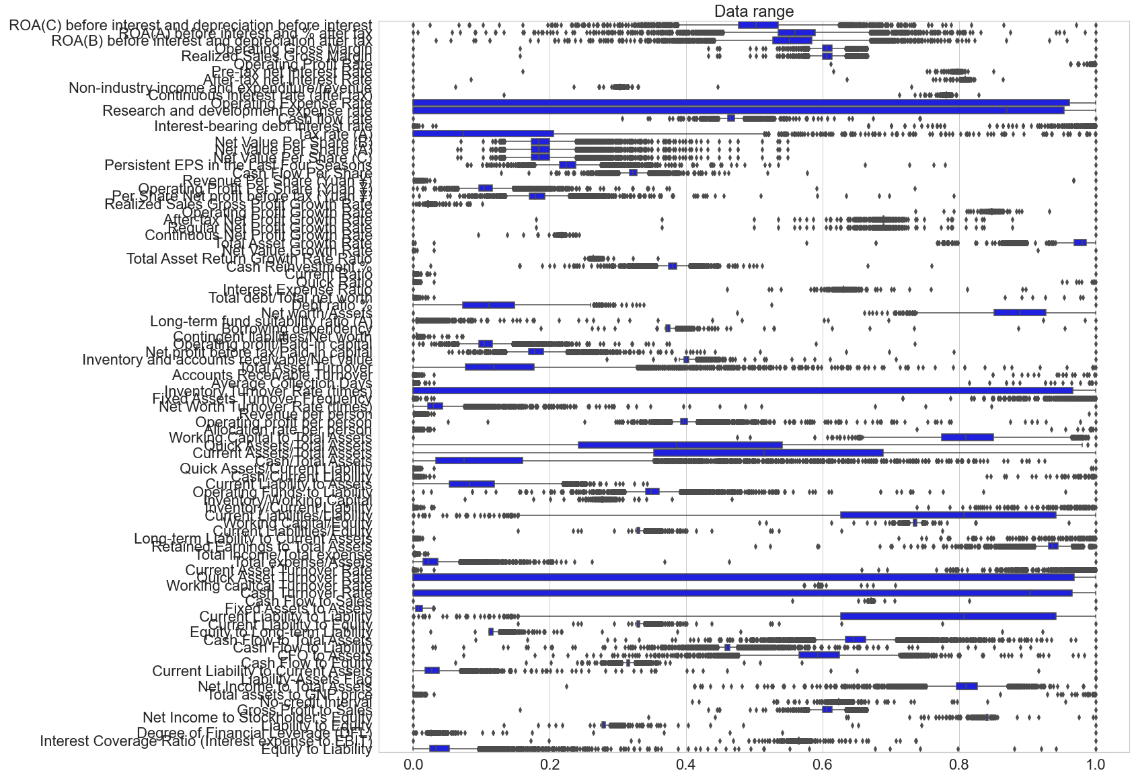
<https://www.kaggle.com/fedesoriano/company-bankruptcy-prediction>

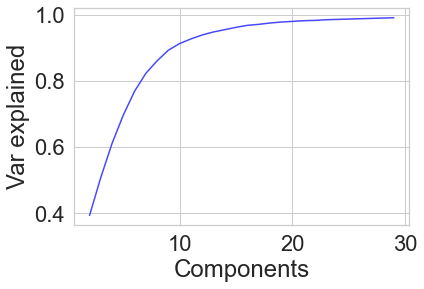
**Data preparation, exploration, visualization**

There are 95 features with binary dependent variable ‘Bankrupt?’

