Course Project

Ramatu, Emily, Evan, Melody

An Empirical Comparison:

Classical, Boosting, and Deep Learning Models

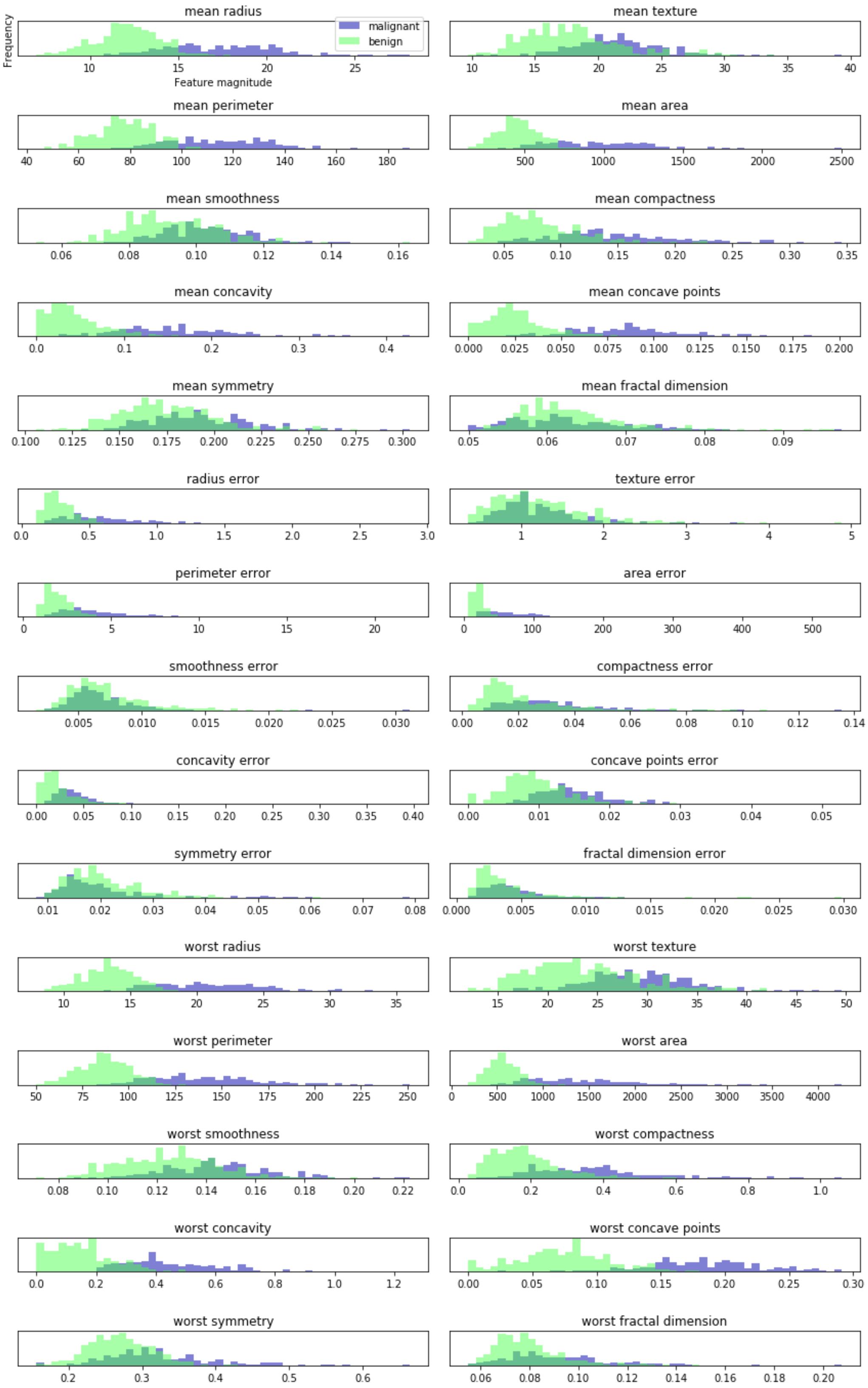
To conduct a comparison of classification across our different models, we chose 3 different datasets that are of varying size and complexity. This report will break down the comparison first by giving some explanation of the datasets and exploratory data analysis, then we will discuss each individual model's performance on each dataset. Finally, we will summarize a comparison between the models on each dataset. In this way, we hope to test the 'no free lunch' theorem, which stipulates that no one model will be the best selection for every dataset.

Dataset Descriptions

Wisconsin Breast Cancer – a small, unbalanced binary classification dataset

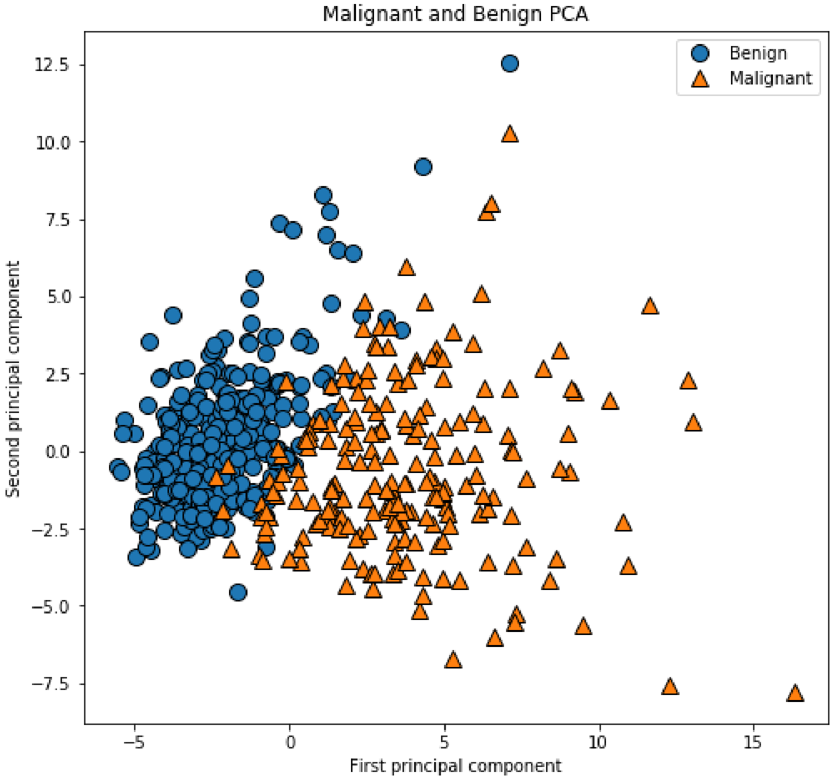
The cancer dataset was the same one that was used in class for our first individual assignments. The dataset presents columns with different analytical measures of various tumors and classifies them into benign or malignant tumors. Although favoured toward the negative class, this dataset presents a relatively balanced spread of data, with 357 benign cases and 212 malignant cases. We chose this dataset because each member has previously analyzed it with classic Machine Learning techniques, and so it would be interesting to revisit the dataset armed with new techniques. Thus, our will also have the contextual familiarity having analyzed it with basic machine learning models.

```
There are 569 rows in the dataset
There are 31 columns (features) in the dataset
There are 0 nulls in the dataset

Composition of the Dataset's Classes:

Breast Cancer Diagnosis: 37.3 %
No Breast Cancer: 62.7 %
```


mean radius
malignant
benign
Frequency
Feature magnitude
10 15 20 25
mean texture
10 15 20 25 30 35 40
mean perimeter
40 60 80 100 120 140 160 180
mean area
500 1000 1500 2000 2500
mean smoothness
0.06 0.08 0.10 0.12 0.14 0.16
mean compactness
0.05 0.10 0.15 0.20 0.25 0.30 0.35
mean concavity
0.0 0.1 0.2 0.3 0.4
mean concave points
0.000 0.025 0.050 0.075 0.100 0.125 0.150 0.175 0.200
mean symmetry
0.100 0.125 0.150 0.175 0.200 0.225 0.250 0.275 0.300
mean fractal dimension
0.05 0.06 0.07 0.08 0.09
radius error
0.0 0.5 1.0 1.5 2.0 2.5 3.0
texture error
1 2 3 4 5
perimeter error
0 5 10 15 20
area error
0 100 200 300 400 500
smoothness error
0.005 0.010 0.015 0.020 0.025 0.030
compactness error
0.00 0.02 0.04 0.06 0.08 0.10 0.12 0.14
concavity error
0.00 0.05 0.10 0.15 0.20 0.25 0.30 0.35 0.40
concave points error
0.00 0.01 0.02 0.03 0.04 0.05
symmetry error
0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08
fractal dimension error
0.000 0.005 0.010 0.015 0.020 0.025 0.030
worst radius
10 15 20 25 30 35
worst texture
15 20 25 30 35 40 45 50
worst perimeter
50 75 100 125 150 175 200 225 250
worst area
0 500 1000 1500 2000 2500 3000 3500 4000
worst smoothness
0.08 0.10 0.12 0.14 0.16 0.18 0.20 0.22
worst compactness
0.0 0.2 0.4 0.6 0.8 1.0
worst concavity
0.0 0.2 0.4 0.6 0.8 1.0 1.2
worst concave points
0.00 0.05 0.10 0.15 0.20 0.25 0.30
worst symmetry
0.2 0.3 0.4 0.5 0.6
worst fractal dimension
0.06 0.08 0.10 0.12 0.14 0.16 0.18 0.20

The per-class feature histogram shows that there is a lot of overlap of the 2 distributions between say, texture error benign/malignant, so we will not glean much information from that feature. However, looking at worst concave points, for instance, we can see that there is a lot of variance between the 2 populations, so we will glean a lot more information from that feature.

Running a PCA analysis on the Breast Cancer dataset, we can retain 95% + of the variation reducing to 1 feature.



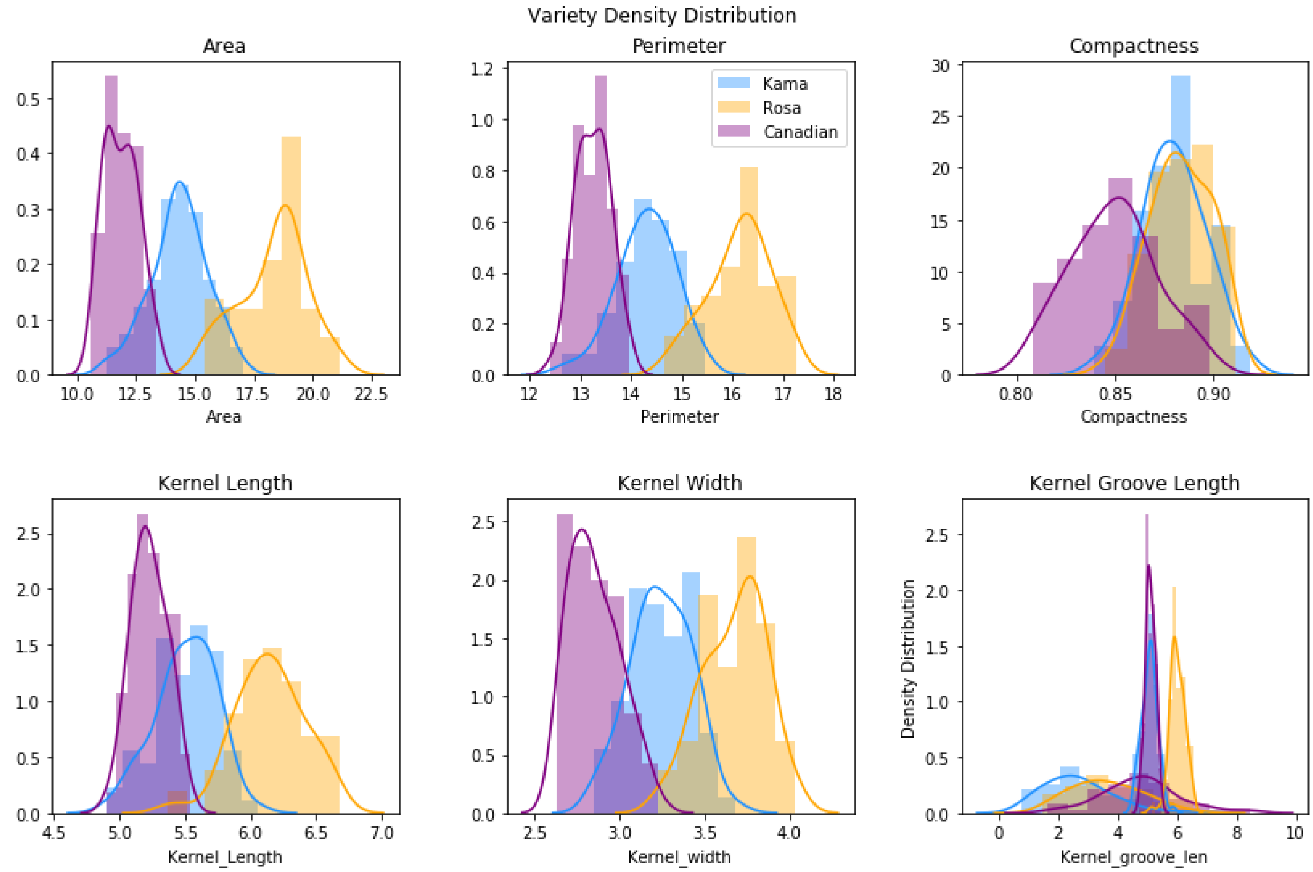
We can see the separation of classes across the first two principal components, they separate reasonably cleanly.

Wheat Dataset – a small, balanced multiclass classification dataset

The wheat dataset is smaller, with only 210 rows of data, divided into kernel data used to determine the type of wheat the kernel belongs to. Like the iris classification dataset, the dependent variable is classification of 3 different types of wheat. With each variety comprising 33% of the rows, this is an evenly balanced dataset that would provide a different outcome than the comparatively unbalanced cancer dataset.

