse data conditions were explored as part of this project.

- Dependence between variables (breaking the assumption of conditional independence)
- High number of predictors and low number of observations (as found in biomedical data)
- Non-gaussian distributions (as found in manufacturing data)
- Missing values (as found in survey data)
- High class imbalance (as found in text classification data)
- Combinations of the above conditions

Methods

Dependence Between Variables

To explore the effects of dependence between variables on Naive Bayes classification, we generated data using the "True Model" shown in **Figure 1** with the conditional probabilities tables provided in **Appendix A**.

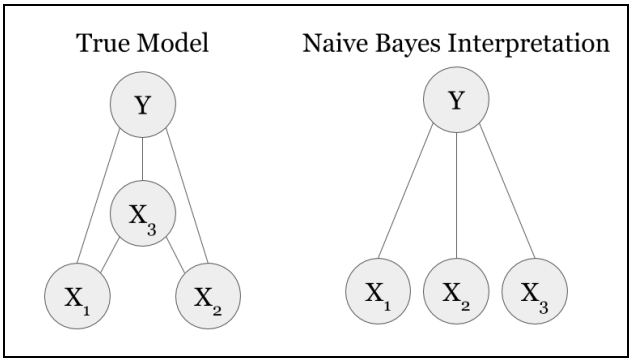


Figure 1: (left) A Bayes Network showing the true model used in our examples (right) the Naive Bayes interpretation of this model

Naive Bayes classification is robust to certain types of variable dependence due to the fact that only specific combinations of dependency produce misclassification as shown by Zhang, 2004.

In the 2-class classification problem shown above in the “True Model”, the Naive Bayes interpretation is potentially at risk of over-classifying or under-classifying observations due to the dependencies between X3, X2 and X1. What Zhang, 2004 shows, is that the amount by which a Naive Bayes prediction, f_{nb}, is different than the true model’s prediction, f_b, can be quantified as $f_b = f_{nb} * DF(E)$.

$DF(E)$ is the *dependence distribution factor* for an observation E. This is a function of each variable’s local *dependence derivatives (dd)* and their *dependence derivative ratios (ddr)*. Each variable’s local *dependence derivative* is expressed as a per-class ratio of the variable’s conditional probability including its parent predictors over the conditional probability excluding its parent predictors. These terms are defined in **Appendix A**.

Because $DF(E)$ is the product of individual ddr calculations, we can see how a variable with a ddr weighted in one direction can balance out the ddr of another variable weighted in the other direction, thus leaving the Naive Bayes prediction equivalent to the Bayes Net prediction. This is shown graphically in **Appendix A** under “Dependency Balance”

Classifying XOR Data

While the Naive Bayes model does not necessarily draw a linear decision boundary in the dimensions of its feature space, it suffers from the inability to classify XOR data which is an issue common to most linear classification methods. For our purposes, XOR data is defined as observations with at least 2 predictors whose class labels are dependent on only 1 predictor taking an “on” value and the other taking an “off” value.

We demonstrate this issue clearly in the results section by generating data in the format described above and displaying the decision boundary drawn by Naive Bayes as compared to a non-linear classifier such as a Decision Tree.

Non-Gaussian Feature Distribution

The Naive Bayes models used to classify the above cases assumes the likelihood of each feature to be gaussian. However, a gaussian naive bayes model still performs comparably to other generative models in the presence of non-gaussian data. This project sought to demonstrate the efficacy of a Gaussian Naive Bayes classifier, trained on non-normal data to better understand the impact of applying an assumption of feature-wise gaussian distributions.

To test the impact of assuming the likelihood of each feature follows a normal distribution in the presence of non-normal data, two unique and clearly non-gaussian datasets we generated. Four additional models were trained on the training set of the data, enabling the comparison of model efficacy against that of the Gaussian Naive Bayes model. Comparing the Naive Bayes

model against other generative models was expected to be the most comparable due to the generative practice of deriving the decision boundary from the class distribution.

The two examples of non-gaussian distributions that best exemplified the strength of Naive Bayes were clustered class distributions, and skewed or multimodal features. As shown in **Appendix C** below, the data, for both test cases, were generated to represent two classes, each with two features, X1 and X2. The clustered data was generated with each class being represented by two clusters initially generated from a normal distribution centered about random values, introducing random interdependence with a defined degree of class separation adding various types of "further noise" to synthesize the data[1].The skewed dataset was engineered by drawing random samples from a non-central, chi-square distribution, and further manipulated to control the separability of each class.

To characterize the performance Gaussian Naive Bayes models in each test case, the Naive Bayes model was compared against logistic regression, linear discriminant analysis, quadratic discriminant analysis and a Naive Bayes model fit using Kernel Density Estimation. On the metrics of accuracy and recall, each of the five models are directly comparable as they were trained and evaluated using a common training and test set.

Missing Values

Another common issue with data is that of missing values. When the values are sparse and random, sometimes simple imputation techniques, such as mean imputation, can be used without too much adverse effect on the predictive model. However, we hypothesized that missing data would hamper some modeling techniques more than others. Specifically we compared Naive Bayes (NB), Support Vector Machines (SVM) and Logistic regression with varying amounts of missing data and two imputation methods, mean imputation and KNN imputation.

We generated two predictors with gaussian distributions and a binary response variable. From there we randomly selected from the predictors which values to remove. We performed an exploratory analysis with 20% missing data to see how basic summary statistics varied for the predictors.

After understanding the data better we graphed the accuracy and recall of the models with an increasing proportion of missing values. Finally we drew decision boundaries for the models to look at how the models might classify new data outside the range of our generated set.

[1] Sci kit learn make.classification method:
https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

Class Imbalance

Since the Naive Bayes classifier, like many other generative classifiers, takes a prior class probability into account while making predictions, it becomes interesting to explore the effect of class imbalance on the performance of the classifier. In order to demonstrate how Naive Bayes is impacted by the presence of class imbalance in the data, we generated data from an adapted two-class classification example from Rennie, et al. (2003) as shown in **Figure 2**. For every sample from Class 1 which has a higher rate of heads (0.25), we sample twice from Class 2 which has a lower rate of heads (0.2).

Class 1 $\theta = 0.25$	Class 2 $\theta = 0.2$	$p(\text{data})$	$\hat{\theta}_1$	$\hat{\theta}_2$	Label for H
T	TT	0.48	0	0	*none*
T	{HT,TH}	0.24	0	$\frac{1}{2}$	Class 2
T	HH	0.03	0	1	Class 2
H	TT	0.16	1	0	Class 1
H	{HT,TH}	0.08	1	$\frac{1}{2}$	Class 1
H	HH	0.01	1	1	*none*

$$p(\hat{\theta}_1 > \hat{\theta}_2) = 0.24$$
$$p(\hat{\theta}_2 > \hat{\theta}_1) = 0.27$$

Figure 2: Sample Classification Data

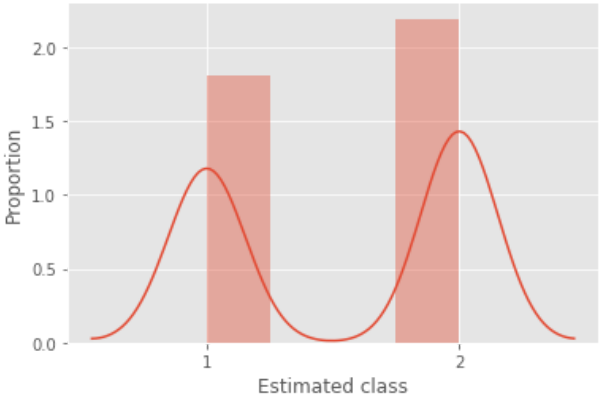


Figure 3: Sampling Distribution of Predictions

This method of generating data gives us twice as much data from Class 2 than we have from Class 1. After repeating this process 1000 times and generating the sampling distribution of the predictions, we see from **Figure 3** that we predict heads to be from Class 2 more often than we do Class 1 even though Class 1 has the higher rate of heads. Therefore, we can conclude that when the prior probabilities that we take into account do not accurately represent the population probabilities for that class, it is likely that we are inducing bias in our classifier by accounting for them.

Oversampling is a method that is commonly used to tackle the problem of unbalanced classes in practice. However, since the focus of the paper is on exploring the robustness of Naive Bayes as a classifier, we wanted to look at whether we can address class imbalance while preserving the structure of the Naive Bayes classifier. To this end, we explored the effectiveness of the Complement Naive Bayes classifier (Rennie et al.) which is a modified version of Naive Bayes that ensures that the model is trained on an equal amount of data for each class. It uses the following formula for estimation:

$\hat{\theta}_{\tilde{c}i} = \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha}$ where $N_{\tilde{c}_i}$ is the number of times word i occurred in documents in classes other than c and $N_{\tilde{c}}$ is the total number of word occurrences in classes other than c, and α_i

and α are smoothing parameters. As explained in **Appendix E1**, we assign a data point to the class that has the lowest complement parameter estimate.

For oversampling, we chose the random oversampling technique. We looked at some sample sentiment analysis data (Kotzias et. al,. KDD 2015) to contrast the performance of the different classification methods. The data consists of sentences that are to be classified as having a positive or negative sentiment, with equal representation for both classes. We considered three variations of the same dataset with higher degrees of class imbalance: 40% Class 1, 20% Class 1 and less than 10% Class 1 **Appendix E2**. We then compared the performance of Complement Naive Bayes on the imbalanced datasets with that of Bernoulli Naive Bayes on the oversampled datasets. We also looked at Logistic Regression and Support Vector Machine performance on the oversampled datasets in order to gauge the effect on class imbalance on models that did not take a generative approach.