1. No duplicated data
2. No missing data
3. Drop ‘Net Income Flag’ constant column.
4. There are two types of features, ones that are between 0 and 1 and ones that go from 0 to 1e10
   1. For those extreme features, I use log1p, then minmax\_scale so all features are between 0 and 1



1. Principle components analysis
   1. A simple PCA shows that around 10 components can already explain 90% of the variance in the response



**Review research design and modeling methods**

1. Train\_test\_split into 80-20 train and test sets, with the stratify parameter so that the proportion of bankruptcies will be the same in the two sets as it is in the total dataset. Since ‘Bankrupt?’ is so imbalanced (30:1 for 0s:1s), it is necessary to do this to prevent a random split eliminating too many of the minority class in the test or training sets.
2. I can also fit PCA onto the training set, then transform both training and test sets to conduct the regression. Of note here is, many of the published Kaggle notebooks appear to do the PCA transformation on the entire dataset before splitting, which seems to be a major data leak faux pas.
3. I will also consider oversampling methods on the minority class, with both random oversampling and SMOTE.
4. Since many variables are highly skewed, I will use PowerTransformer in a pipeline.

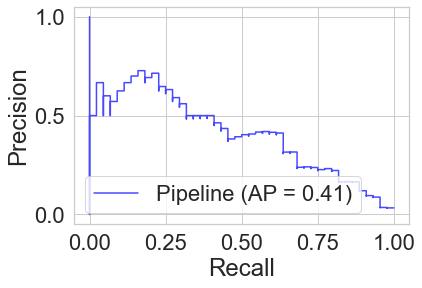
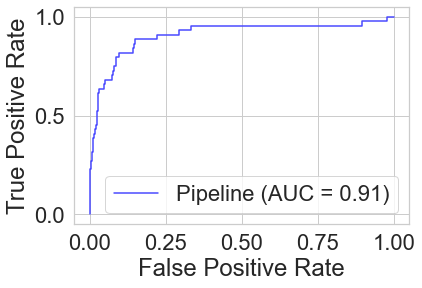
Each model will be evaluated with a confusion matrix, precision, accuracy, recall, and f1\_scores. F1 scores in particular come in several flavors of weighting and I will report them all for comparison.

1. Logistic Regression will be done once with no special parameters, and with Elastic Net regularization through LogisticRegressionCV
2. SVMs hyperparameter tuning
   1. kernels, regularization, gamma, class weighting
3. Naïve Bayes will be through GaussianNB with tuning on var\_smoothing
4. LDA and QDA will also be attempted but with less refinement