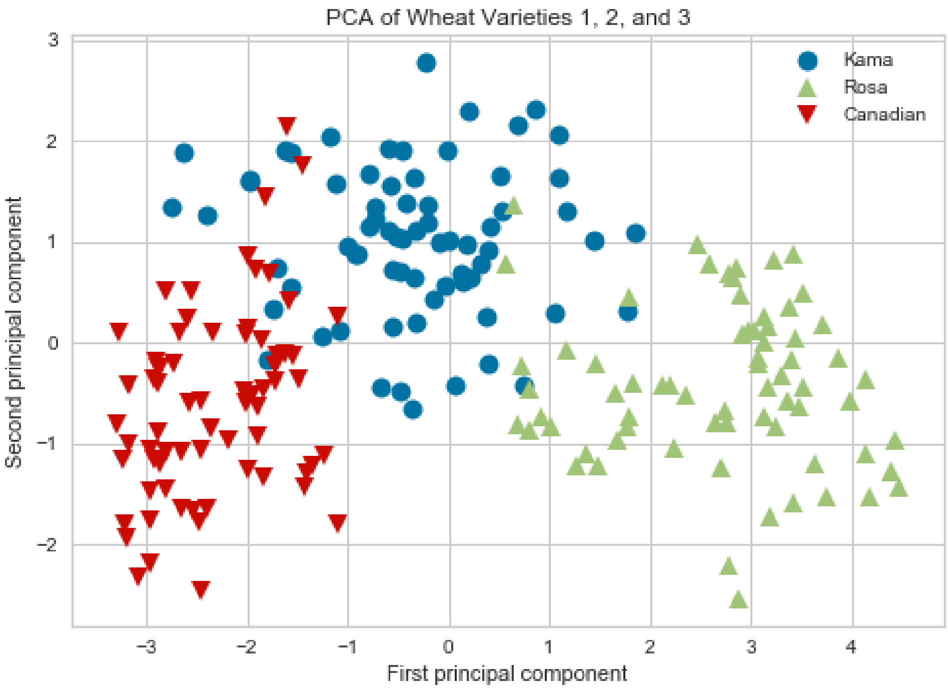
```
There are 210 rows in the dataset
There are 8 columns (features) in the dataset
There are 0 nulls in the dataset

Composition of the Dataset's Classes:

Wheat Variety 1: 33.3 %
Wheat Variety 2: 33.3 %
Wheat Variety 3: 33.3 %
```

Here, can see the density distribution across the features of the 3 different kernel types:



Running a PCA analysis, we can retain the following variability:

```
Total variability accounted for with 3 features: 99.87%
```



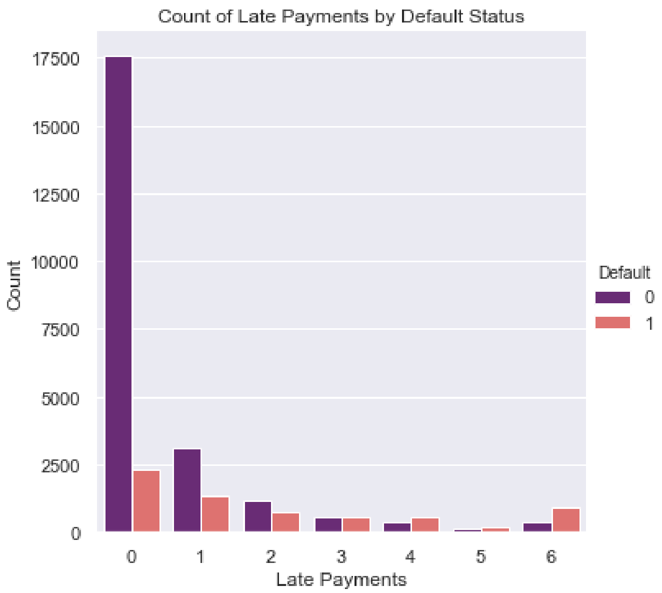
These 3 classes separate quite cleanly across the first 2 principal components.

Credit Card Dataset – a massive, unbalanced binary classification dataset

Our third choice is the Taiwan Credit Card dataset, which attempts to predict credit default status under certain criteria such as amount of credit given, marital status, bill amounts across time, etc. The dataset is heavily skewed to the negative class – that is, non-defaults— and may therefore be an interesting case to run oversampling looking to better determine default cases. Compared to our other two datasets, this is much larger, at 30,000 rows. This size proved computationally challenging for most of our models.

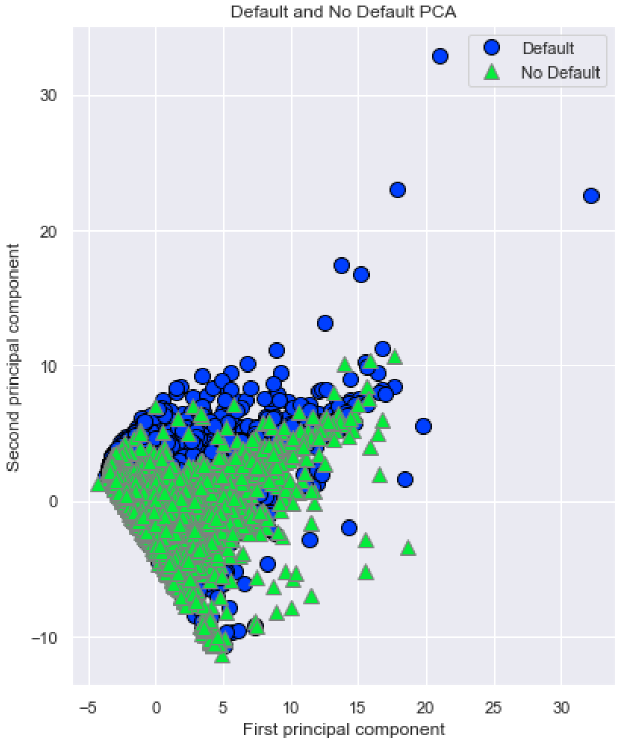
```
There are 30000 rows in the dataset
There are 24 columns (features) in the dataset
There are 0 nulls in the dataset
Composition of the Dataset's Classes:

Default: 22.1 %
No Default: 77.9 %
```

We can see the count of late payments, vs the client’s default status:



After running PCA on the credit card dataset, we can retain 98% of the dataset’s variability with 8 of the 23 features:

```
Number of features required to retain 98% of the dataset's variability:  8
```



---| Classic Machine Learning Models |---

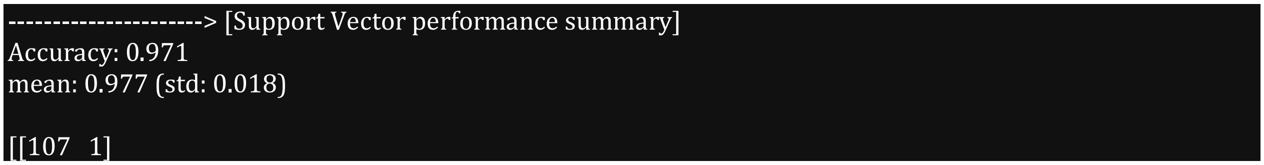
Support Vector Machine (SVM)

SVM is a classifier that finds an optimal hyperplane that maximises the margin between two classes, finding an optimal line to separate the boundary between classes. A linear model which can classify data which is not linearly separable, SVM enables us to classify additional elements in the delineated classes. We can use kernel method to separate data for nonlinear data.

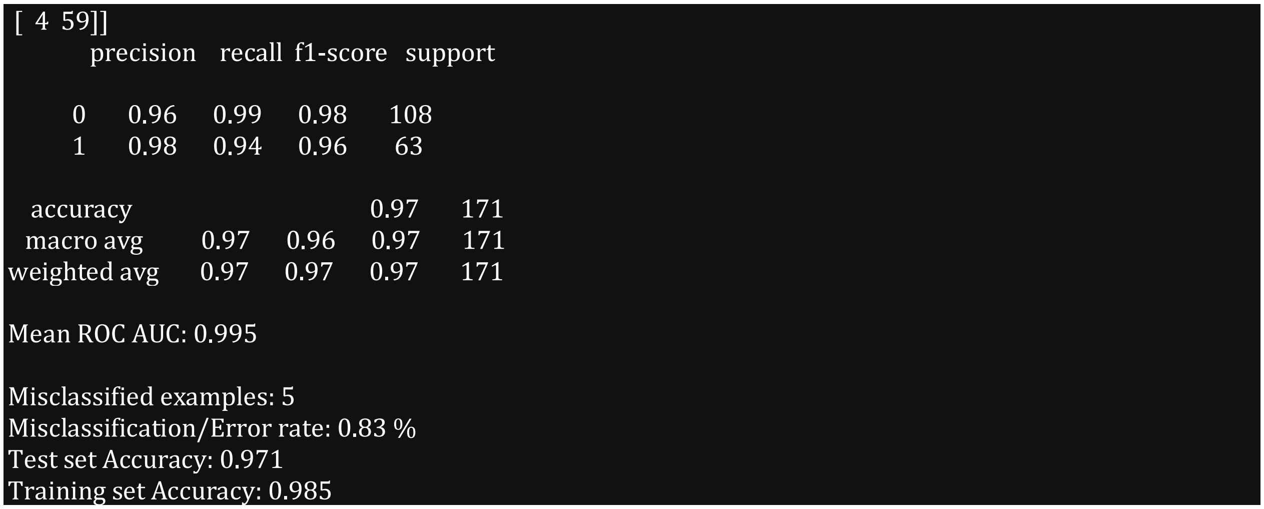
In SVM, support vectors are a subset of the training points defined as decision boundaries. So, to decide for a new point the distance to each support vector is measured. The classification decision is made based on the distances to the support vectors that were learned during the training.

SVM on Breast Cancer Dataset

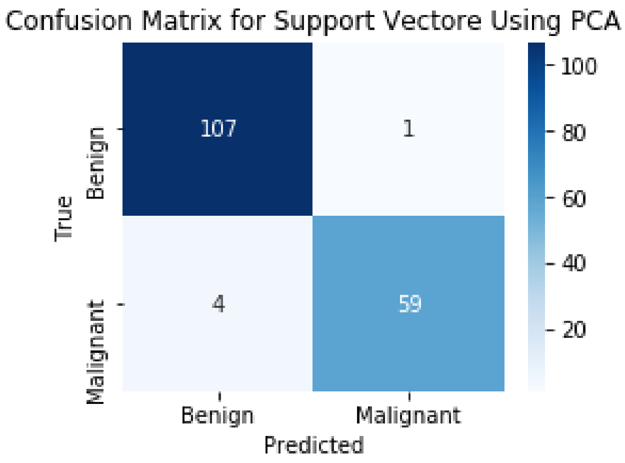
After feeding the dataset through a pipeline which scales, applies PCA, fits the model, and searches for the best parameters to use, our SVM model yielded the following results:



```
---------------------> [Support Vector performance summary]
Accuracy: 0.971
mean: 0.977 (std: 0.018)

[[107   1]
```



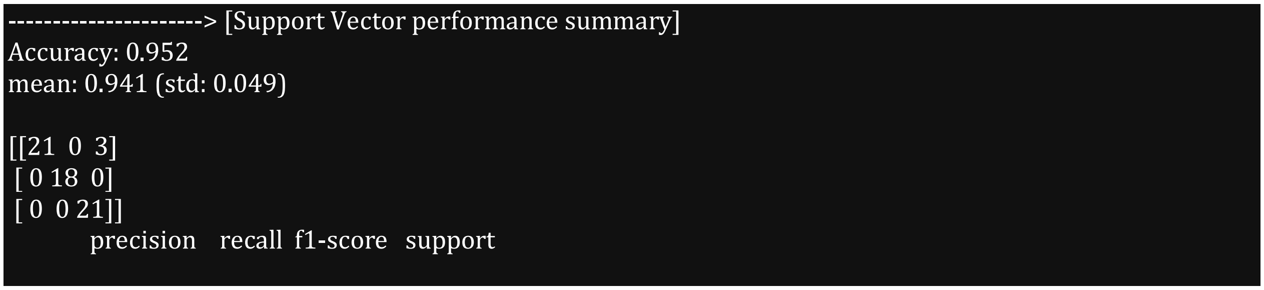
```
 [ 4  59]]
            precision    recall  f1-score   support

        0       0.96      0.99      0.98       108
        1       0.98      0.94      0.96        63

   accuracy                           0.97       171
  macro avg       0.97      0.96      0.97       171
weighted avg       0.97      0.97      0.97       171

Mean ROC AUC: 0.995

Misclassified examples: 5
Misclassification/Error rate: 0.83 %
Test set Accuracy: 0.971
Training set Accuracy: 0.985
```



SVM performed quite well on the Breast Cancer dataset, with cross-validated accuracy at 97.1%. Despite the reasonable unbalanced dataset, it still had admirable recall for the positive class, at 0.94. It had more trouble predicting malignant when the case was actually benign than the opposite.