Combined Issues

Lastly, we wanted to explore the robustness of the Naive Bayes classifier when facing a combination of the issues explored above. Combining the methods for all the individual sections, we generated data with no issues, and with medium and high degrees of the combined issues **Appendix F1**. For the data with medium degree of issues, we replaced 25% of the data with missing values, dropped 25% of Class 1 observations, and added moderate skew and collinearity to the data. For the data with high degree of issues, we replaced 50% of the data with missing values, dropped 50% of Class 1 observations and added high skew and collinearity to the data. We then performed mean imputation and random oversampling in order to address the issues of missing values and class imbalance. We compared the performance of Naive Bayes on these datasets with that of Logistic Regression, QDA and Random Forest by graphing the decision boundaries and comparing the accuracy and recall of the models. In order to understand the primary factor that impacts the performance of Naive Bayes, we also tested out one high degree issue at a time.

Results

Dependence Between Variables

It's possible to quantify the specific effect of variable dependence using the method described by Zhang, 2004. We show results empirically by generating data from a Bayes net that uses the model shown in **Figure 1** along with the conditional probability tables (CPTs) and code provided in **Appendix A**. The resulting Bayes network produces examples of both balanced and imbalanced dependence depending on the observation in question.

Each predictor is a Bernoulli random variable. We generated 10,000 samples from the true model and made predictions using a Naive Bayes classifier. When predicting the observation $E_{1,0,1} = [X1 = 1, X2 = 0, X3 = 1]$ which has a true response of Y=1, our Naive Bayes classifier produces $f_{nb} = .781$ which predicts Y=0. Using the formulas above, we can show that this misclassification is explained through the dependency ratios introduced by the CPTs. This is shown under **Appendix A**, "Empirical Evidence".

However, using the same CPTs, the Naive Bayes prediction for a different observation, $E_{1,1,0} = [X1 = 1, X2 = 1, X3 = 0]$, is unaffected because $ddr(X1) = .59$, $ddr(X2) = 1.75$ and $DF(E_{1,1,0}) = 1.03$ which does not change the predicted class.

Classifying XOR Data

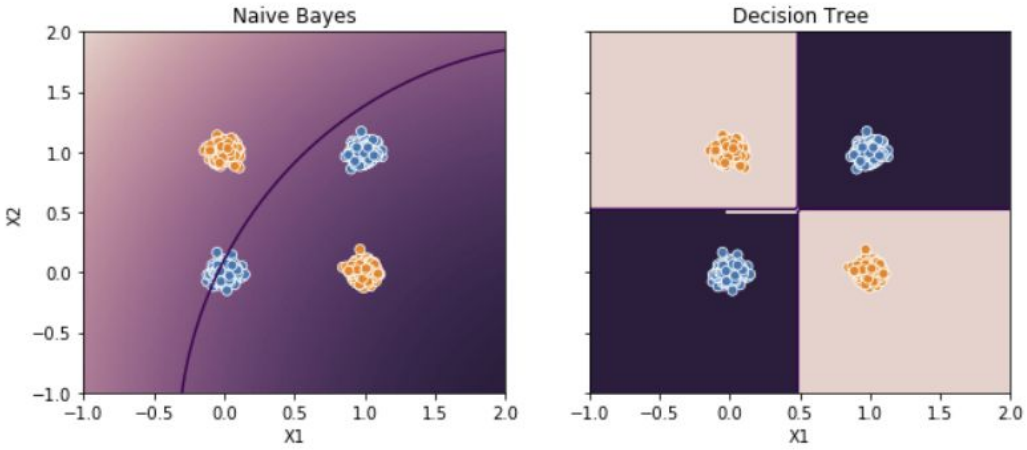


Figure 4: (left) XOR Data classified using Gaussian Naive Bayes (right) Same data classified using Decision Tree model

As shown above, Naive Bayes does a poor job of classifying XOR data. This is because Naive Bayes considers each predictor independent of one another given their class. With XOR data, however, the class label is dependent on a combination of features.

Non-Gaussian Feature Distribution

Clustered Classes

As shown in **Appendix B** below, the Gaussian Naive Bayes model performs comparably to that of logistic regression, LDA, and QDA on the clustered dataset. The plots depict the prediction of class by coloring each {X1, X2} observation with the color of the predicted class. The scatter plot of predictions is plotted in atop the the colored contours pertaining to the modeled likelihood of each class, respectively, with the decision boundary denoted by a solid blue line. Incorrectly predicted points are shown with a red 'X' through them to indicate their mis-classification.

Each of the above method's measured accuracy deviates only by ~0.01 (1%) indicating a very similar degree of performance. Logistic regression and LDA yielded the lowest accuracy due the non-separable nature of both classes and their attempts to draw a linear decision boundary between them. While Naive Bayes and QDA performed almost identically to each other, as their quadratic decision boundaries afforded both models more flexibility producing very similar looking non-linear decision boundaries. Better yet was the accuracy measurement and performance of the Kernelized Naive Bayes model.

Through the use of the gaussian kernel, the Kernel Density Estimation was obtained and used to train a Naive Bayes model. Better representing the "true" distribution of each class in each dimension, the kernelized Naive Bayes model yielded the highest recall of all models, as expected given its ability to best represent the true distribution of each class, as shown in **Appendix B**. The comparison of each method can be found in the same appendix**.**

Skewed Distribution of Features

Similar to that of the clustered example above, a dataset containing two classes represented by two heavily skewed distributions that were then tested by all four of the below methods to characterize the performance of a Gaussian Naive Bayes model. As show in **Appendix B** below**,** it would be fair to expect lowest relative measures of accuracy from the models that generate a linear decision boundary, namely logistic regression and LDA. With another non-separable dataset, it is realistic to expect a model with a more flexible decision boundary to perform better at separating the two classes. Better still was the Kernelized Naive Bayes model, inline with the predictions of the project hypothesis. Performance of each model increased proportionally with respect to class separation, as expected. Both the visualization of the test set predictions and tabulated results of the skewed class adverse case can be found in **Appendix C**.

Missing Values

When first exploring the data that was imputed using the mean, variance dropped dramatically **Appendix D1**. For KNN imputation the variance did decrease but not by the same degree. This gave us some forewarning about potential issues with Naive Bayes performance.

Graphing the accuracy and recall for an increasing proportion of missing values, we see that Naive Bayes seems to perform comparably to other methods with respect to data imputed with the mean, however this changes when looking at recall **Appendix D2**. The recall of Logistic Regression outperforms Naive Bayes in both mean and KNN imputation. SVM is a less consistent technique with missing data as it only relies on the data that are support vectors.

Continuing our analysis with both 20% and 50% missing values, we graphed the decision boundaries for Mean and KNN imputation. We can clearly see from the graphs of 50% missing

values that the reduction in variance due to mean imputation reduces Naive Bayes generalizability **Appendix D3**. Naive Bayes has a much smaller area for one class and will not be able to correctly classify new data of that label which does not fall in that tight circle. In addition, NB performs worse than all the other three models in terms of both accuracy and recall.

Naive Bayes does not seem to be very robust to reduction in variance which is a by-product of many imputation techniques. More robust models might be SVM or Logistic Regression.

Class Imbalance

In order to assess model performance in the presence of class imbalance, we will look at the recall of each model. While all models seem to achieve similar performance when the classes are balanced, the recall of the NB model drops quickly and drastically as the severity of class imbalance increases **Appendix E2**. It falls as low as 2% when the data contains only 20% of Class 1.

Since it ensures that the priors do not induce any bias in the classification, Complement Naive Bayes shows a significant improvement in recall over Bernoulli Naive Bayes as class imbalance increases, and is almost comparable in performance to Logistic Regression and Support Vector Machine classification on oversampled data **Appendix E4**.

However, for our test example, the Bernoulli NB model trained on on oversampled data has the best recall. We can conclude that NB on its own may not be robust to class imbalance. However, if priors are adjusted accordingly, either using oversampling or through training the model on the complement of the data for each class, we can achieve comparable recall to discriminative models such as Logistic Regression and Support Vector Machine using Naive Bayes.

Combined Issues

As expected, Naive Bayes achieves comparable performance to QDA, Logistic Regression and Random Forest when there are no issues with the data, as shown in **Appendix F2**. However, we must note that this is rarely the case with data in the real world. When we scale up the issues, we find that Naive Bayes' performance begins to suffer. While it still has a reasonable accuracy and recall, it's fairly low compared to the other models **Appendix F3**. On the dataset with a high degree of issues, all models begin to perform worse but Naive Bayes remains the worst performing of all models - both in terms of accuracy and recall **Appendix F4**. From the results, it is clear that Naive Bayes is not robust when it comes to a combination of all the issues discussed above. However, the primary factor that impacts the performance of Naive Bayes ends up being high collinearity, as we can see in **Appendix F5**. Therefore, we can conclude that the 'naive' assumption of the model is in fact what breaks it when it faces a combination of these issues.

Discussion

This project was inspired by the frequent reference to Naive Bayes' "surprising" abilities. Our analysis attempted to unpack what makes Naive Bayes' predictive capabilities so surprising. What we learned is that there are several features that make Naive Bayes robust to certain data quality issues. For instance, its presumption of conditional independence between predictors means that useful predictions can be made even when specific combinations of predictor values are not present in the data. When the data is corrected for class imbalance, Naive Bayes also has surprisingly good recall compared to discriminative classifiers.