**Review results, evaluate models**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **FN** | **FP** | **TP** | **TN** | **FPR** | **TPR** | **ACC** | **PREC** | **F1** | **F1 micro** | **F1 macro** | **F1 weighted** |
| **LogisticReg** | 33 | 8 | 11 | 1312 | 0.01 | 0.25 | 0.97 | 0.58 | 0.35 | 0.97 | 0.67 | 0.96 |
| **Tuned LogReg** | 25 | 28 | 19 | 1292 | 0.02 | 0.43 | 0.96 | 0.4 | 0.42 | 0.96 | 0.7 | 0.96 |
| **Log PCA** | 44 | 0 | 0 | 1320 | 0 | 0 | 0.97 | n/a | 0 | 0.97 | 0.49 | 0.95 |
| **Log random over** | 0 | 610 | 44 | 710 | 0.46 | 1 | 0.55 | 0.07 | 0.13 | 0.55 | 0.41 | 0.68 |
| **Log SMOTE** | 1 | 483 | 43 | 837 | 0.37 | 0.98 | 0.65 | 0.08 | 0.15 | 0.65 | 0.46 | 0.76 |
| **SVC** | 44 | 0 | 0 | 1320 | 0 | 0 | 0.97 | n/a | 0 | 0.97 | 0.49 | 0.95 |
| **Tuned SVC** | 21 | 34 | 23 | 1286 | 0.03 | 0.52 | 0.96 | 0.4 | **0.46** | 0.96 | 0.72 | 0.96 |
| **SVC PCA** | 44 | 0 | 0 | 1320 | 0 | 0 | 0.97 | n/a | 0 | 0.97 | 0.49 | 0.95 |
| **SVC random over** | 5 | 312 | 39 | 1008 | 0.24 | 0.89 | 0.77 | 0.11 | 0.2 | 0.77 | 0.53 | 0.84 |
| **SVCSMOTE** | 5 | 227 | 39 | 1093 | 0.17 | 0.89 | 0.83 | 0.15 | 0.25 | 0.83 | 0.58 | 0.88 |
| **NB** | 12 | 114 | 32 | 1206 | 0.09 | 0.73 | 0.91 | 0.22 | 0.34 | 0.91 | 0.64 | 0.93 |
| **Tuned NB** | 21 | 58 | 23 | 1262 | 0.04 | 0.52 | 0.94 | 0.28 | 0.37 | 0.94 | 0.67 | 0.95 |
| **NB PCA** | 44 | 0 | 0 | 1320 | 0 | 0 | 0.97 | n/a | 0 | 0.97 | 0.49 | 0.95 |
| **NB random over** | 10 | 134 | 34 | 1186 | 0.1 | 0.77 | 0.89 | 0.2 | 0.32 | 0.89 | 0.63 | 0.92 |
| **NBSMOTE** | 8 | 197 | 36 | 1123 | 0.15 | 0.82 | 0.85 | 0.15 | 0.26 | 0.85 | 0.59 | 0.9 |
| **LDA** | 30 | 17 | 14 | 1303 | 0.01 | 0.32 | 0.97 | 0.45 | 0.37 | 0.97 | 0.68 | 0.96 |
| **Tuned LDA** | 28 | 22 | 16 | 1298 | 0.02 | 0.36 | 0.96 | 0.42 | 0.39 | 0.96 | 0.69 | 0.96 |
| **LDA PCA** | 44 | 0 | 0 | 1320 | 0 | 0 | 0.97 | n/a | 0 | 0.97 | 0.49 | 0.95 |
| **LDA random over** | 6 | 202 | 38 | 1118 | 0.15 | 0.86 | 0.85 | 0.16 | 0.27 | 0.85 | 0.59 | 0.89 |
| **LDASMOTE** | 6 | 176 | 38 | 1144 | 0.13 | 0.86 | 0.87 | 0.18 | 0.29 | 0.87 | 0.61 | 0.91 |
| **QDA** | 25 | 76 | 19 | 1244 | 0.06 | 0.43 | 0.93 | 0.2 | 0.27 | 0.93 | 0.62 | 0.94 |
| **Tuned QDA** | 30 | 30 | 14 | 1290 | 0.02 | 0.32 | 0.96 | 0.32 | 0.32 | 0.96 | 0.65 | 0.96 |
| **QDA PCA** | 41 | 21 | 3 | 1299 | 0.02 | 0.07 | 0.95 | 0.12 | 0.09 | 0.95 | 0.53 | 0.95 |
| **QDA random over** | 11 | 185 | 33 | 1135 | 0.14 | 0.75 | 0.86 | 0.15 | 0.25 | 0.86 | 0.59 | 0.9 |
| **QDA SMOTE** | 12 | 214 | 32 | 1106 | 0.16 | 0.73 | 0.83 | 0.13 | 0.22 | 0.83 | 0.56 | 0.89 |

With the tuned SVC model:



**Results**

1. PCA in all models caused precision to drop to almost 0, where no positives were predicted.
2. Both Random Oversampling and SMOTE methods caused the maximum recall, or TPR, but also caused many much higher false positives.
3. The tuned SVC model resulted in the highest unweighted F1 score, with around 0.5 in both precision and recall.
4. It also had the highest F1\_micro score.
5. Among the oversampling-based models, Naïve Bayes had the lowest FP rate.
6. The untuned Logistic regression model had the highest precision, due to its low FP rate.

**Exposition, problem description, and management recommendations**

If I was a bank trying to determine which companies to lend to, the purpose of this model would be to avoid companies with high likelihood of bankruptcies, I would choose the model with a high recall in order to minimize risk.

If I’m a hedge fund who vacations on Epstein’s island, I’d want a model to find companies with a very high likelihood of bankruptcy, so a simple logistic regression model will work for a low rate of false positives.