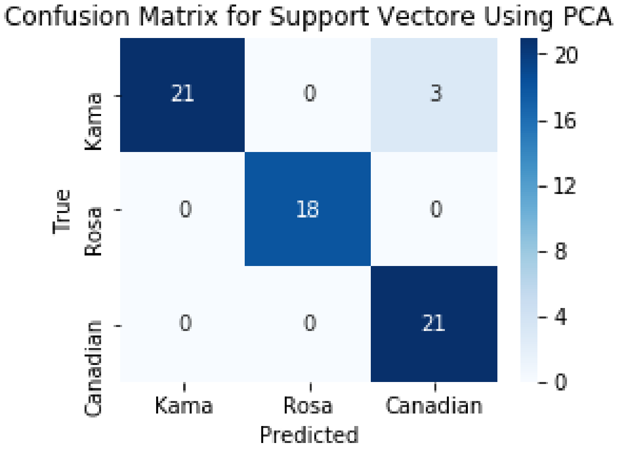
SVM on Wheat Dataset:

The pipelined SVM model performed well as a multiclass classification problem:



```
---------------------> [Support Vector performance summary]
Accuracy: 0.952
mean: 0.941 (std: 0.049)

[[21  0  3]
 [ 0 18  0]
 [ 0  0 21]]
            precision    recall  f1-score   support
```

```
        1     1.00      0.88      0.93       24
        2     1.00      1.00      1.00       18
        3     0.88      1.00      0.93       21

   accuracy                           0.95       63
  macro avg        0.96      0.96      0.96       63
weighted avg       0.96      0.95      0.95       63

Misclassified examples: 3
Misclassification/Error rate: 2.3 %
Test set Accuracy: 0.952
Training set Accuracy: 0.980
```

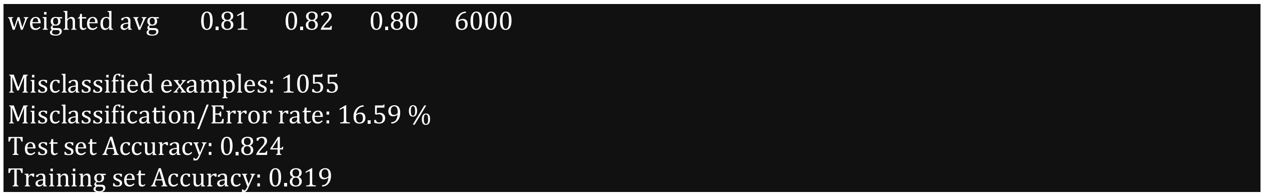

Confusion Matrix for Support Vectore Using PCA

As we can see with the Confusion Matrix, SVM only had difficulty with Kama kernel, where it misclassified 3 predicting them as Canadian. Otherwise, it had 0 misclassifications.

SVM on Credit Card Dataset:

As with our other models, SVM had difficulty with the credit card dataset due to its imbalance. After feeding through the pipeline, we obtained the following results:

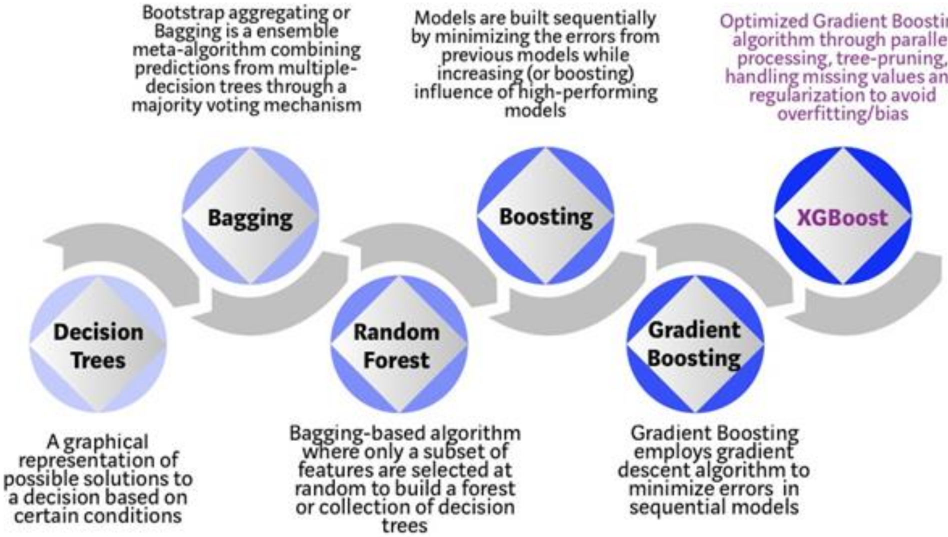
```
---------------------> [Support Vector performance summary]
Accuracy: 0.824
mean: 0.818 (std: 0.006)

[[4491  182]
 [ 873  454]]
           precision    recall  f1-score   support

        0     0.84      0.96      0.89     4673
        1     0.71      0.34      0.46     1327

   accuracy                          0.82      6000
  macro avg       0.78      0.65      0.68      6000
```



```
weighted avg      0.81      0.82      0.80      6000

Misclassified examples: 1055
Misclassification/Error rate: 16.59 %
Test set Accuracy: 0.824
Training set Accuracy: 0.819
```

SVM is slow to train even on a small data set, so the Credit Card dataset was so computationally expensive that it caused 3 of our members' laptops to (figuratively) catch fire. Only one team member's powerful desktop machine could handle training the SVM. This time challenge is a big drawback to using this model; practically speaking, ease-of-use is an important factor in deciding which models to use on a given dataset. So, for large datasets, SVM is not a good option even if it yields good accuracy.

Tree-based Bagging-Boosting Methods

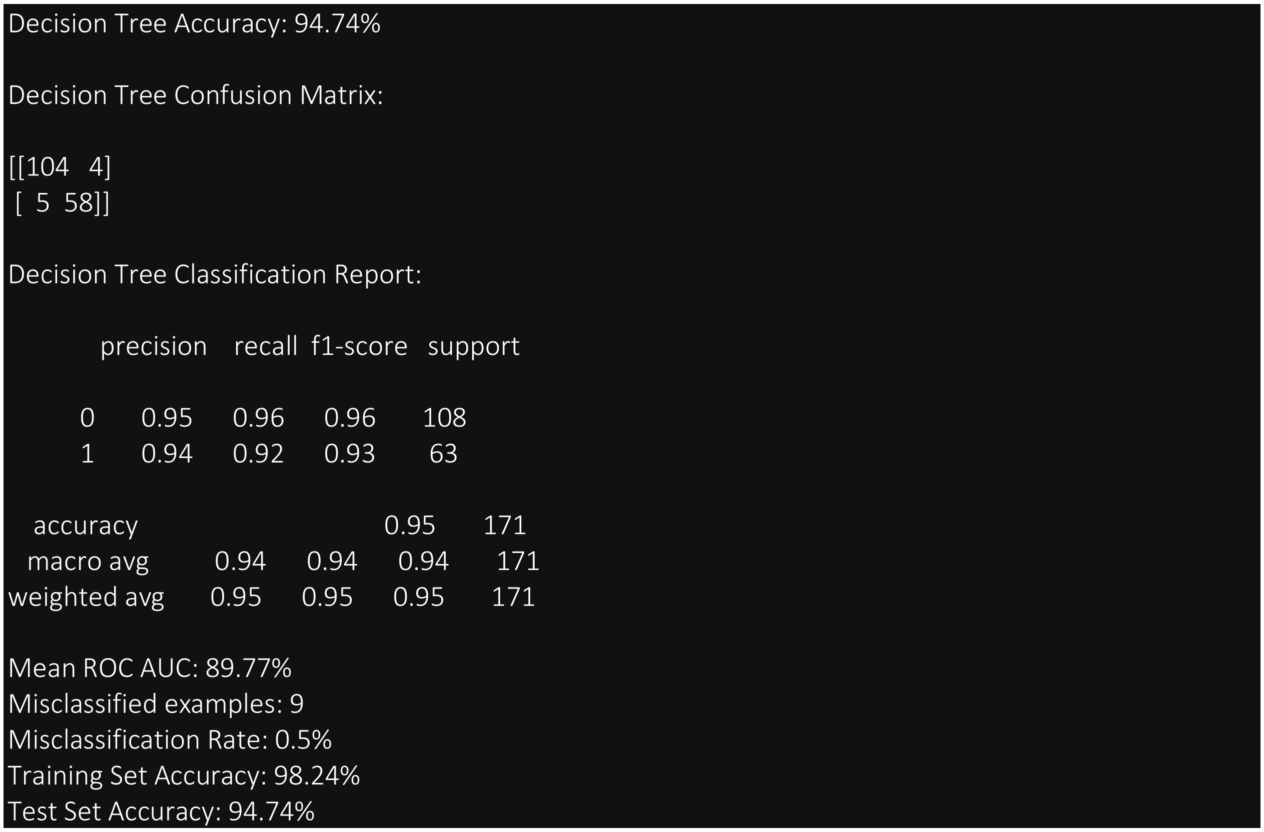

Bootstrap aggregating or Bagging is a ensemble meta-algorithm combining predictions from multiple-decision trees through a majority voting mechanism
Models are built sequentially by minimizing the errors from previous models while increasing (or boosting) influence of high-performing models
Optimized Gradient Boosting algorithm through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias
Bagging
Boosting
XGBoost
Decision Trees
Random Forest
Gradient Boosting
A graphical representation of possible solutions to a decision based on certain conditions
Bagging-based algorithm where only a subset of features are selected at random to build a forest or collection of decision trees
Gradient Boosting employs gradient descent algorithm to minimize errors in sequential models

Decision Tree: The Roots

The way we decided to go about looking at Random Forest modeling was to compare it to its simpler cousin, the Decision Tree Classifier. In a nutshell, Random Forest is fairly similar to Decision Trees in that it is actually a "forest' of Decision Trees that are placed into the Random Forrest with the idea that many more trees should be more accurate on average due to the nature of being clumped into the same model. As such, we wanted to compare Decision Trees to the Random Forest Model. We did so in multiple ways: We took the base line PCA reduced Decision Tree, then a pruned Decision Tree, compared those to the a base model Random Forest, and finally a Random Forest that has had its parameters tuned to provide the best possible result.

Decision Tree on Breast Cancer Dataset

Looking at the reasonably balanced dataset, the Breast Cancer data, the confusion matrix and accuracy are already very strong, with a Misclassification rate of just 6%, and the validation accuracy sitting at 93.5%. Given that the training set accuracy is 100%, there is a good chance this model is overfitting. Very strong start but we believe that this is due to the balanced nature of the dataset. Since there is a fairly even split of results, the splitting that transpire within the Decision Tree model is able to sift the results and there is a reasonable chance that the classifier will place it right after learning from the training data. This is only more noticeable with the pruned tree. By using the tree with the least amount of impurity, we measure this by look at the alpha rate of the decision trees that yield the highest accuracy while also avoiding over-fitting, we can improve the validation accuracy. And we do with our alpha that matches the above criteria, with the corresponding tree yielding the same misclassification rate of 5% but the validation accuracy jumps up higher to 94.4%. Interestingly, our training accuracy falls which by our estimation would mean that the model is no longer over-fitting, but the training accuracy remains at 98.2%, which is more than enough to prove that our model is working.



```
Decision Tree Accuracy: 94.74%

Decision Tree Confusion Matrix:

[[104   4]
 [  5  58]]

Decision Tree Classification Report:

              precision    recall  f1-score   support

           0       0.95      0.96      0.96       108
           1       0.94      0.92      0.93        63

    accuracy                           0.95       171
   macro avg       0.94      0.94      0.94       171
weighted avg       0.95      0.95      0.95       171

Mean ROC AUC: 89.77%
Misclassified examples: 9
Misclassification Rate: 0.5%
Training Set Accuracy: 98.24%
Test Set Accuracy: 94.74%
```

Decision Tree on Wheat Dataset

With the Wheat data, this dataset was smaller, with only 209 rows of data. But the interesting thing with this dataset was that the results were split into 3 possible results. We were curious to see how our models would handle these.

```
Decision Tree Accuracy: 88.89%

Decision Tree Confusion Matrix:

[[17  2  5]
 [ 0 18  0]
 [ 0  0 21]]

Decision Tree Classification Report:

            precision    recall  f1-score   support

         0       1.00      0.71      0.83        24
         1       0.90      1.00      0.95        18
         2       0.81      1.00      0.89        21

   micro avg       0.89      0.89      0.89        63
   macro avg       0.90      0.90      0.89        63
weighted avg       0.91      0.89      0.88        63
 samples avg       0.89      0.89      0.89        63

Mean ROC AUC: 100.00%
Misclassified examples: 14
Misclassification Rate: 0.22%
Training Set Accuracy: 91.16%
Test Set Accuracy: 88.89%
```

Decision Tree on Credit Card Dataset

Next, we looked at a skewed dataset to judge the performance of the Decision Tree classifier with a non-balanced dataset. This dataset was very difficult to work with, as the dataset was massive in comparison to the much more manageable breast cancer dataset, and it must be noted that the question of hardware selection came into question here. Moving forward, the base Decision Tree suffers with the skewed data, sporting a 30% misclassification rate, 5 times the rate in our breast cancer dataset, and a validation accuracy of 70.38%. Using the pruned tree gives a nice boost in performance, dropping the misclassification rate to 20% and raising the validation accuracy to 80.03%. We'd like to point out that our f-1 scores were the primary tool we were using to measure the credit card data set as the f1 score looks more closely at both False Negatives and Positives. In this way, we see that our f1-score for the positive scenario was at a dismal 37%, meaning that it was not correctly predicting defaults and would not be suitable for this kind of data analysis at all.

```
Decision Tree Accuracy: 80.03%

Decision Tree Confusion Matrix:

[[6667  333]
 [1464  536]]

Decision Tree Classification Report:

            precision    recall  f1-score   support
```

```
       0      0.82      0.95      0.88      7000
       1      0.62      0.27      0.37      2000

   accuracy                           0.80      9000
  macro avg       0.72      0.61      0.63      9000
weighted avg      0.77      0.80      0.77      9000

Mean ROC AUC: 65.88%
Misclassified examples: 1797
Misclassification Rate: 20.0%
Training Set Accuracy: 80.27%
Test Set Accuracy: 80.03%
```

---| Bag & Booster |---

Random Forest on Breast Cancer Dataset

Now when look at the Random Forest Model we get a misclassification rate of 6% and a validation accuracy of 93.57%, identical to our unpruned Decision Tree. Plugging in our best parameters yields the same validation accuracy, but drops the misclassification rate to 5%. From this we can see that running Random Forest is not necessarily a good thing to do with this kind of dataset.

```
Random Forest Accuracy: 94.74%

Random Forest Confusion Matrix:

[[104   4]
 [  5  58]]

Random Forest Classification Report:

           precision    recall  f1-score   support

        0      0.95      0.96      0.96       108
        1      0.94      0.92      0.93        63

   accuracy                           0.95       171
  macro avg       0.94      0.94      0.94       171
weighted avg      0.95      0.95      0.95       171

Mean ROC AUC: 91.99%
Misclassified examples: 9
Misclassification Rate: 0.05%
Training Set Accuracy: 100.00%
Test Set Accuracy: 93.57%
```

Given that the Training Set accuracy is 100% in Random Forest, there’s a good chance this model is overfitting.