That being said, it also became clear that Naive Bayes had certain limitations that in many cases made other models more preferred. Data with a high degree of predictor dependence proved to be especially problematic. While our section on Dependence Between Variables showed that dependence is not always problematic, a researcher would not be able to determine this without having access to the true underlying model generating the data.

In addition, Naive Bayes uses the distribution of predictors to make predictions, so as we imputed missing values or oversampled for classes with a low number of observations, we artificially lowered the variance of the predictors. This gave us a tighter decision boundary which would not generalize well to new data.

In order to improve our analysis, we could move away from synthetic data and run our analysis on additional real-world data that exhibits the issues described in our report. For instance, survey data is often missing data values and web analytics conversion data often has class imbalance. Results from this proposed analysis may contradict our findings in certain cases and would stimulate further review.

Ultimately, we judge that Naive Bayes is a simple, efficient model that provides useful predictions in many cases despite underlying data quality issues. For these reasons, our recommendation would be to use it as a starting baseline before evaluating more complicated models.

References

Zhang, Harry. (2004). The Optimality of Naive Bayes. Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004. 2.

Rennie et al. (2003). Tackling the Poor Assumptions of Naive Bayes Text Classifiers. Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003), Washington DC, 2003.

Kotzias et. al,. (2015) From Group to Individual Labels using Deep Features. KDD ‘15. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Appendix

Appendix A - Dependencies Between Variables

Definition of Dependence Distribution Factor

The amount by which a Naive Bayes prediction, f_{nb}, is different than the true model’s prediction, f_b, can be quantified as:

$$f_b = f_{nb} * DF(E)$$

$DF(E)$ is the *dependence distribution factor* for an observation E. This is a function of each variable’s local *dependence derivatives (dd)* and their *dependence derivative ratios (ddr)*. Each variable’s local *dependence derivative* is expressed as a per-class ratio of the variable’s conditional probability including its parent predictors over the conditional probability excluding its parent predictors.

$$dd(x|pa(x)) = \frac{p(x|pa(x), Y)}{p(x|Y)}$$

A variable’s *dependence derivative ratio* is then defined as:

$$ddr(x) = \frac{dd^{+}(x|pa(x))}{dd^{-}(x|pa(x))}$$

Once the ddr for each variable is found, $DF(E)$ is computed as follows where x_i represents the ith predictor:

$$DF(E) = \Pi_{i=1}^{n} ddr(x_i)$$

Dependency Balance

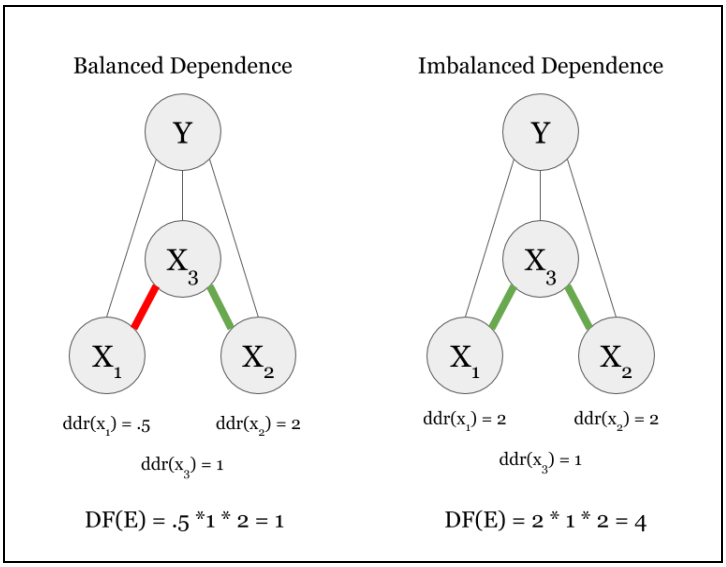


Figure 2: (left) A Bayes net with balanced global dependencies that do not change its prediction given E (right) A bayes net with imbalanced global dependencies

Empirical Evidence

$$f_b = \frac{p(Y=1)p(X_1=1|X_3=1,Y=1)P(X_2=0|X_3=1,Y=1)P(X_3=1|Y=1)}{p(Y=0)p(X_1=1|X_3=1,Y=0)P(X_2=0|X_3=1,Y=0)P(X_3=1|Y=0)} = 1.333$$

$$f_{bn} = \frac{p(Y=1)p(X_1=1|Y=1)p(X_2=0|Y=1)p(X_3=1|Y=1)}{p(Y=0)p(X_1=1|Y=0)p(X_2=0|Y=0)p(X_3=1|Y=0)} = .781$$

$$ddr(X1) = 1.182\,;\, ddr(X2) = 1.444\,;\, ddr(X3) = 1$$
$$DF(E_{1,0,1}) = 1.182 * 1.444 * 1 = 1.706$$

$$f_b = f_{bn} * DF(E_{1,0,1}) = .781 * 1.706 = 1.333$$

Data Generation Code

```
#Probability of Y=1 is .5
#Conditional Probabilities for X3
a_1=.5
b_1=.5
a_0=.5
b_0=.5
#Conditional Probabilities for X1
```

```
x1w_1=.9
x1x_1=.1
x1y_1=.2
x1z_1=.8
x1w_0=.9
x1x_0=.1
x1y_0=.4
x1z_0=.6
#Conditional Probabilities for X2
x2w_1=.2
x2x_1=.8
x2y_1=.6
x2z_1=.4
x2w_0=.4
x2x_0=.6
x2y_0=.3
x2z_0=.7
Y = DiscreteDistribution({1: .5, 0: .5})
X3 =
ConditionalProbabilityTable([[1,1,a_1],[1,0,b_1],[0,1,a_0],[0,0,b_0]],
[Y])
X1 = ConditionalProbabilityTable(
    [[1,1,1,x1w_1],
     [1,1,0,x1x_1],
     [1,0,1,x1y_1],
     [1,0,0,x1z_1],
     [0,1,1,x1w_0],
     [0,1,0,x1x_0],
     [0,0,1,x1y_0],
     [0,0,0,x1z_0]],[Y,X3])
X2 = ConditionalProbabilityTable(
    [[1,1,1,x2w_1],
     [1,1,0,x2x_1],
     [1,0,1,x2y_1],
     [1,0,0,x2z_1],
     [0,1,1,x2w_0],
     [0,1,0,x2x_0],
     [0,0,1,x2y_0],
     [0,0,0,x2z_0]],[Y,X3])

#Calculate ddr for X1 and X2
#X1
x1_w=(x1w_1*(x1w_0*a_0+x1y_0*b_0))/(x1w_0*(x1w_1*a_1+x1y_1*b_1))
```

```
x1_x=(x1x_1*(x1x_0*a_0+x1z_0*b_0))/(x1x_0*(x1x_1*a_1+x1z_1*b_1))
x1_y=(x1y_1*(x1w_0*a_0+x1y_0*b_0))/(x1y_0*(x1w_1*a_1+x1y_1*b_1))
x1_z=(x1z_1*(x1w_0*a_0+x1y_0*b_0))/(x1z_0*(x1w_1*a_1+x1y_1*b_1))

print("ddr X1:",x1_w,x1_x,x1_y,x1_z)

#X2
x2_w=(x2w_1*(x2w_0*a_0+x2y_0*b_0))/(x2w_0*(x2w_1*a_1+x2y_1*b_1))
x2_x=(x2x_1*(x2x_0*a_0+x2z_0*b_0))/(x2x_0*(x2x_1*a_1+x2z_1*b_1))
x2_y=(x2y_1*(x2w_0*a_0+x2y_0*b_0))/(x2y_0*(x2w_1*a_1+x2y_1*b_1))
x2_z=(x2z_1*(x2w_0*a_0+x2y_0*b_0))/(x2z_0*(x2w_1*a_1+x2y_1*b_1))
print("ddr X2:",x2_w,x2_x,x2_y,x2_z)

#Calculate DF given E. We know that ddr(X3) = 1
print("DF(E):",x1_w*x2_w,x1_x*x2_x,x1_y*x2_y,x1_z*x2_z)

sY = State(Y, name="Class")
sX3 = State(X3, name="X3")
sX2 = State(X2, name="X2")
sX1 = State(X1, name="X1")

# Create the Bayesian network object
model = BayesianNetwork("Augmented Naive Bayes Network")

# Add the three states to the network
model.add_states(sY,sX3,sX2,sX1)
model.add_edge(sY, sX3)
model.add_edge(sY, sX2)
model.add_edge(sY, sX1)
model.add_edge(sX3, sX2)
model.add_edge(sX3, sX1)
model.bake()

#Table of all possible observations to sample from to generate data
table=[[0,0,0,0],
    [1,0,0,0],
    [0,1,0,0],
    [1,1,0,0],
    [0,0,1,0],
    [1,0,1,0],
    [0,1,1,0],
    [1,1,1,0],
    [0,0,0,1],
```

```
    [1,0,0,1],
    [0,1,0,1],
    [1,1,0,1],
    [0,0,1,1],
    [1,0,1,1],
    [0,1,1,1],
    [1,1,1,1]]
probabilities=model.probability(table)

#Sample from the indexes of the joint probability table above
elements = list(range(16))
n=10000
idx=np.random.choice(a=elements, size=n, p=probabilities)

#Generate data points using n indexes generated
D=np.asarray(table)[idx]
Y = D[:,0]
train, test = train_test_split(D, shuffle=False)
trainX=train[:,1:4]
testX=test[:,1:4]
trainY=train[:,0]
testY=test[:,0]

#Classification
bnb = BernoulliNB()
bnb.fit(trainX,trainY)
bnb_predictions = bnb.predict(testX)
tn, fp, fn, tp = confusion_matrix(testY,bnb_predictions).ravel()
print("Class balance:",np.sum(Y)/Y.shape[0])

print("\nNB Accuracy:",((tp+tn)/(tn+fp+fn+tp)))
print(confusion_matrix(testY,bnb_predictions))

#Compare to bayes net classifier using probabilities from model
np_predictions =
model.predict(np.concatenate((np.full(test.shape[0],np.nan).reshape(-1
,1),test[:,1:4]),axis=1))
predictions=[]
for i in np_predictions:
    predictions.append(list(i))
bn_predictions=np.asarray(predictions)[:,0]
tn, fp, fn, tp = confusion_matrix(testY,bn_predictions).ravel()
print("\nBayes Net Accuracy:",((tp+tn)/(tn+fp+fn+tp)))
```

```
print(confusion_matrix(testY,bn_predictions))

#Compare to logistic regression
lr = LogisticRegression(solver='lbfgs')
#lr = LinearSVC(C=1.0)
#lr = DecisionTreeClassifier(random_state=0)
#lr = QuadraticDiscriminantAnalysis()
lr.fit(trainX,trainY)
predictions = lr.predict(testX)
tn, fp, fn, tp = confusion_matrix(testY,predictions).ravel()
print("\nLogistic Regression Accuracy:",((tp+tn)/(tn+fp+fn+tp)))
print(confusion_matrix(testY,predictions))
```

Output

```
ddr X1: 1.1818181818181819 0.7777777777777776 0.5909090909090908
1.5757575757575757

ddr X2: 0.4374999999999999 1.444444444444444 1.75 0.5

DF(E): 0.5170454545454545 1.1234567901234562 1.034090909090909
0.7878787878787878

Class balance: 0.4954

NB Accuracy: 0.5488
[[826 443]
 [685 546]]

Bayes Net Accuracy: 0.6212
[[758 511]
 [436 795]]

Logistic Regression Accuracy: 0.5056
[[594 675]
 [561 670]]
```

Appendix B - Non-Gaussian Test Set Visualization

B1. Non-Gaussian Cases considered

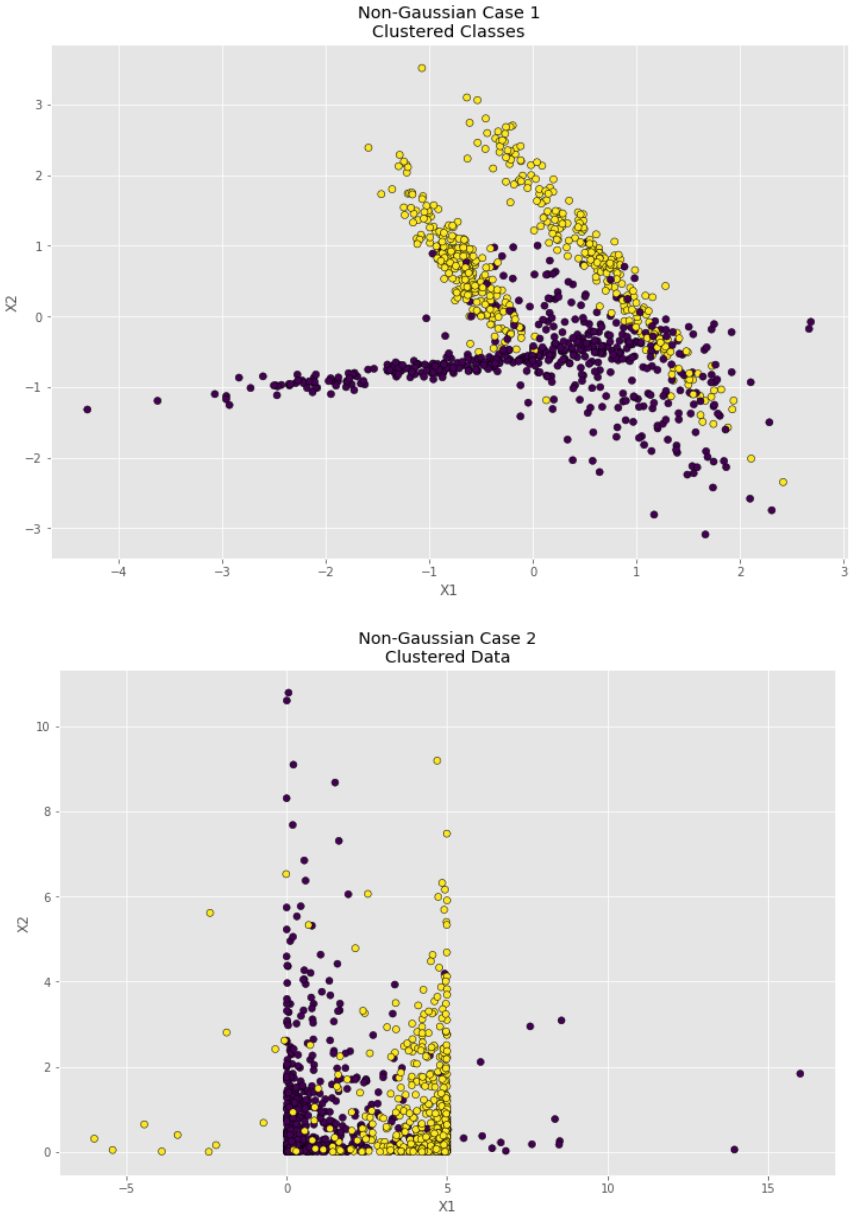
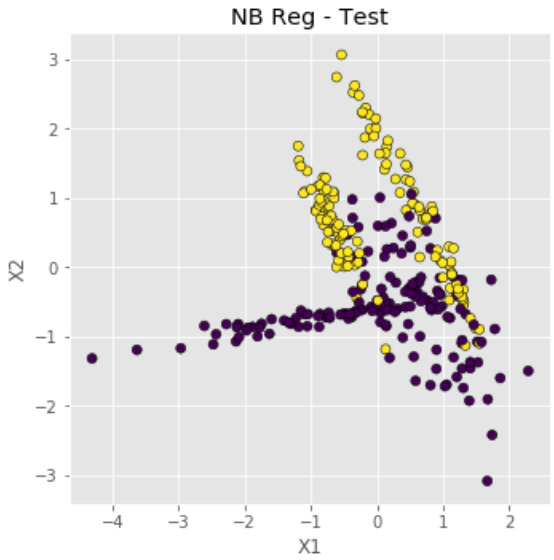
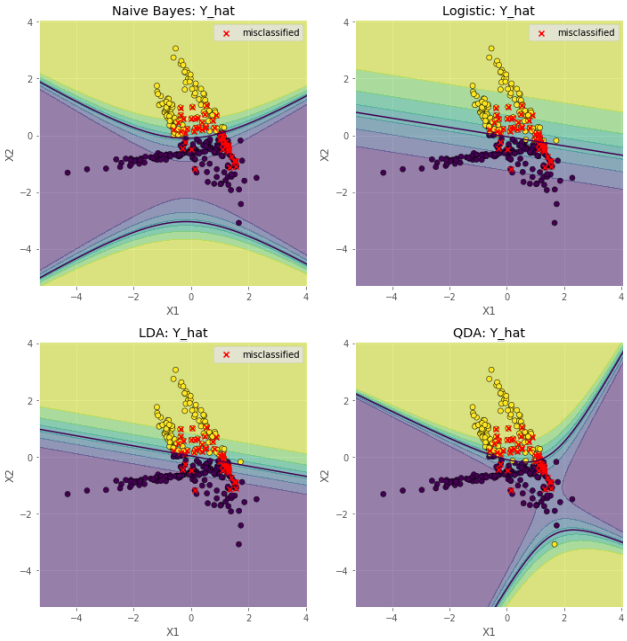


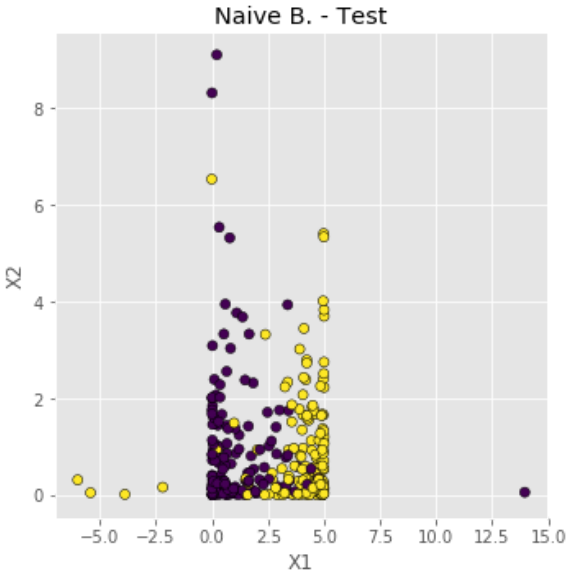
Figure 3: 2 cases considered to represent Non-Gaussian data