Tree-Based Bagging Method: Random Forest

Random Forest on Credit Card Dataset

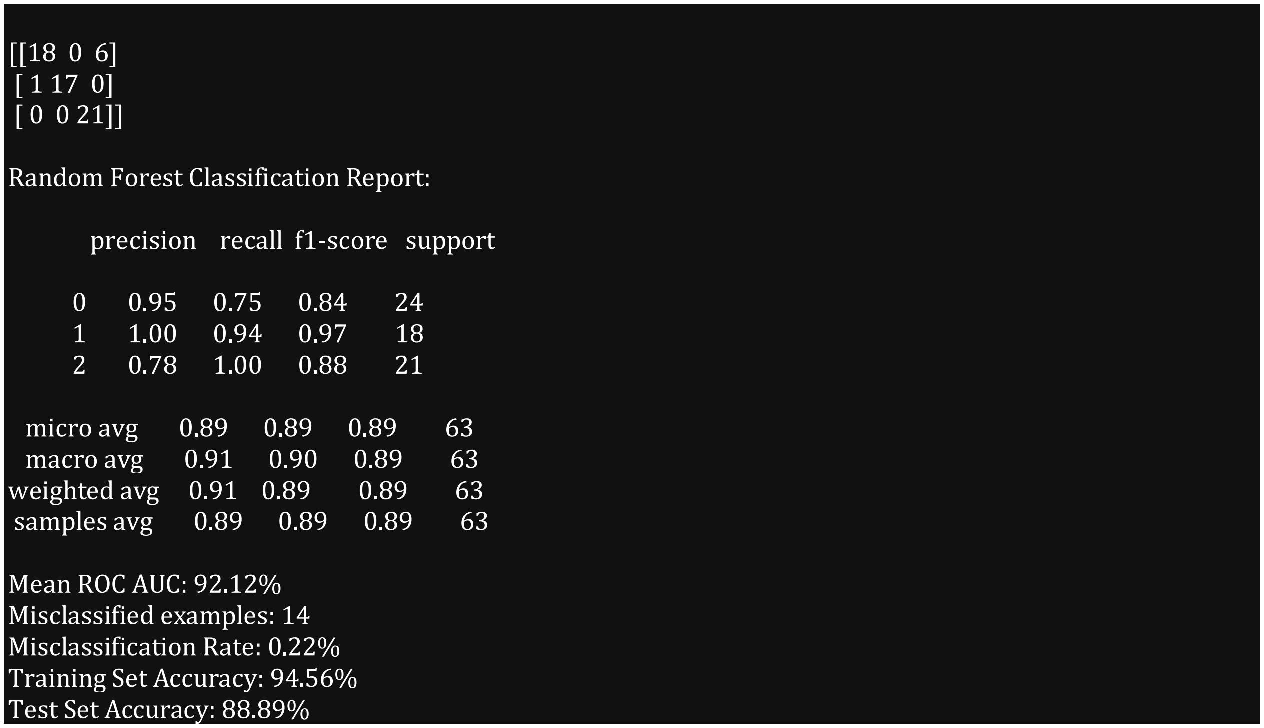
Random Forest results were marginally better than the base Decision Tree, with a misclassification rate of 28% and validation accuracy of 71.8%. Unfortunately, even with the best parameters chosen, the Random Forest model fairs worse than the pruned Decision Tree, with a 25% misclassification rate and a 74.88% validation accuracy. This does make sense however, as the Random Forest is looking to find the average of a forest of decision trees so even if the Forest does have a stronger tree than the pruned Decision Tree, that likely will not matter as the other trees will average it out. Similar to the Decision, we see similar results with the positive f1-score, at 36% instead of 37 but this makes sense as well, seeing as we are taking the average of the Decision Trees.

```
Random Forest Accuracy: 74.88%

Random Forest Confusion Matrix:

[[6112  888]
 [1373  627]]

Random Forest Classification Report:

          precision    recall  f1-score   support

       0       0.82      0.87      0.84      7000
       1       0.41      0.31      0.36      2000

    accuracy                           0.75      9000
   macro avg       0.62      0.59      0.60      9000
weighted avg       0.73      0.75      0.74      9000

Mean ROC AUC: 65.40%
Misclassified examples: 2261
Misclassification Rate: 0.25%
Training Set Accuracy: 88.26%
Test Set Accuracy: 74.88%
```

Random Forest on Wheat Dataset

With the multiclass classification on wheat dataset, we found that inputting the optimal parameters actually lowered our accuracy results; however, this likely means reduced overfitting because the initial training accuracy 100%, 8% higher than the baseline test accuracy. After finding optimal parameters, we yielded the following:

```
Random Forest Accuracy: 88.89%

Random Forest Confusion Matrix:
```



```
[[18  0  6]
 [ 1 17  0]
 [ 0  0 21]]

Random Forest Classification Report:

          precision    recall  f1-score   support

        0      0.95      0.75      0.84        24
        1      1.00      0.94      0.97        18
        2      0.78      1.00      0.88        21

  micro avg      0.89      0.89      0.89        63
  macro avg      0.91      0.90      0.89        63
weighted avg     0.91    0.89        0.89        63
 samples avg      0.89      0.89      0.89        63

Mean ROC AUC: 92.12%
Misclassified examples: 14
Misclassification Rate: 0.22%
Training Set Accuracy: 94.56%
Test Set Accuracy: 88.89%
```

Looking at the Random Forest model, we see identical results to the decision tree: a misclassification rate of 16% and a validation accuracy of 16% in our base model, and a misclassification rate of 22% coupled with a validation accuracy of 88.89% in the Random Forest model with best parameters. What we can say then, is that Random Forest does not handle the multi-class problem any better than the Decision Tree model, which in and of itself is interesting; if the Random Forest is looking at the average of the Decision Trees, then it is odd that it would share the exact same results as the tree. We are not quite sure why this is case and would require further research.

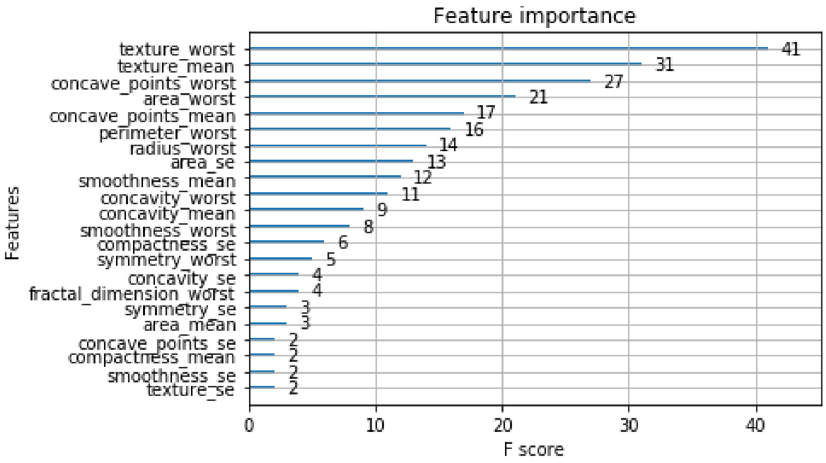
The base Decision Tree had a Misclassification rate of 16%, coupled with a 92.06% training accuracy. Thus, while the model had some trouble with the multi-class nature of the model, it was still extremely accurate. Pruning the tree resulted in a rise in misclassification rate, to 22%, and the validation accuracy also suffered, falling to 88.89%.

XG Boost Classification (Extreme Gradient Boosting)

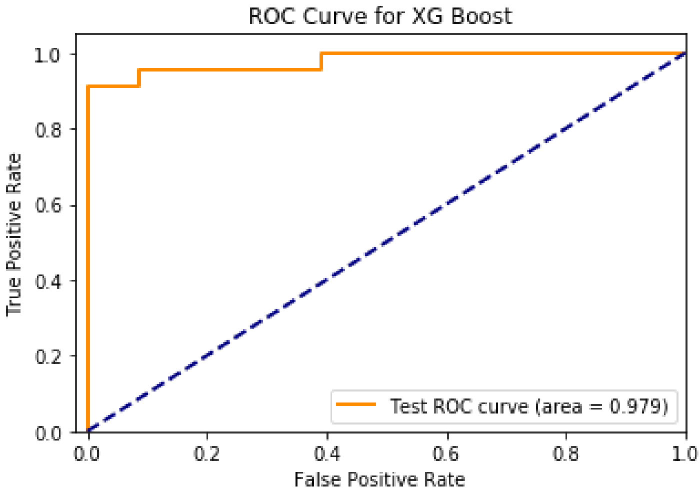
Extreme Gradient Boost (XGBoost Classifcication) is a custom tree building algorithm that can be used for classification, regression, and ranking. Using gradient-boosted decision trees, it is designed for speed and performance. The name refers to the engineering goal to push the limit of computations resources for boosted tree algorithms, by taking a bunch of weak learners (trees) are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. XGBoost makes splits up to the max_depth specified and then start pruning the tree backwards and remove splits beyond which there is no positive gain.

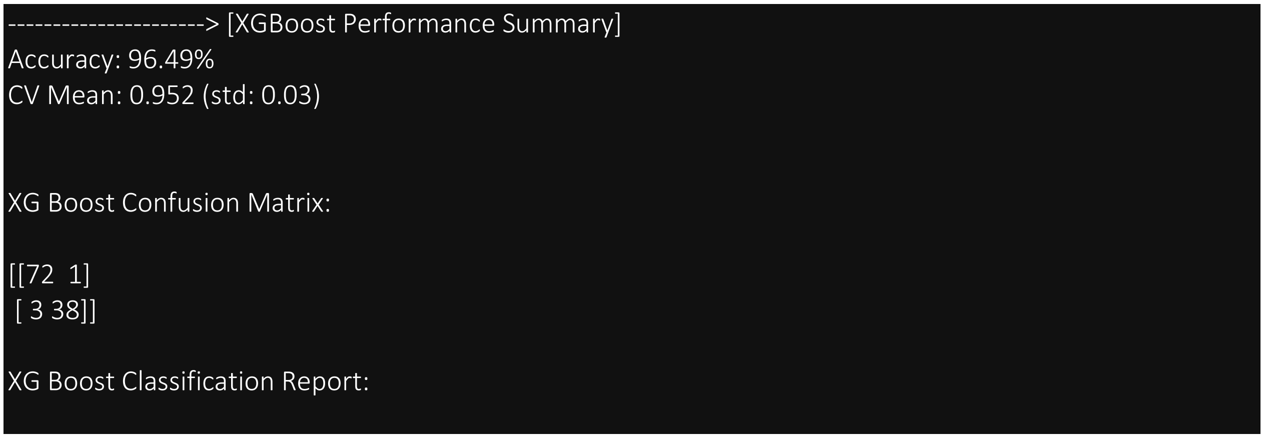
XGBoost on Breast Cancer Dataset

Using XGBoost’s feature importance, we determine the following rankings of features based on f-score:

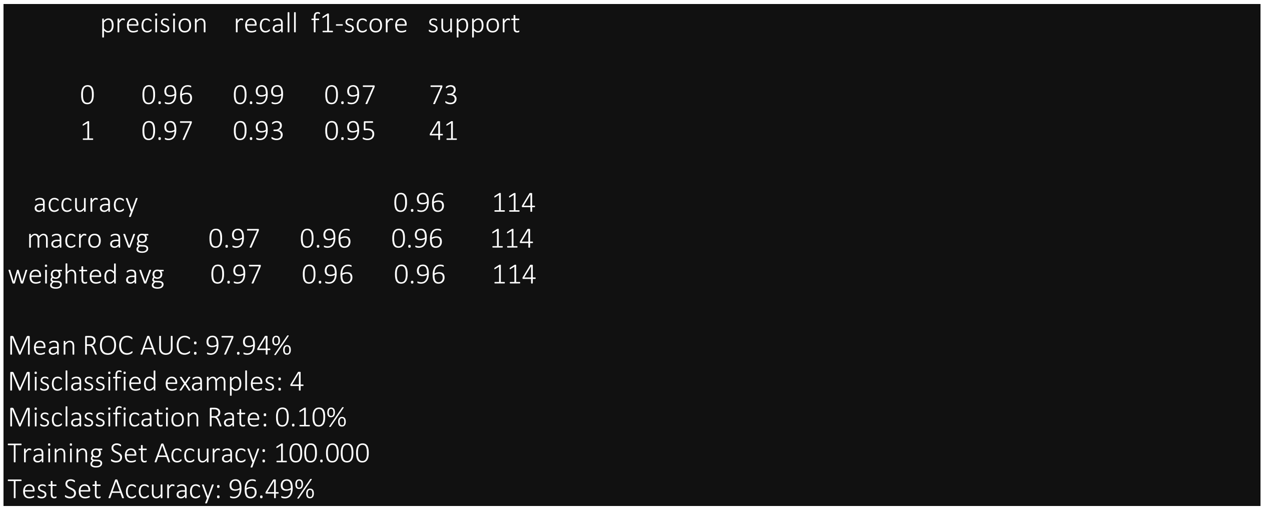


Plotting True Positive vs False Positive rates on Breast cancer via XGBoost:





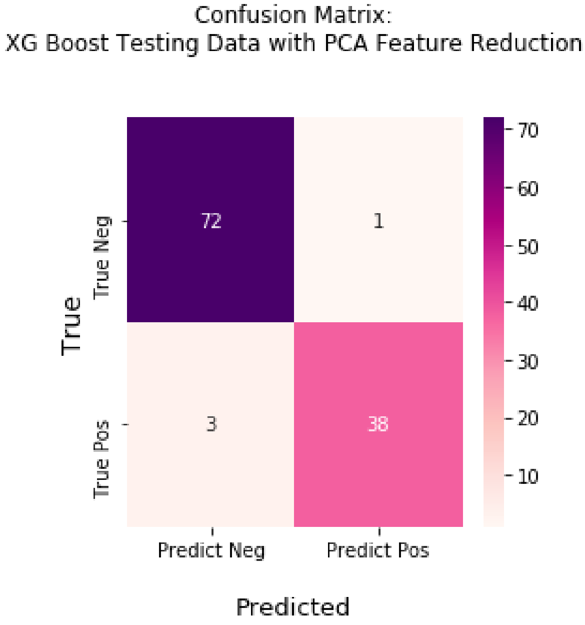
```
---------------------> [XGBoost Performance Summary]
Accuracy: 96.49%
CV Mean: 0.952 (std: 0.03)


XG Boost Confusion Matrix:

[[72  1]
 [ 3 38]]

XG Boost Classification Report:
```



```
          precision    recall  f1-score   support

        0      0.96      0.99      0.97        73
        1      0.97      0.93      0.95        41

   accuracy                        0.96       114
  macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114

Mean ROC AUC: 97.94%
Misclassified examples: 4
Misclassification Rate: 0.10%
Training Set Accuracy: 100.000
Test Set Accuracy: 96.49%
```

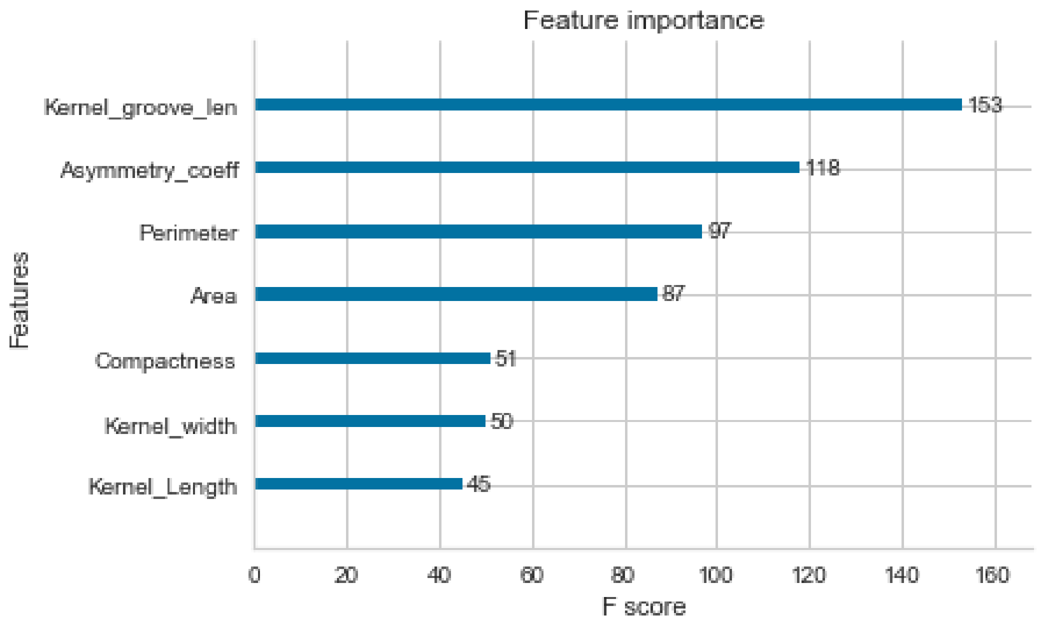
XGBoost has a reasonably high accuracy, and although the Breast Cancer dataset is unbalanced, both the precision and recall are good. There were 3 false negatives and 1 false positive.

Overall CV accuracy:



XGBoost on Wheat Dataset

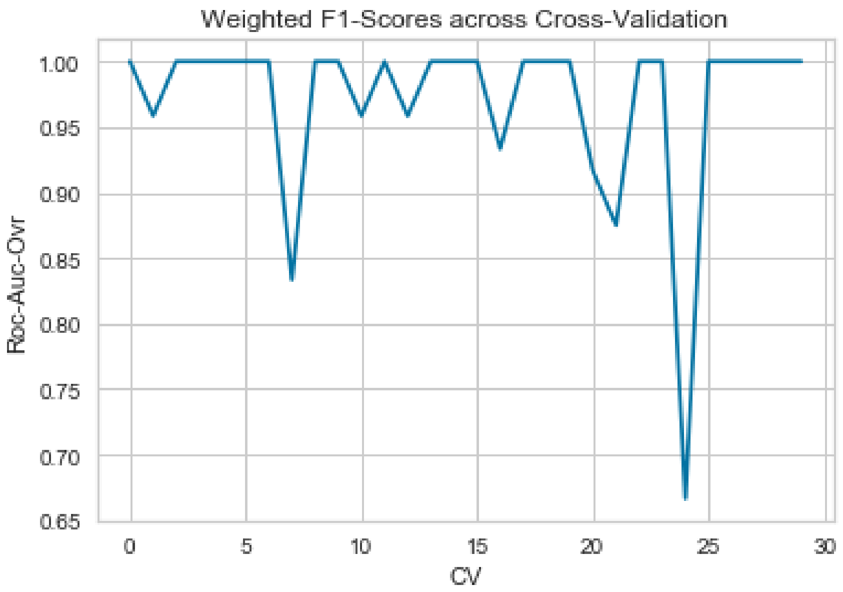
After running XGB's feature importance on the Wheat dataset, we can see the following are most important features:



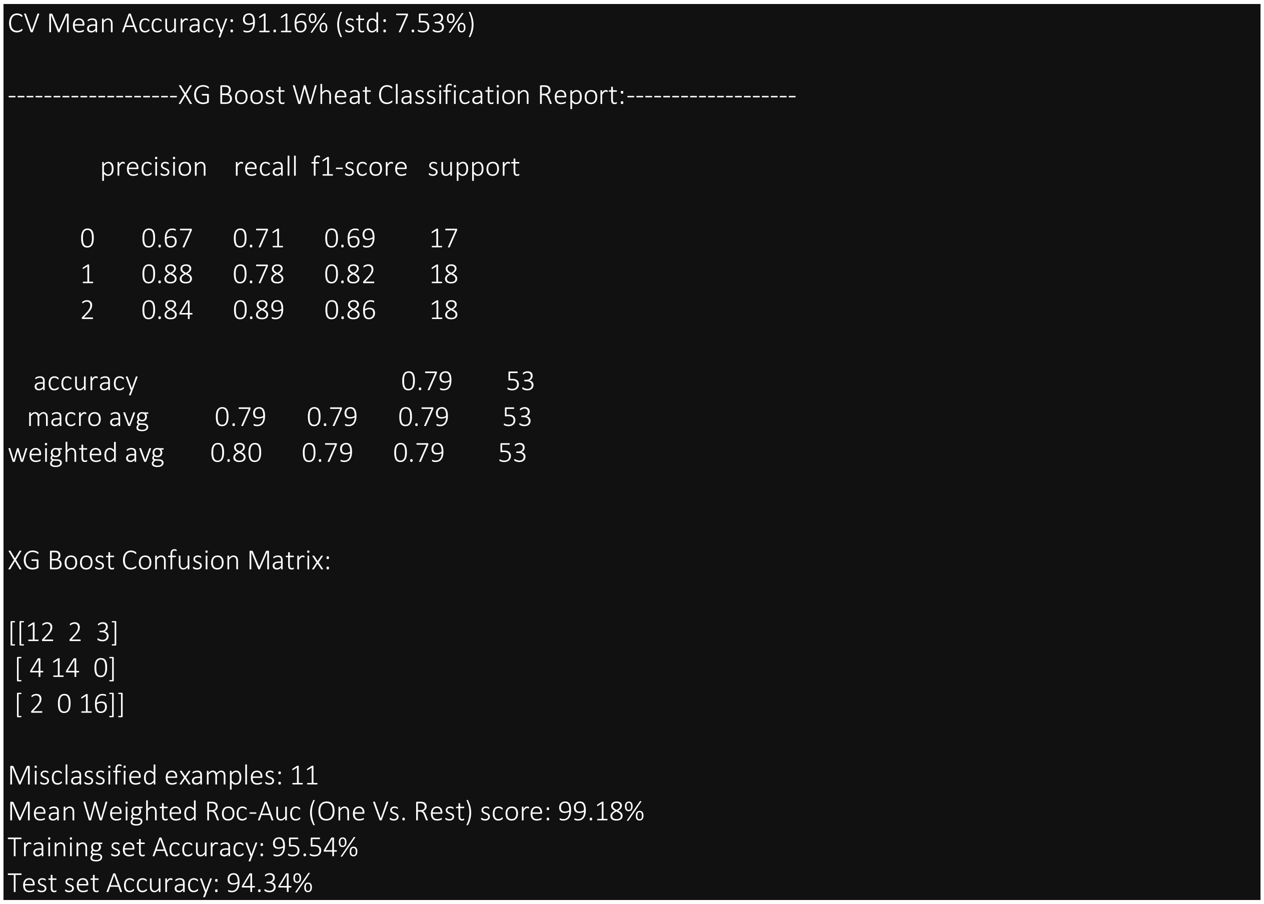
Kernel groove length and asymmetry coefficient are the most important features.

We found the optimal parameter settings by running the dataset through the scale, PCA, fit, and parameter selection pipeline. This yielded the following results:

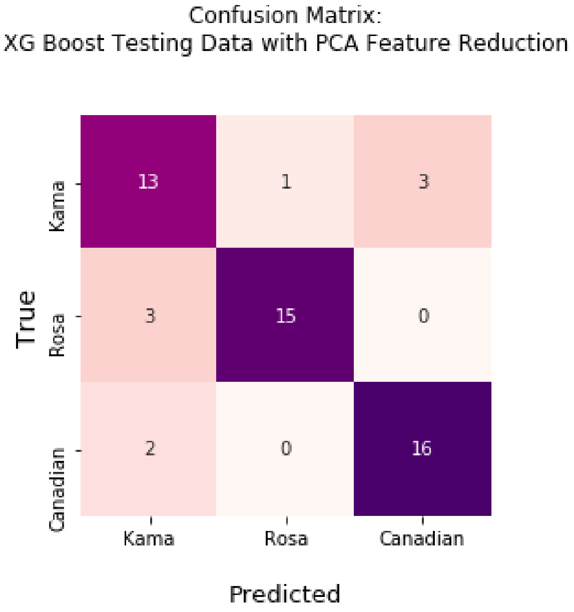
Because this is a multiclass classification problem, we used Roc-Auc-Ovr (One vs Rest) as a measuring metric. F1 is the weighted average of precision and recall.

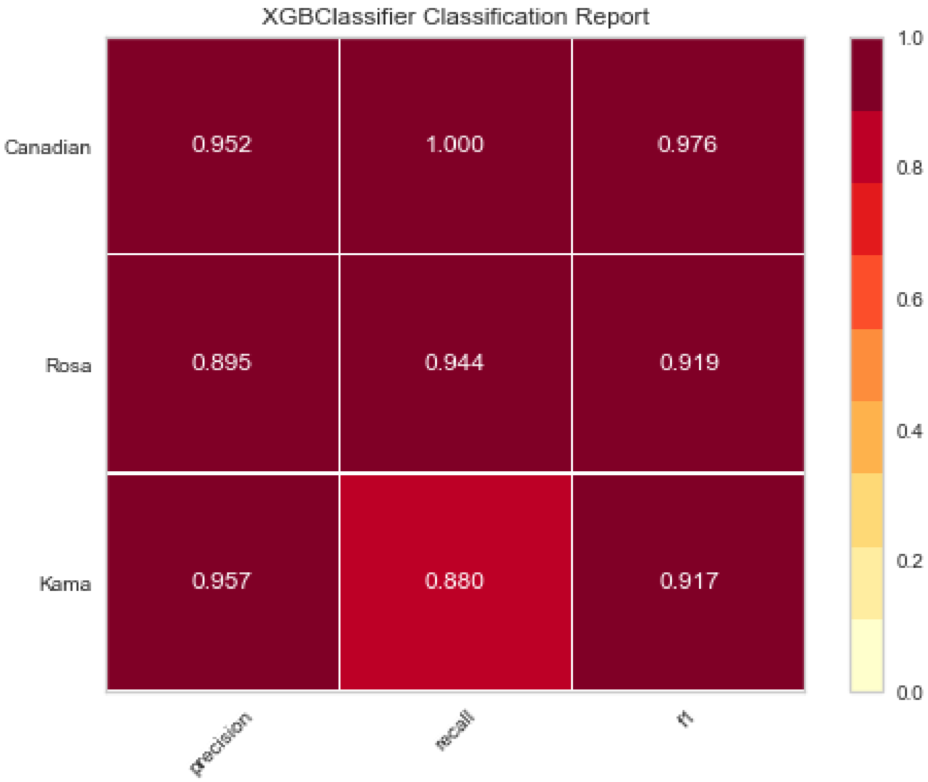


```
CV Mean Accuracy: 91.16% (std: 7.53%)
```