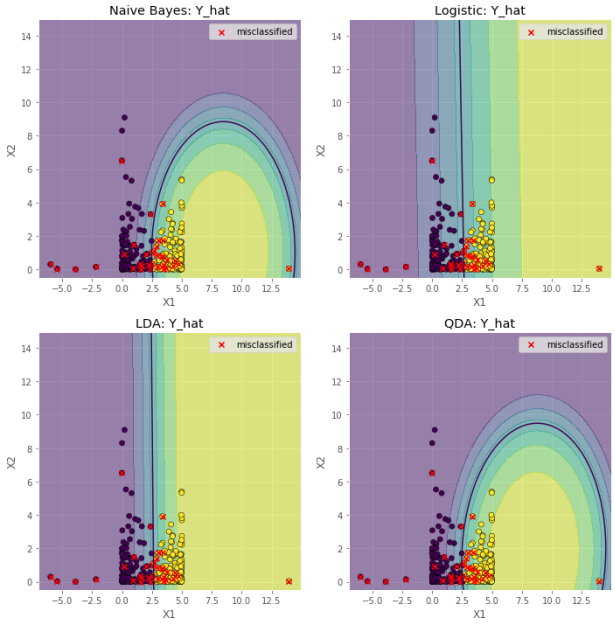
B2. Clustered Classes - Results





B2. Skewed Distribution of Features - Results





B4. Kernel Naive Bayes - Results

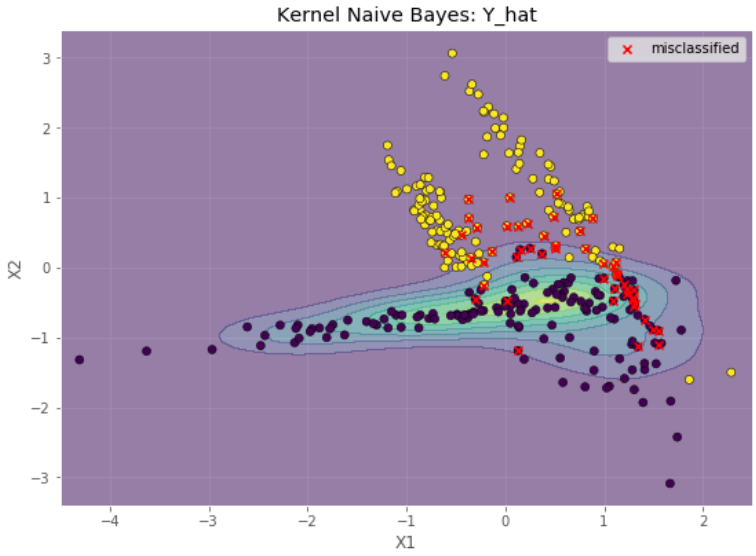


Figure 5: Kernel Naive Bayes model predictions shown over the contour plot of the joint probability distribution of Class 1

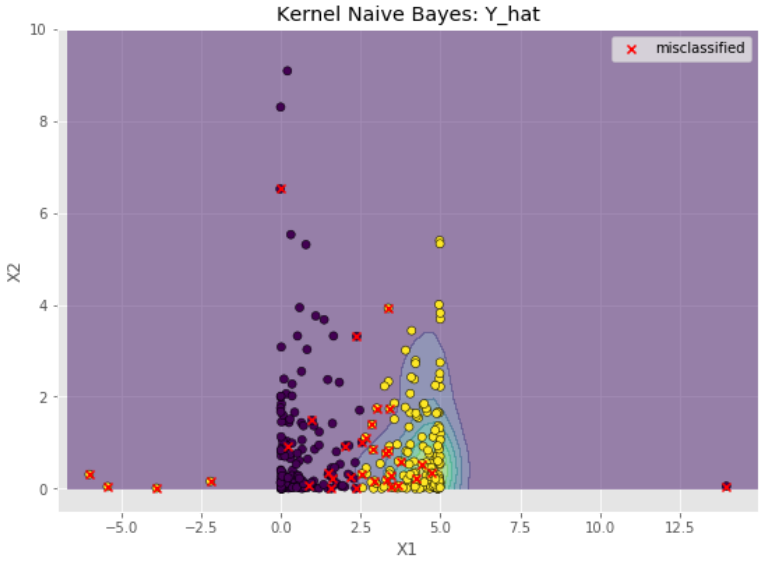


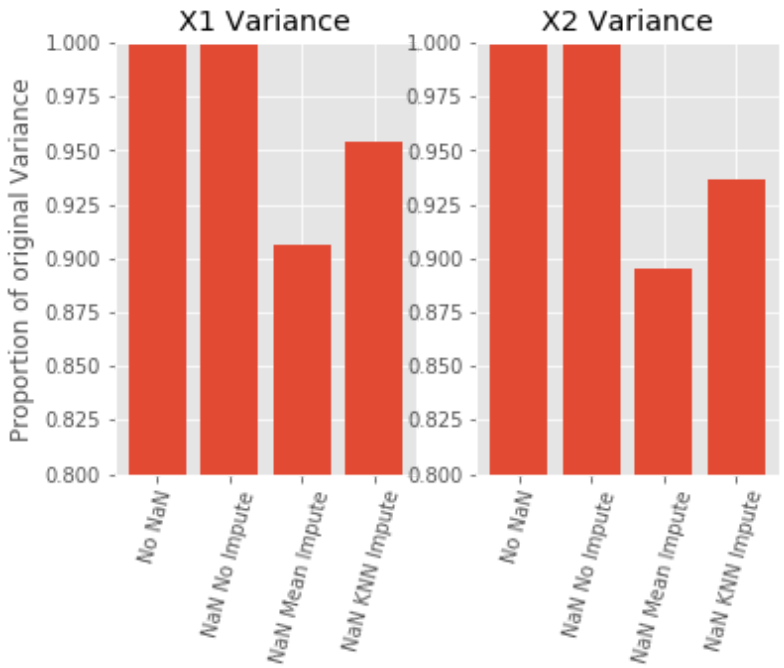
Figure 6: Kernel Naive Bayes model predictions overlayed on the contour of the joint probability distribution of class 1

Appendix C - Non-Normal Data: Results Table

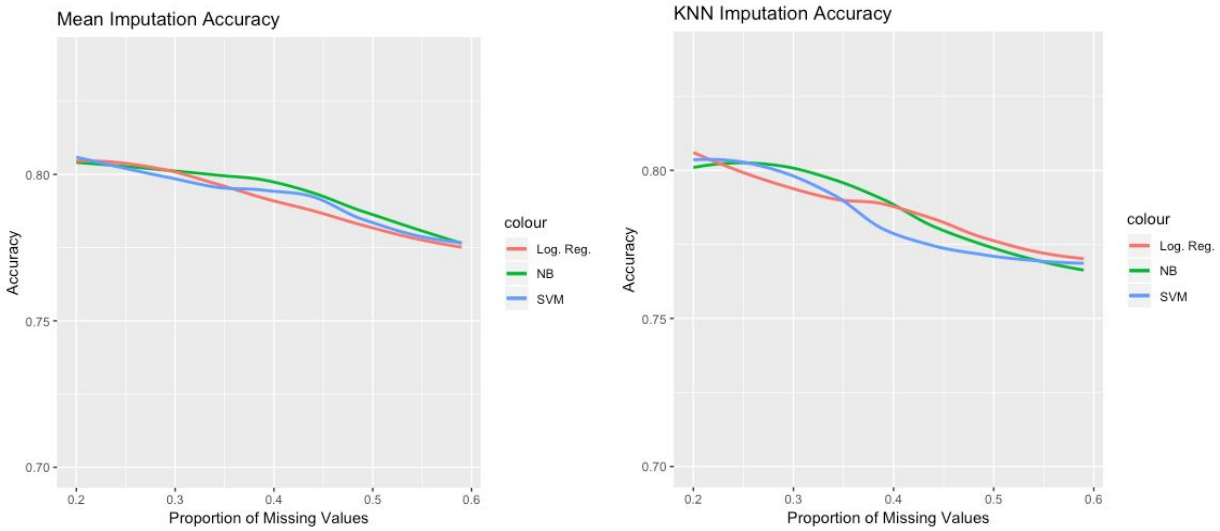
	Clustered Classes		**Skewed Class Dist.**	
	Accuracy	**Recall**	**Accuracy**	**Result**
Naive Bayes	0.81	0.82	0.83	0.82
Logistic Regression	0.83	0.85	0.83	0.85
LDA	0.81	0.81	0.83	0.81
QDA	0.83	0.85	0.83	0.85
Kernel Naive Bayes	0.85	0.86	0.90	0.93

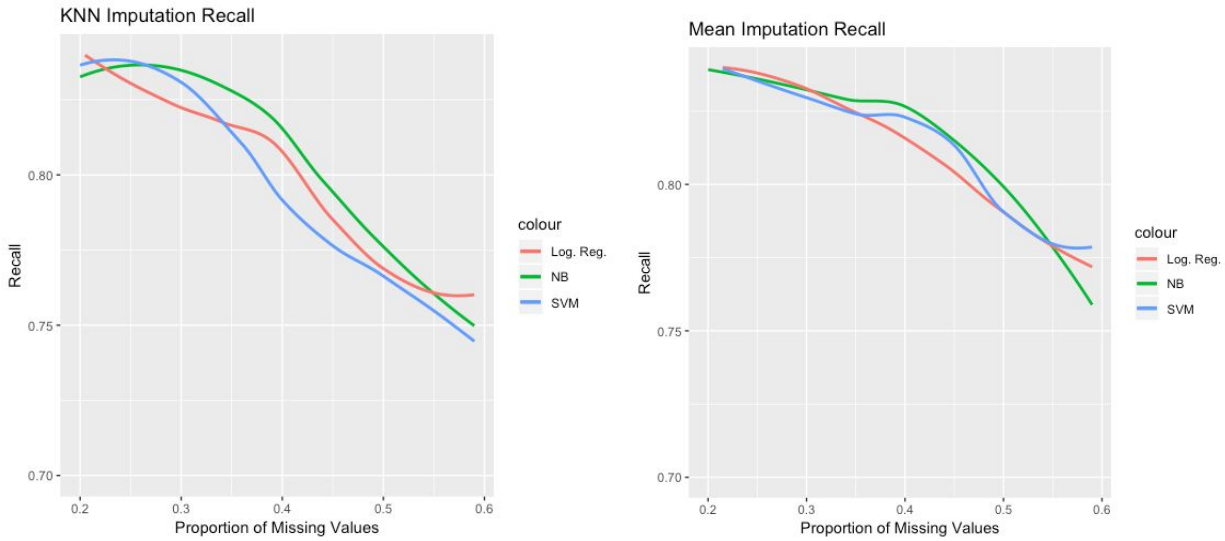
Appendix D - Missing Data

D1. Variance with Imputation

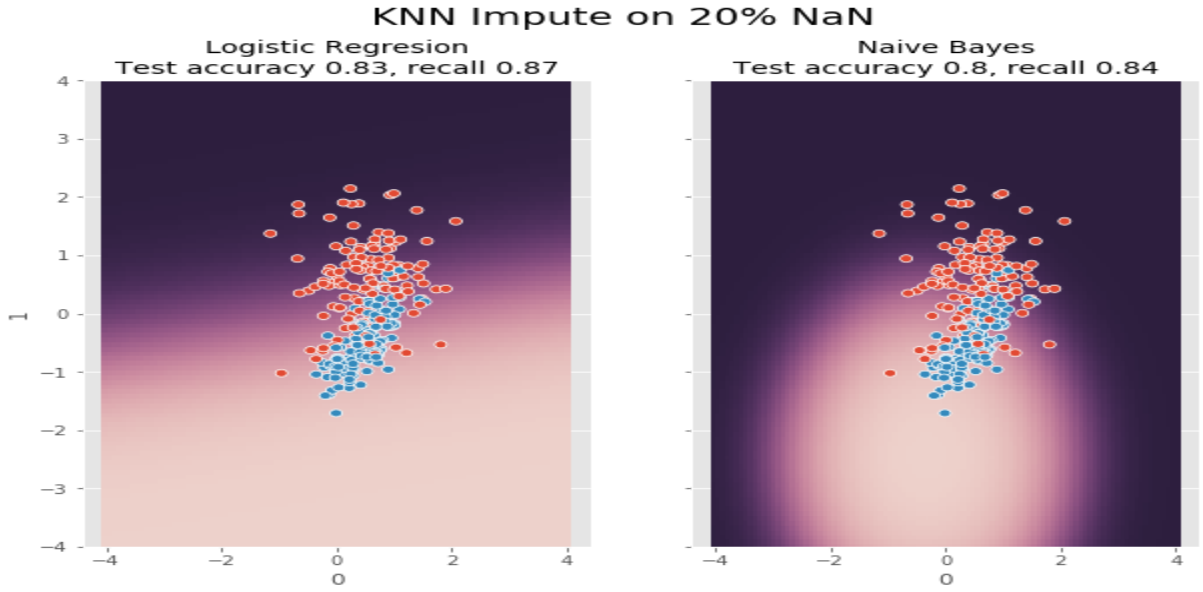


D2. Accuracy vs Recall of Imputation

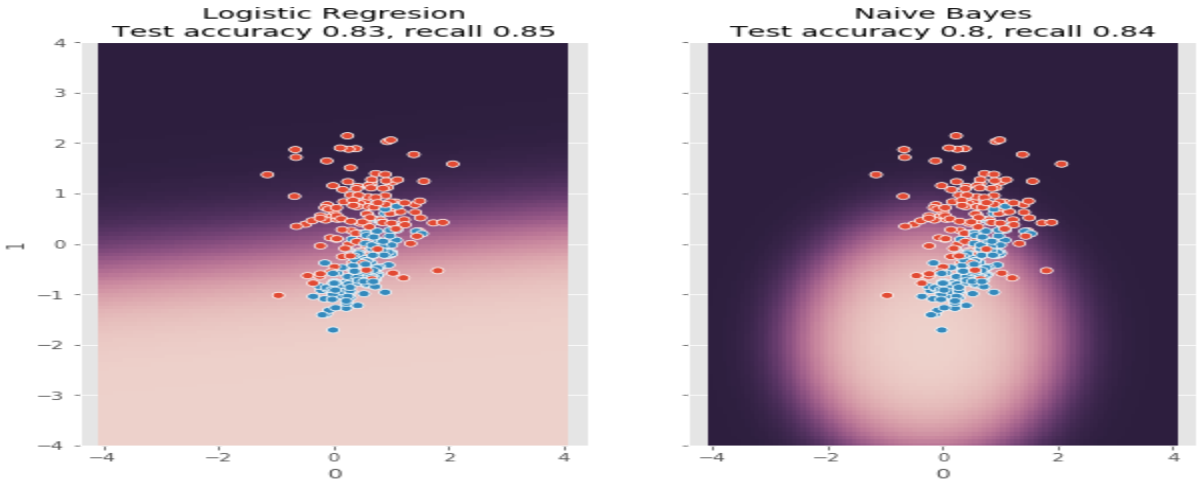




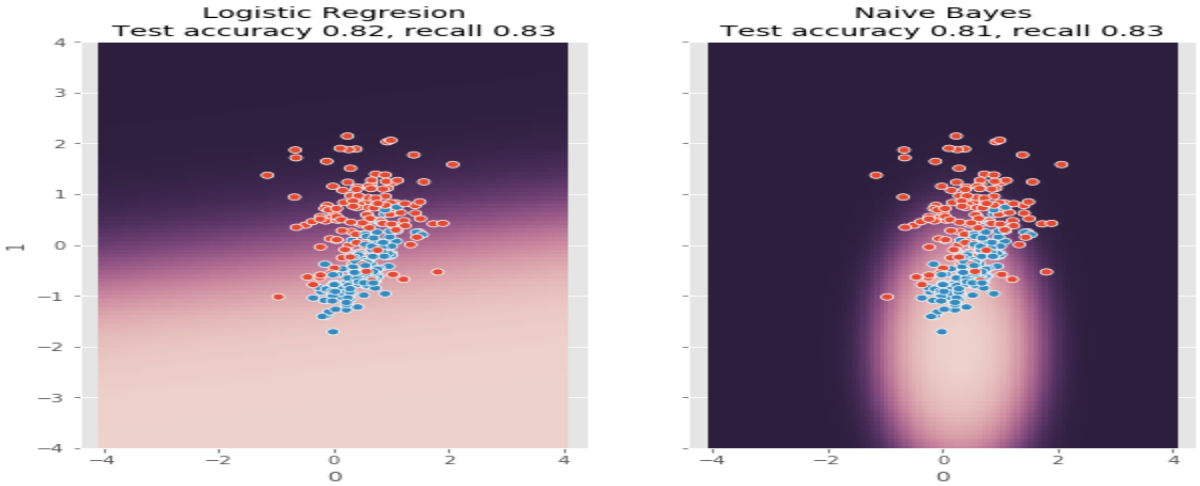
D3. Decision Boundaries



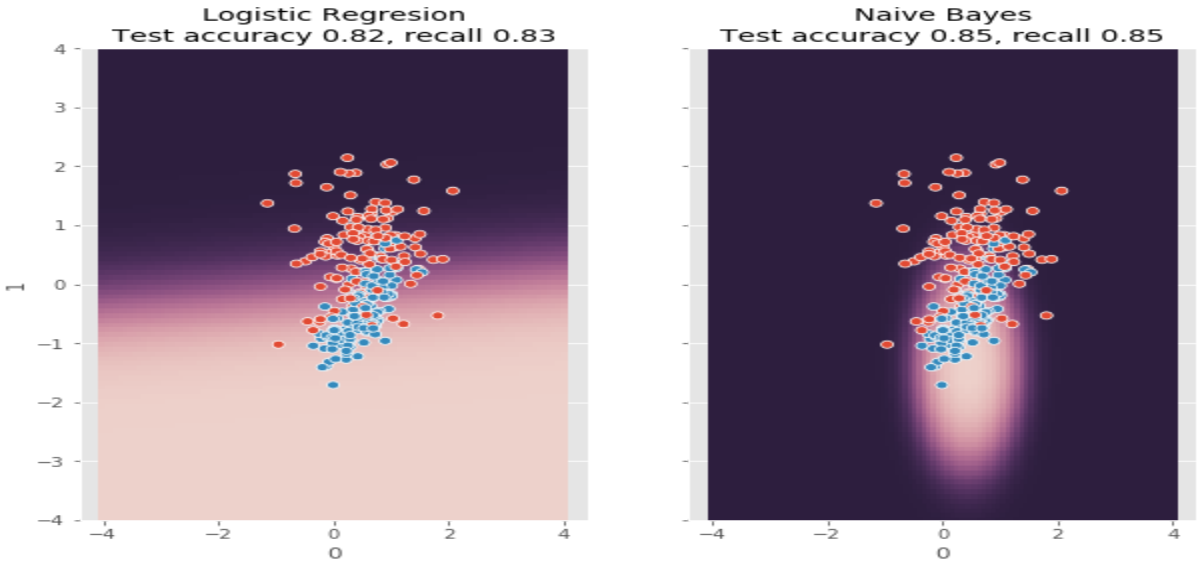
Mean Impute on 20% NaN



KNN Impute on 50% NaN



Mean Impute on 50% NaN



Appendix E - Class Imbalance

E1. Classification for Complement Naive Bayes

$$l_{CNB}(d) = argmax_c \left[\log p(\overrightarrow{\theta}_c) - \sum_i f_i \log \frac{N_{\tilde{c}i} + \alpha_i}{N_{\tilde{c}} + \alpha} \right]$$