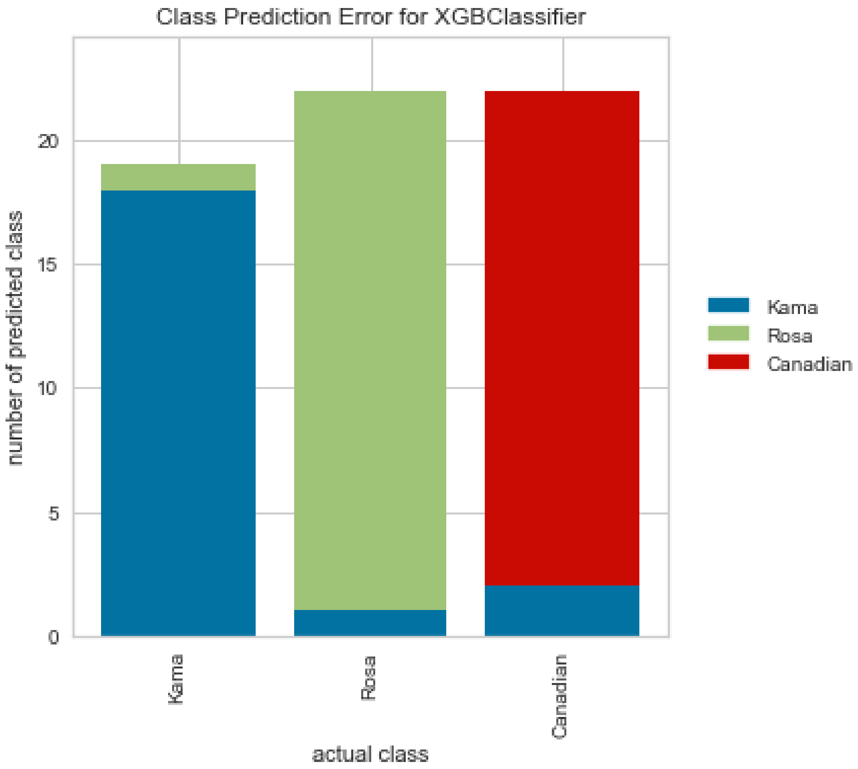


```
CV Mean Accuracy: 91.16% (std: 7.53%)

-------------------XG Boost Wheat Classification Report:-------------------

             precision    recall  f1-score   support

           0       0.67      0.71      0.69        17
           1       0.88      0.78      0.82        18
           2       0.84      0.89      0.86        18

    accuracy                           0.79        53
   macro avg       0.79      0.79      0.79        53
weighted avg       0.80      0.79      0.79        53


XG Boost Confusion Matrix:

[[12  2  3]
 [ 4 14  0]
 [ 2  0 16]]

Misclassified examples: 11
Mean Weighted Roc-Auc (One Vs. Rest) score: 99.18%
Training set Accuracy: 95.54%
Test set Accuracy: 94.34%
```





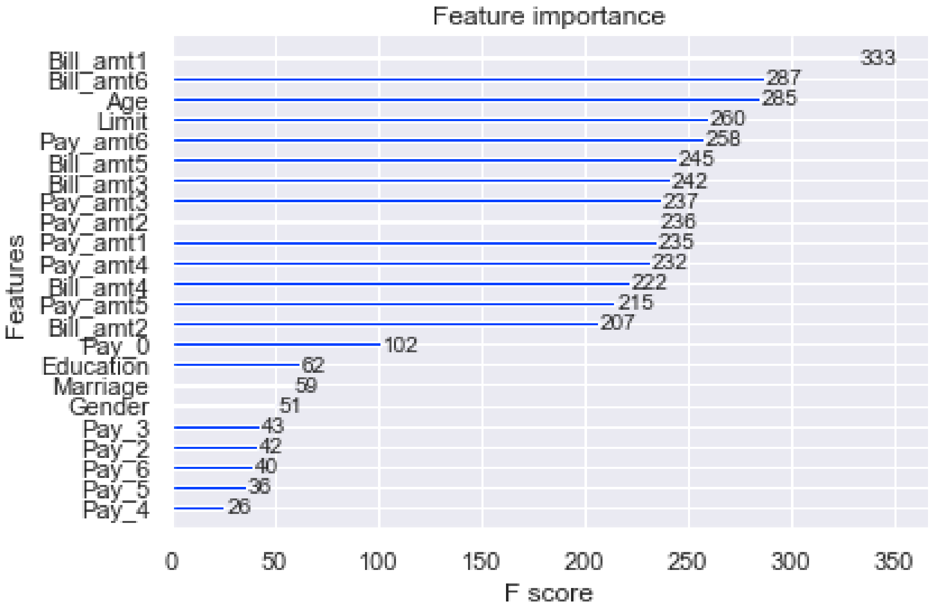
In terms of F1 score, XGB had the best performance on Canadian kernel, then Rosa, then Kama. It had the most difficulty in recall for Kama.



Looking at the proportional prediction error, XGB had more errors predicting Canadian.

XGBoost on Credit Card Dataset

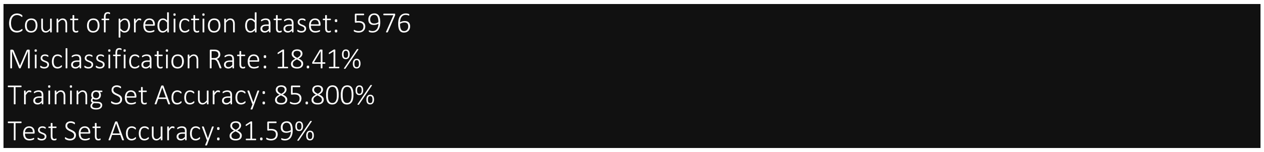
We can see that Bill amount 1, Bill Amount 6, and client's age are the most important features, per XGBoost:



After using pipeline to run the dataset through scaling, PCA, and fitting the model, we ran GridSearchCV to select the best parameter values.

Using those values, XGBoost obtained the following results on the credit card dataset:

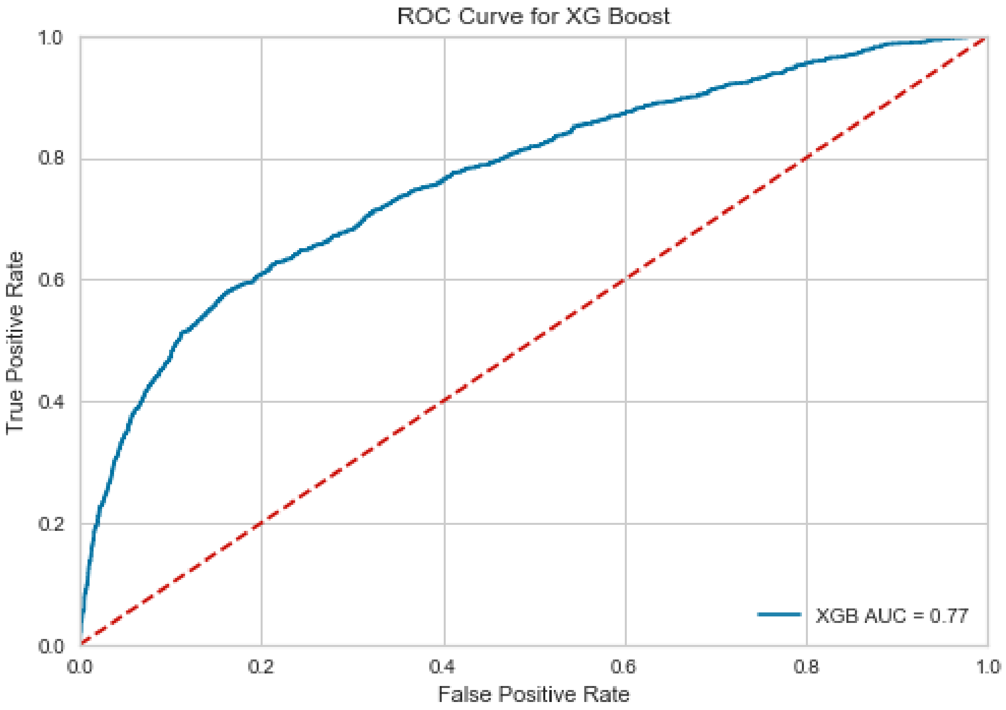
```
------------------------ | XGBoost Performance Summary | --------------------------
Accuracy: 81.59%
CV Mean: 0.815 (std: 0.01)


XG Boost Confusion Matrix:

[[4417  233]
 [ 867  459]]

XG Boost Classification Report:

           precision    recall  f1-score   support

        0       0.84      0.95      0.89      4650
        1       0.66      0.35      0.45      1326

   accuracy                           0.82      5976
  macro avg       0.75      0.65      0.67      5976
weighted avg       0.80      0.82      0.79      5976

Mean ROC AUC: 75.30%
Misclassified examples: 1100
```

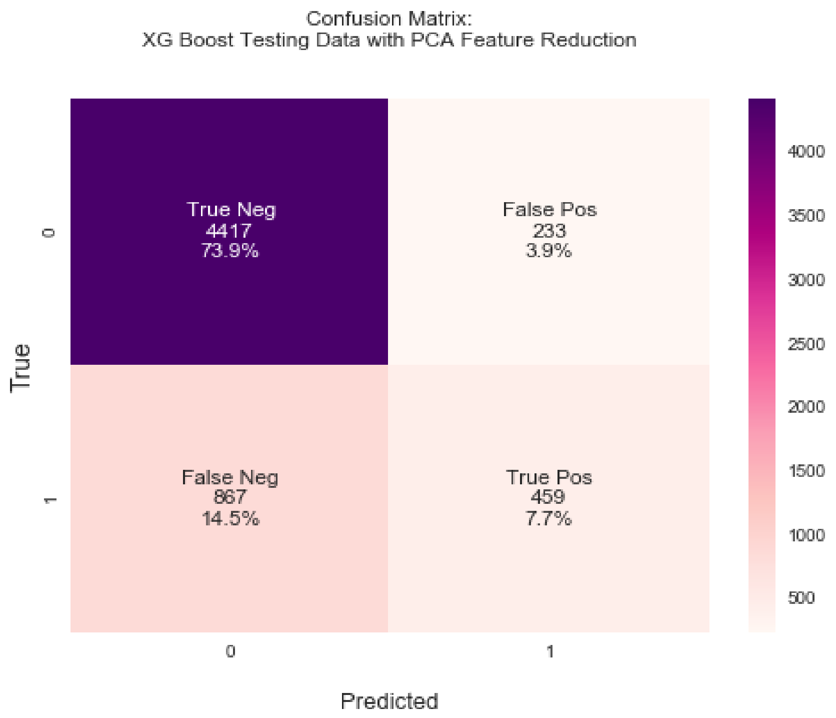


```
Count of prediction dataset:  5976
Misclassification Rate: 18.41%
Training Set Accuracy: 85.800%
Test Set Accuracy: 81.59%
```

From the output, we can see that XGB had poor recall in the positive class, at 0.35; recall is the ability of the model to find all the relevant cases within the dataset. F1 score here is better than either Decision Tree or Random Forest and so it would be more suitable than those two but still not stellar as it scored on a 45% f-1 score.

We can see from the ROC curve plotting true positive rate vs false positive rate that the area under the curve was 77%:





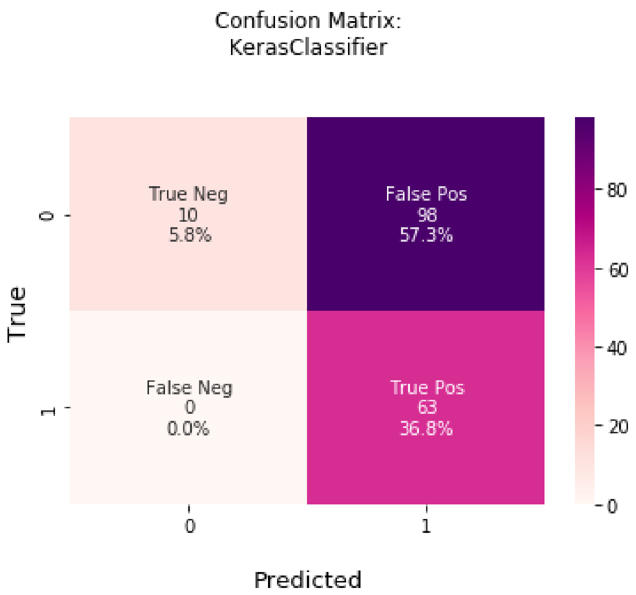
From the confusion matrix, we can see that XGB had a more difficult time with false negatives than it did with false positives. This makes sense, given that this is an unbalanced dataset with far more examples in the negative (0, no default) class than the positive (1, default).

---| Deep Learning |---

Deep Learning on Breast Cancer Dataset

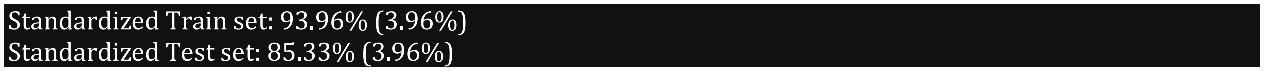
First, we obtained a baseline model, with the plan to sequentially add layers (both depth and width) until the desired network topology was achieved.

```
Baseline Train set: 77.37% (12.94%)
Baseline Test set: 64.93% (18.73%)
```



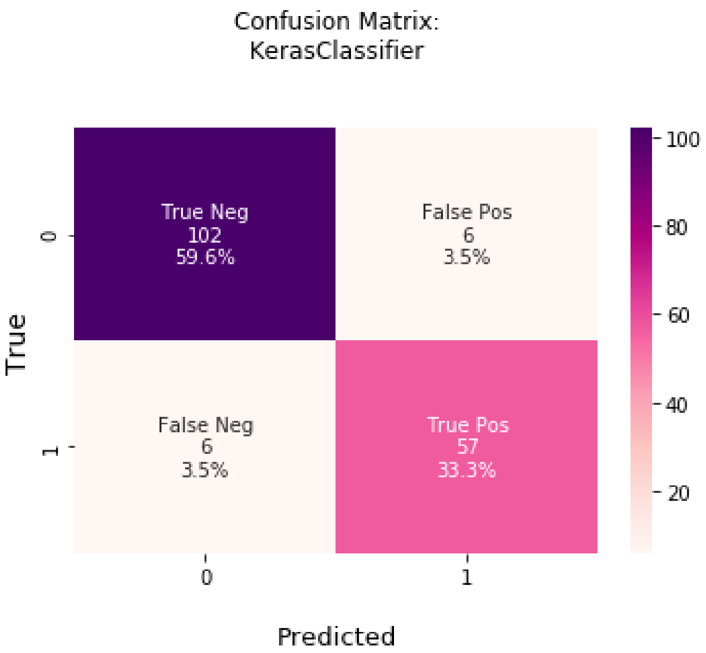
This base model struggled significantly in predicting positive when it was actually negative, clocking in at 57% false negatives.