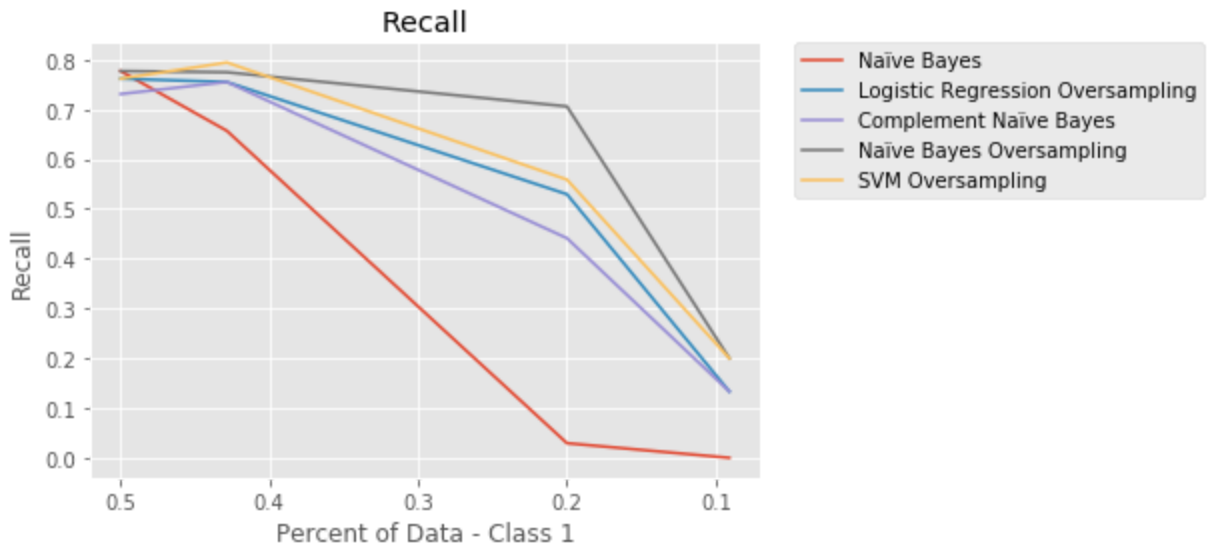
In complement Naive Bayes, we train the model for each class on the complement of the class. For instance, in a problem with classes 1, 2 and 3, we train the model for class 1 on data for classes 2 and 3, and so on. We estimate the model weights for each class, and assign a data point to its poorest complement match.

E2. Proportions of Classes in Data

E3. Performance Results - Recall

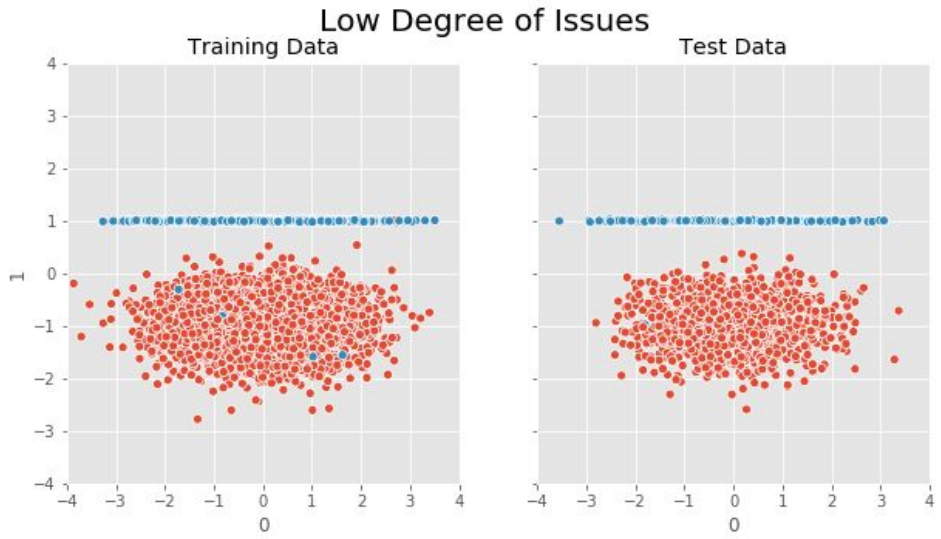
Dataset Class 0 - Class 1	Bernoulli NB	Complement NB	Bernoulli NB (Oversampling)	Logistic Regression (Oversampling)	SVM (Oversampling)
50-50	0.77	0.73	0.77	0.76	0.76
60-40	0.65	0.75	0.77	0.75	0.79
80-20	0.03	0.44	0.70	0.52	0.55
90-10	0.0	0.13	0.20	0.13	0.20

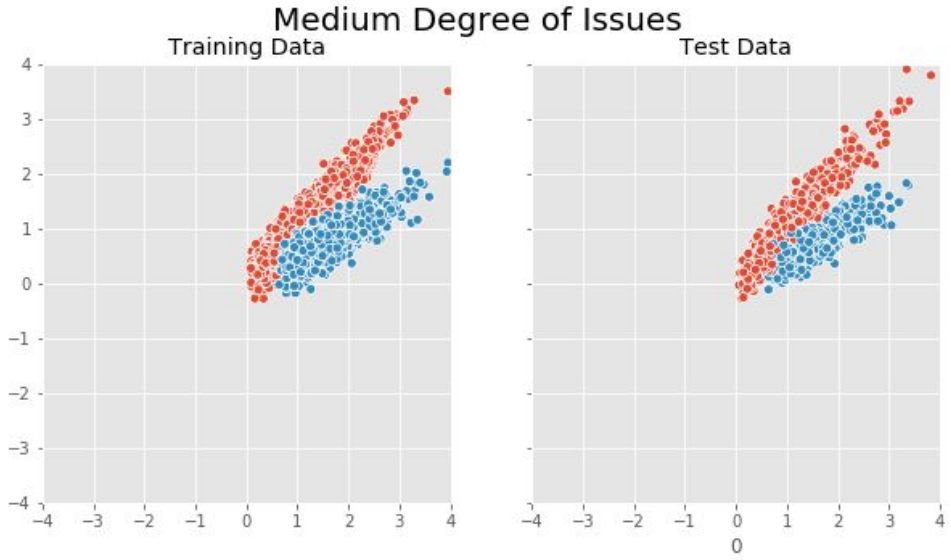
E4 - Comparing Recall performance

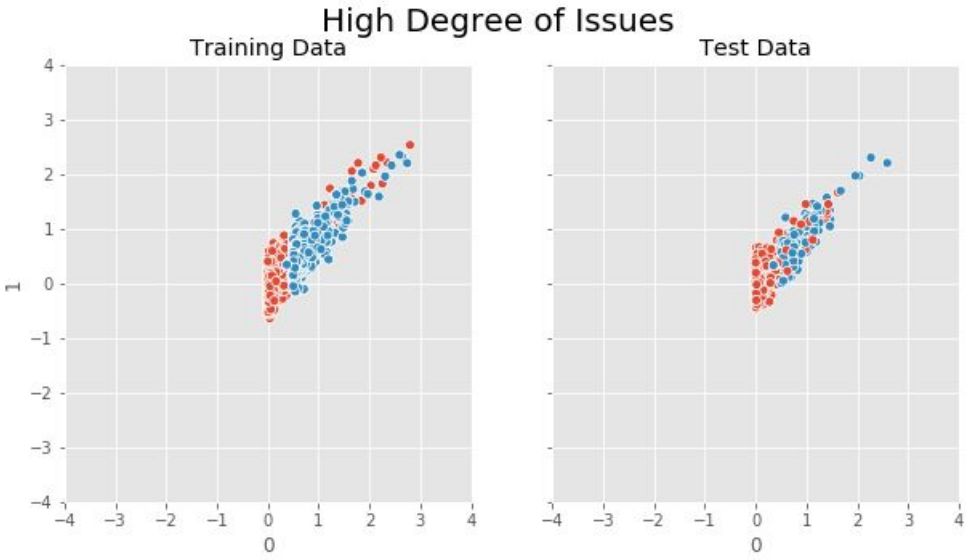


Appendix F - Combined Issues

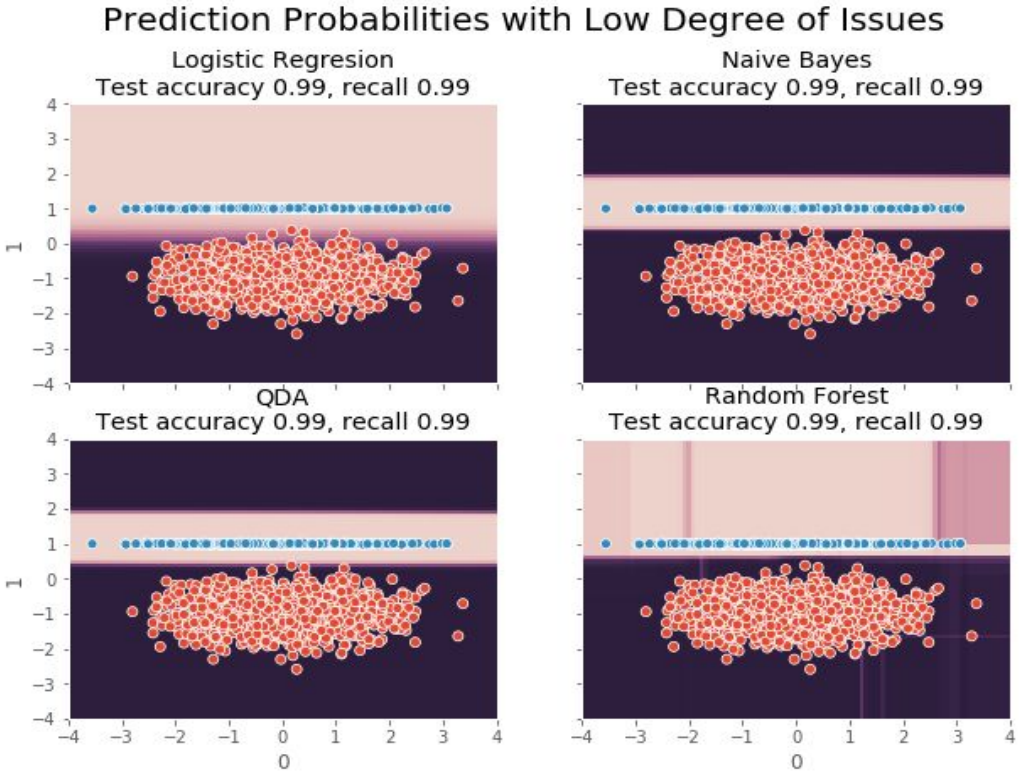
F1. Dataset Visualizations



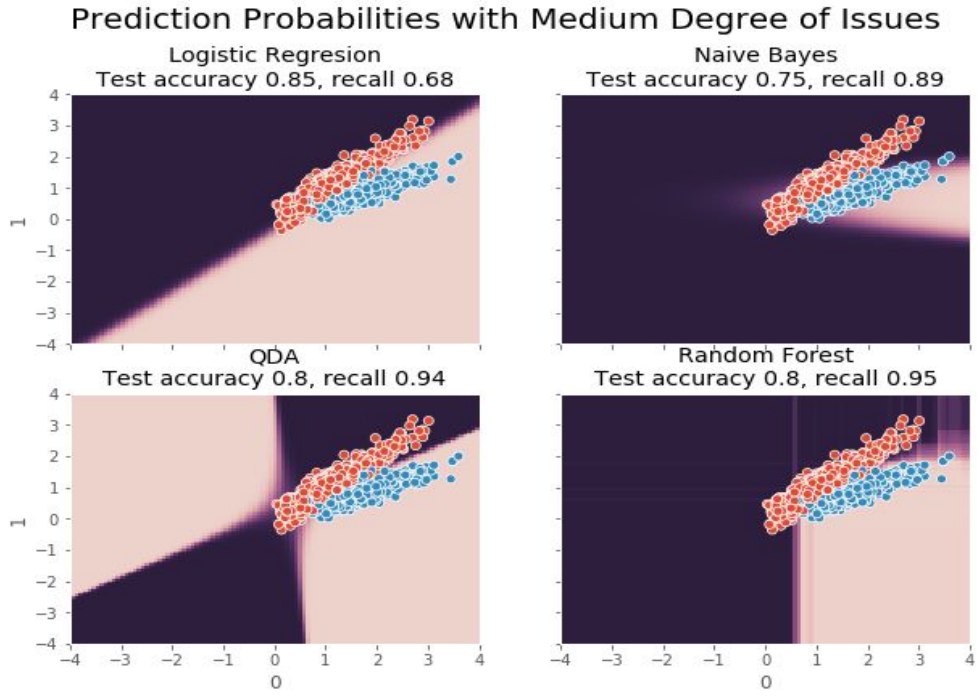




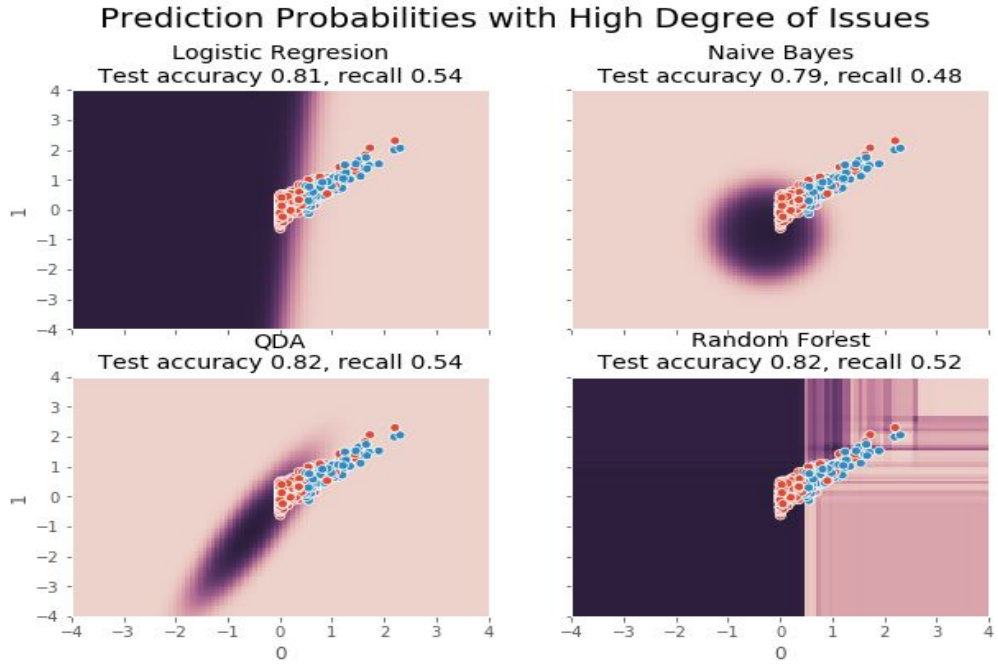
F2. Comparing Combined - Low Degree



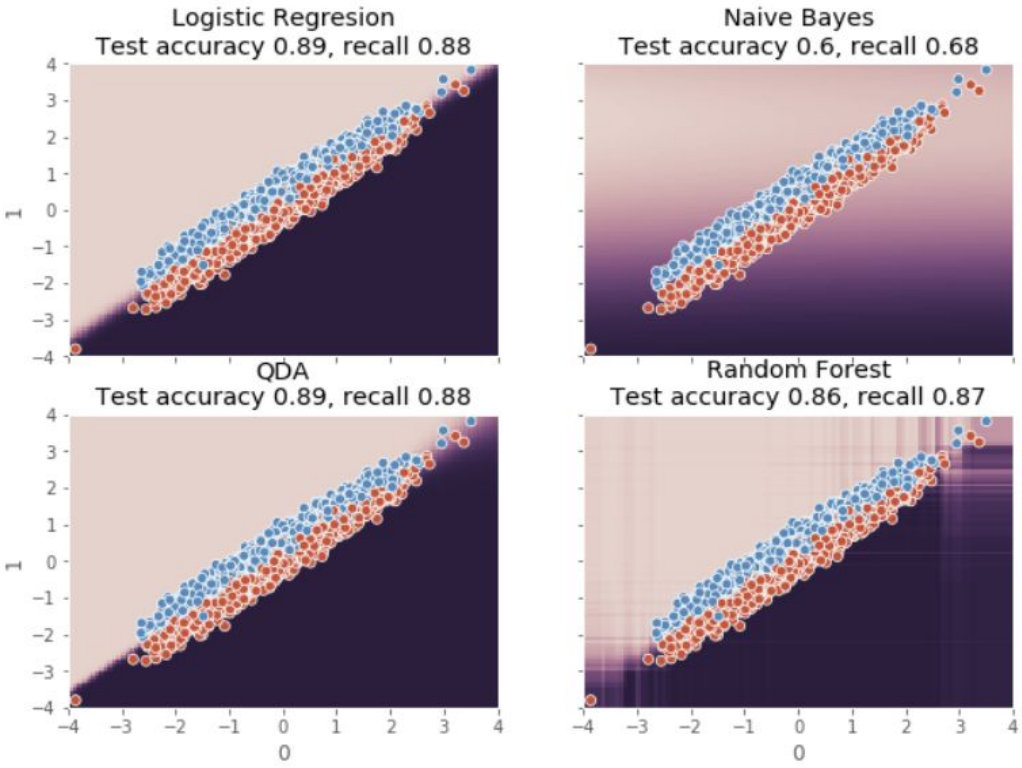
F3. Comparing Combined - Medium Degree



F4. Comparing Combined - High Degree



F5. Collinearity Comparison



Statement of Contributions

In the last separate page, state explicitly how each member of the group contributed to the project.

Andrew Reichel (MSDS 2020) - Clustered Classes(Code, Methods & Results), Skewed Feature Distributions(Code, Methods & Results),
Smruthi Ramesh (MSDS 2020) - Class Imbalance (Code, Methods & Results), Combined Issues (Methods & Results)
Ross Marino (MSDS 2020) - Missing Values (Code, Methods & Results), Discussion, Final Report Edit
Adam Ribaudo (MS DS 2020) - Dependence Between Variables (Code, Methods & Results), XOR Data, Discussion, Combined Issues (Code)