After standardization:



```
Standardized Train set: 93.96% (3.96%)
Standardized Test set: 85.33% (3.96%)
```

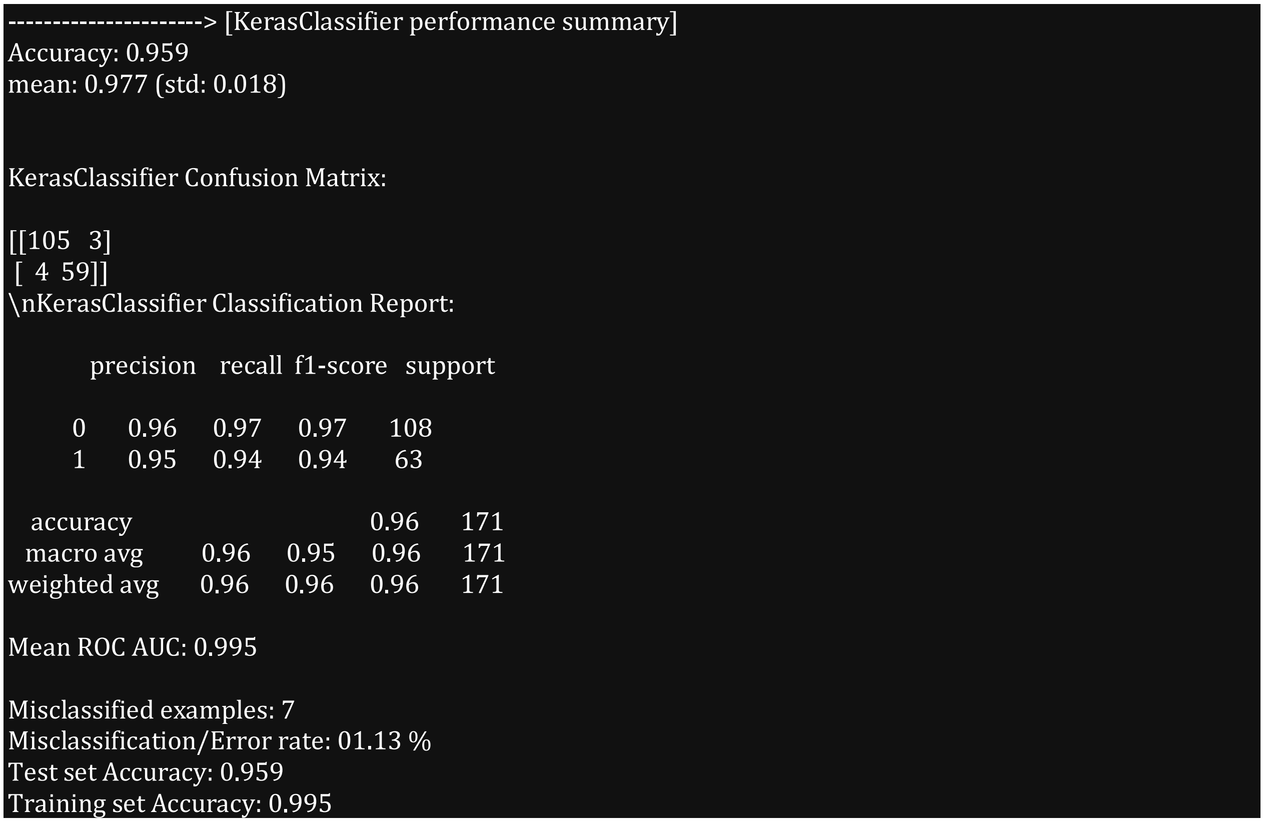
Performance greatly improved.



As this confusion matrix shows, the model had more false negatives (3.5%) than previous, but had only 3.5% false positives, as opposed to 57% from the baseline model

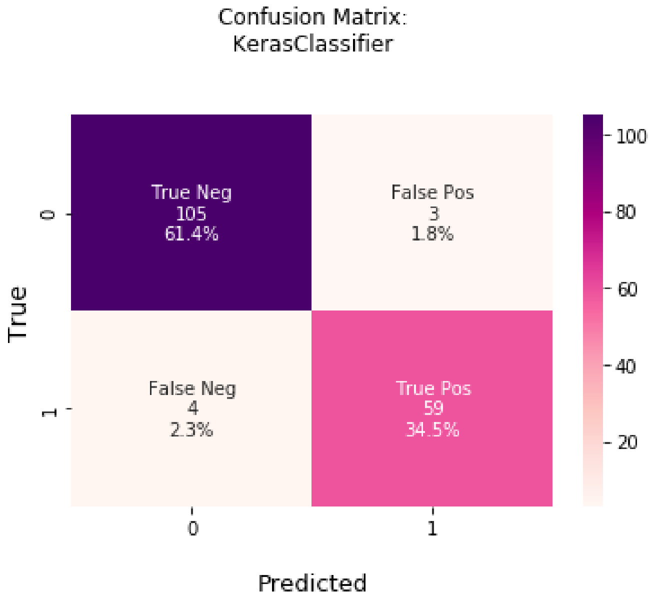
For our final NN model, the first hidden layer has 34 neurons, using uniform initialization method, and relu activation function. The second layer has 30 neurons. Finally, the output layer has 1 neuron and uses sigmoid activation function, to ensure that the network output is between 0 and 1. Thus, the final layer predicts class, malignant or benign.

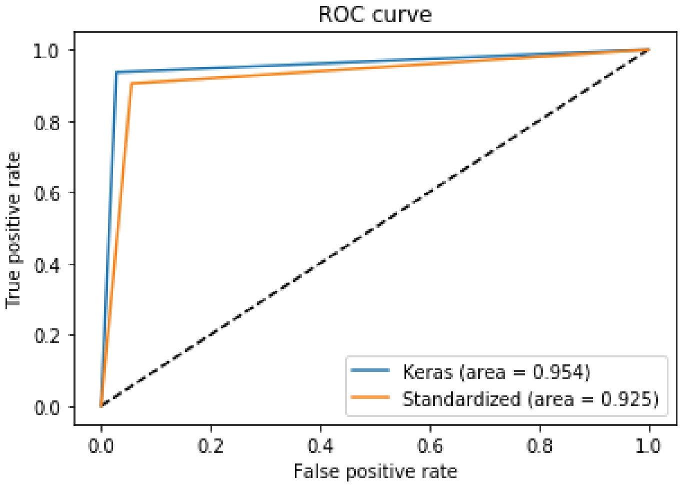
Then, after running the model through a pipeline to scale, fit, and search for the best parameters, our model yielded the following better-still results:



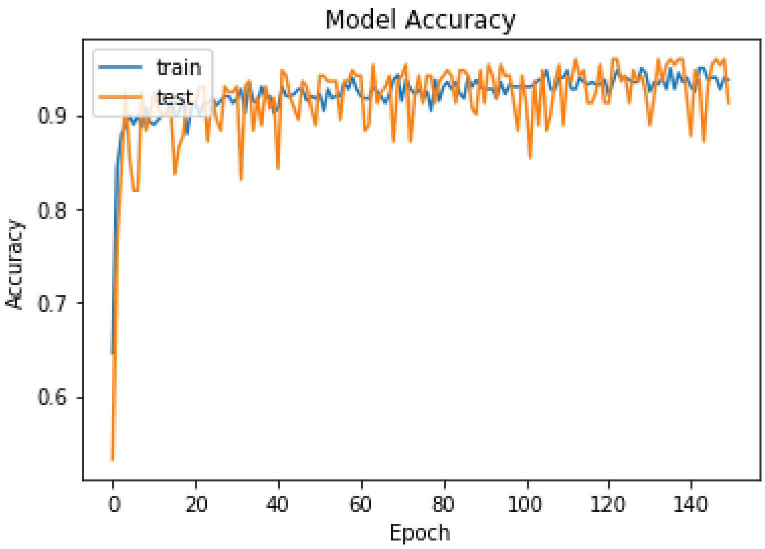
```
---------------------> [KerasClassifier performance summary]
Accuracy: 0.959
mean: 0.977 (std: 0.018)


KerasClassifier Confusion Matrix:

[[105   3]
 [  4  59]]
\nKerasClassifier Classification Report:

            precision    recall  f1-score   support

         0       0.96      0.97      0.97       108
         1       0.95      0.94      0.94        63

   accuracy                           0.96       171
  macro avg       0.96      0.95      0.96       171
weighted avg      0.96      0.96      0.96       171

Mean ROC AUC: 0.995

Misclassified examples: 7
Misclassification/Error rate: 01.13 %
Test set Accuracy: 0.959
Training set Accuracy: 0.995
```

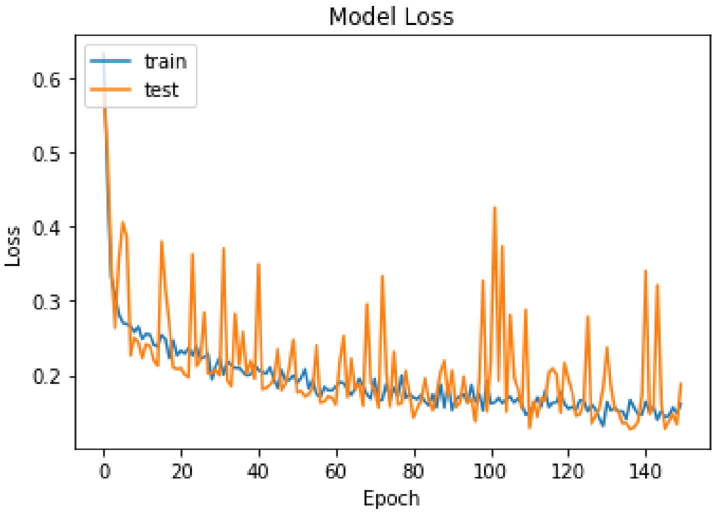
Only 1.8% false positive and 2.3% false negative rate.





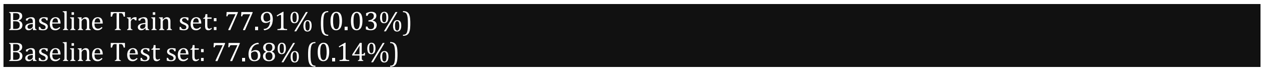
Using learning curves to plot model accuracy and loss vs epochs, we can see that the test set had a non-smooth learning rate:





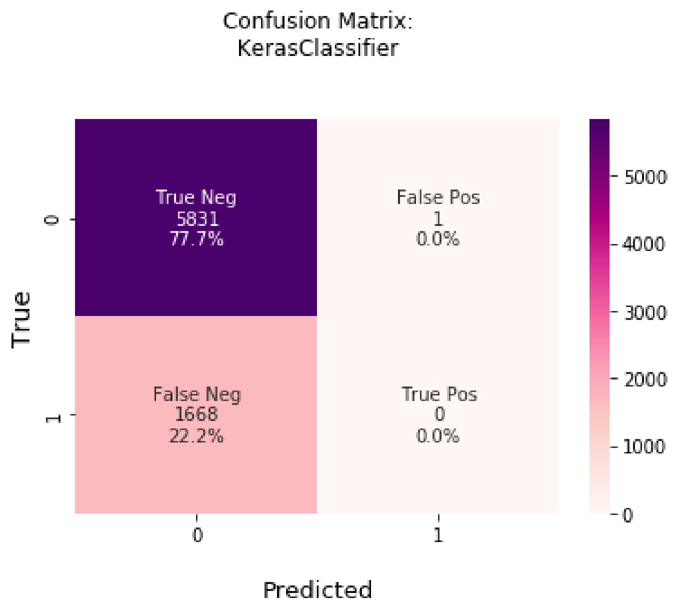
Deep Learning on Credit Card Dataset:

After feature selection to reduce the number of features and prevent overfitting, our baseline NN model showed the following:

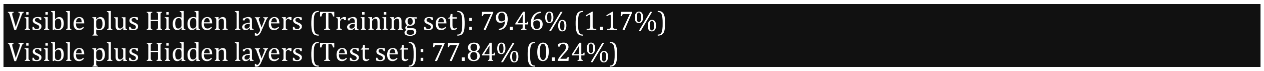


```
Baseline Train set: 77.91% (0.03%)
Baseline Test set: 77.68% (0.14%)
```

This baseline model yielded a very high proportion of false *positives* this time, at 22%:

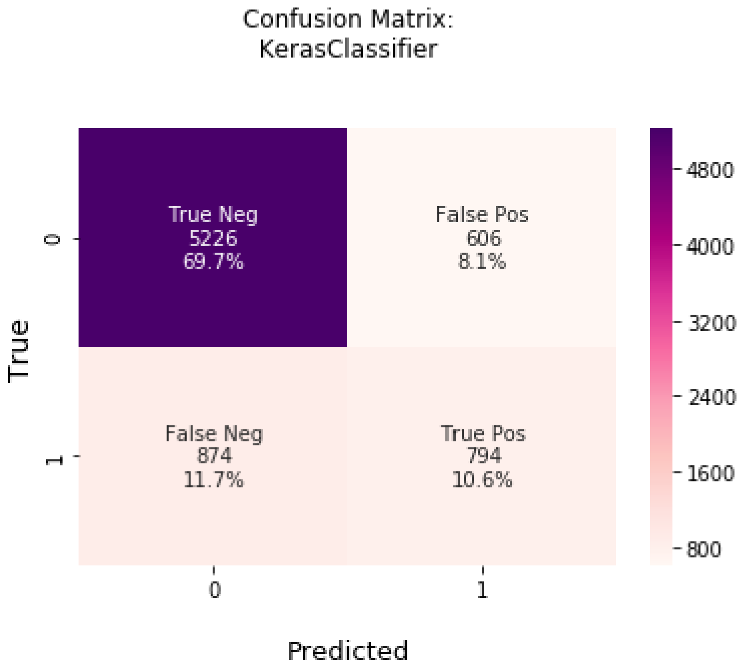


Then, using dropout regularization yielded the following:



```
Visible plus Hidden layers (Training set): 79.46% (1.17%)
Visible plus Hidden layers (Test set): 77.84% (0.24%)
```

In doing so, we yielded more false positives but significantly less false negatives:

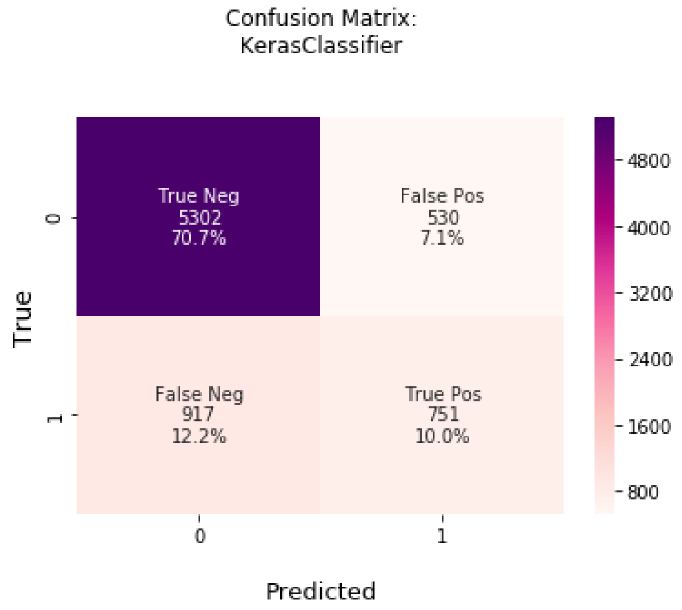


Using dropout regularization on the full dataset:

```
Visible plus Hidden layers (Training set): 80.78% (1.17%)
Visible plus Hidden layers (Test set): 78.64% (0.24%)
```

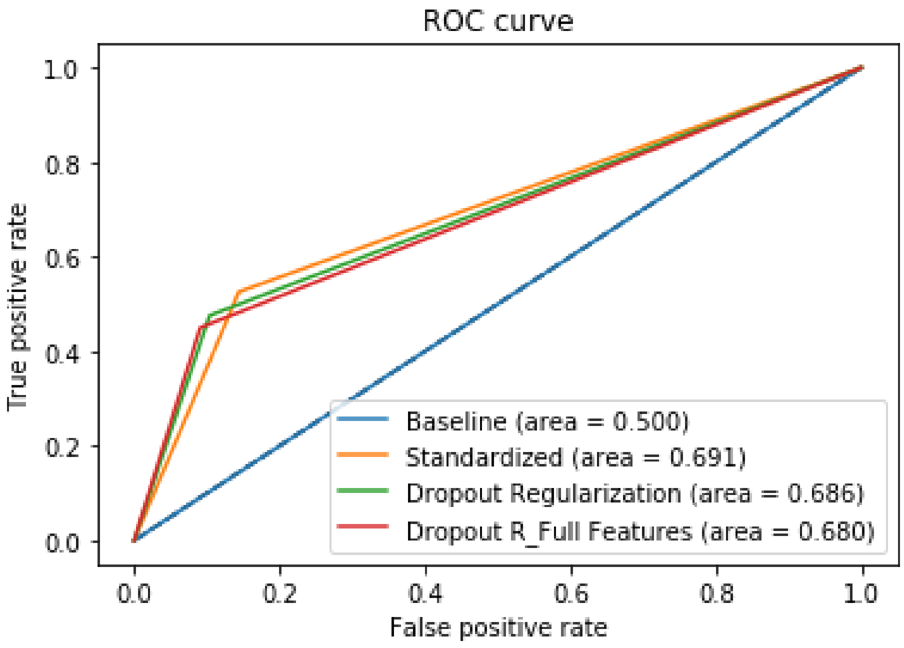
```
---------------------> [KerasClassifier performance summary]
Accuracy: 0.807

KerasClassifier Confusion Matrix:

[[5302  530]
 [ 917  751]]
\nKerasClassifier Classification Report:

              precision    recall  f1-score   support

           0       0.85      0.91      0.88      5832
           1       0.59      0.45      0.51      1668

    accuracy                           0.81      7500
   macro avg       0.72      0.68      0.69      7500
weighted avg       0.79      0.81      0.80      7500

Mean ROC AUC: 0.708

Misclassified examples: 1447
Misclassification rate: 1.130 %
Test set Accuracy: 0.807
Training set Accuracy: 0.808
```

F1 Score is the best in this case, but again it is really low, in this case it is at 51%. Barely above half correctly classified.



Surprisingly, dropout regularization works somewhat better with more features.

We can see these models compared on the ROC curve:



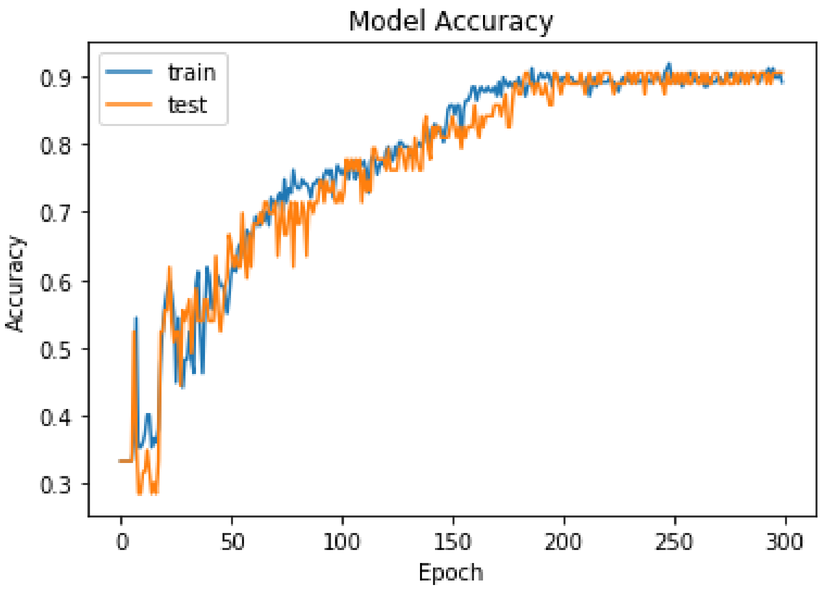
Thus, dropout regularization seems to have the best area under the curve score.

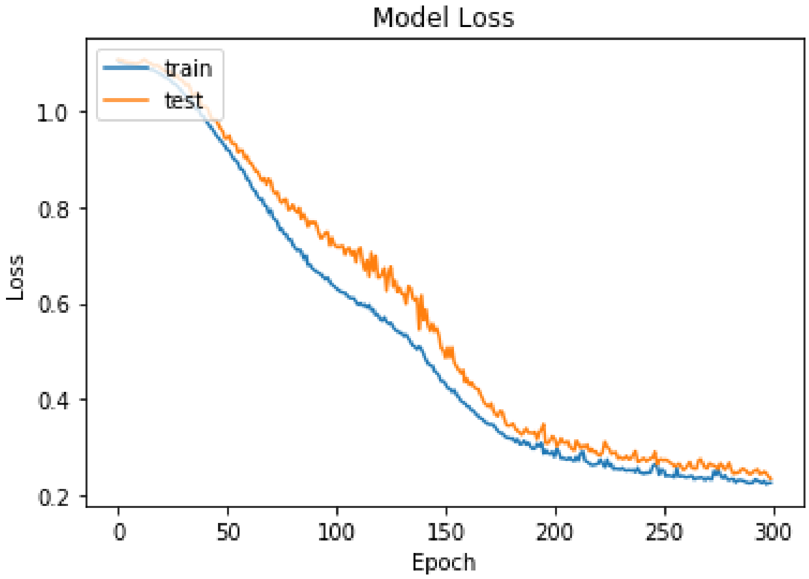
Deep Learning on Wheat Dataset:

Our baseline model had the following results on the multiclass, balanced dataset:

```
(Training set): 88.34% (6.46%)
(Test set): 56.67% (31.32%)
```

On the learning curve, we can see that there is some noisiness in the testing learning. Our model loss had a good-sized gap between train and test, likely indicating it’s learning well and neither underfitting nor overfitting:





Comparison Between Models

We've selected the following metrics to evaluate our models, with the 'best' model by metric bolded. Due to the different nature of each of the datasets, we decided that we would use different judgement metrics. For Breast Cancer, we chose the weighted F1 score, as we want to limit the number of false negatives and false positives. For Credit Card, we chose the F1-score of the default (1) class, because we only care about a model that can tell us if a customer is likely to default on a payment in the next month. For Wheat dataset, a perfectly balanced multiclass dataset, we selected the AUC-score, because we want a model that is equally good at predicting all 3 classes.

Breast Cancer Dataset: Weighted F1 Score

SVM on Breast Cancer Dataset Results:

- Mean ROC AUC: 99.5%
- Test Set Accuracy: 97.1%
- **Weighted F1 score: 97%**
- F1 scores: 0 = 0.98, 1 = 0.96

Decision Tree on Breast Cancer dataset:

- Mean ROC AUC: 89.77%
- Test Set Accuracy: 94.74%
- Weighted F1 score: 95%
- F1 scores: 0 = 0.96, 1 = 0.93

Random Forest on Breast Cancer dataset:

- Mean ROC AUC: 91.99%
- Test Set Accuracy: 93.57%
- Weighted F1 score: 95%
- F1 scores: 0 = 0.84, 1 = 0.36

XGBoost Breast Cancer Results:

- Mean Roc Auc: 97.94%
- Test Set Accuracy: 96.49%
- Weighted F1 score: 96%
- F1 scores: 0 = 0.97, 1 = 0.95

Deep Learning on Breast Cancer Dataset:

- Mean ROC AUC: 99.5%
- Test Set Accuracy: 95.9%
- F1 scores: 0 = 0.97, 1 = 0.94

Best model for Breast Cancer Dataset, based on Weighted F1 score: SVM.

Credit Card Dataset Model Comparison: F1 of Positive Class

SVM on Credit Card Dataset Results:

- Test Set Accuracy: 82.4%
- F1 scores: 0 = 0.89, 1 = 0.46

Decision Tree on Credit Card dataset:

- Mean ROC AUC: 65.88%
- Test Set Accuracy: 80.03%
- F1 scores: 0 =0.88, 1 = 0.37

Random Forest on Credit Card dataset:

- Mean ROC AUC: 65.40%
- Test Set Accuracy: 74.88%
- F1 scores: 0 =0.84, 1 = 0.36

XGBoost Credit Card Results:

- Mean ROC AUC: 75.30%
- Test Set Accuracy: 85.80%
- F1 score: 0 = 0.89, 1 = 0.45

Deep Learning on Credit Card Dataset:

- Mean ROC AUC: 70.8%
- Test Set Accuracy: 80.7%
- F1 scores: 0 = 88, 1 = 0.51

Best model for Credit Card dataset: Deep Learning.

Wheat Dataset Model Comparison: AUC Score

SVM on Wheat Dataset Results:

- Test Set Accuracy: 98%
- F1 scores: 0 = 0.93, 1 = 1.00, 2 = 0.933

Decision Tree on Wheat dataset:

- Mean ROC AUC: 100.00%
- Test Set Accuracy: 88.89%
- F1 scores: 0 = 0.83, 1 =0.95 , 2 = 0.89

Random Forest on Wheat dataset:

- Mean ROC AUC: 92.12%
- Test Set Accuracy: 88.89%
- F1 scores: 0 = 0.84, 1 =0.97, 2 = 0.88

XGBoost Wheat Dataset Results:

- Mean ROC AUC: OVR: 99.18%
- Test Set Accuracy: 94.34 %
- F1 scores: 0 = 0.69, 1 =0.82, 2 = 0.86

Deep Learning on Wheat Dataset:

- Test Set Accuracy: 56%

Best model for Wheat Dataset: Deep Learning

Qualitative factors in model selection

It is worth noting that we cannot simply use the hard numbers in selecting the 'best' models. We have come to understand that computing limitations are an important factor in model selection. For huge datasets such as Credit Card, SVM and Deep Learning are untenable, unless the user has the patience and computing power to spare. To find best parameters, Decision Tree, Random Forest, and XGBoost performed with the least amount of aggravation.

A different model had the best performance for our chosen metric for each dataset. SVM had the best test set accuracy in 2 out of our 3 models; however, due to the aggravation caused by implementing such a computationally-taxing model, we cannot declare it the winner. XGBoost had great overall performance and was less unwieldy. Decision Tree had amazing mean AUC-ROC score in the balanced multiclass dataset. In testing the 'no free lunch' theorem, we have indeed found that no single model is the best for any given dataset. Further, significant consideration must be given to computational power before selecting SVM or Deep Learning models for classification problems.