Stanly Gomes
Student ID: 801118166

Homework 3

GitHub Repository Link: https://github.com/NaraPvP/IntroToML

For all the problems, an 80/20 training split was done on the dataset. Along with this, the features were scaled using standardization.

Problem 1:

For the Naïve Bayesian model, the following was the classification report (which provides the accuracy, precision, and recall):

	precision	recall	f1-score	support
Benign	0.88	0.88	0.88	33
Malignant	0.95	0.95	0.95	81
accuracy			0.93	114
macro avg	0.91	0.91	0.91	114
weighted avg	0.93	0.93	0.93	114

Along with this, the confusion matrix is shown here for the Naïve Bayesian model:

```
[[29  4]
 [ 4 77]]
```

With the logistic regression classifier that was used in the last homework, this was the classification report and confusion matrix:

	precision	recall	f1-score	support
Benign	0.91	0.94	0.93	33
Malignant	0.97	0.96	0.97	81
accuracy			0.96	114
macro avg	0.94	0.95	0.95	114
weighted avg	0.96	0.96	0.96	114

```
[[31  2]
 [ 3 78]]
```

When comparing these two classifiers, all of the classification parameters (accuracy, precision, and recall) are better for the logistic regression model. This implies that the logistic regression model is better for the cancer dataset compared to the Naïve Bayesian model.

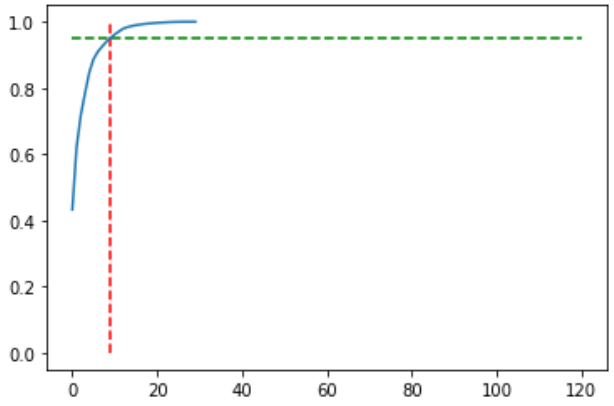
Stanly Gomes
Student ID: 801118166

Problem 2:

After fitting the PCA feature extraction with the training set of our features, I applied the PCA feature extraction on the test set of features. Afterwards, a logistic regression model was trained using the PCA-adjust features as new training and test set. By running the following code, it was determined the optimum number of K in a general case is 9 principal components:

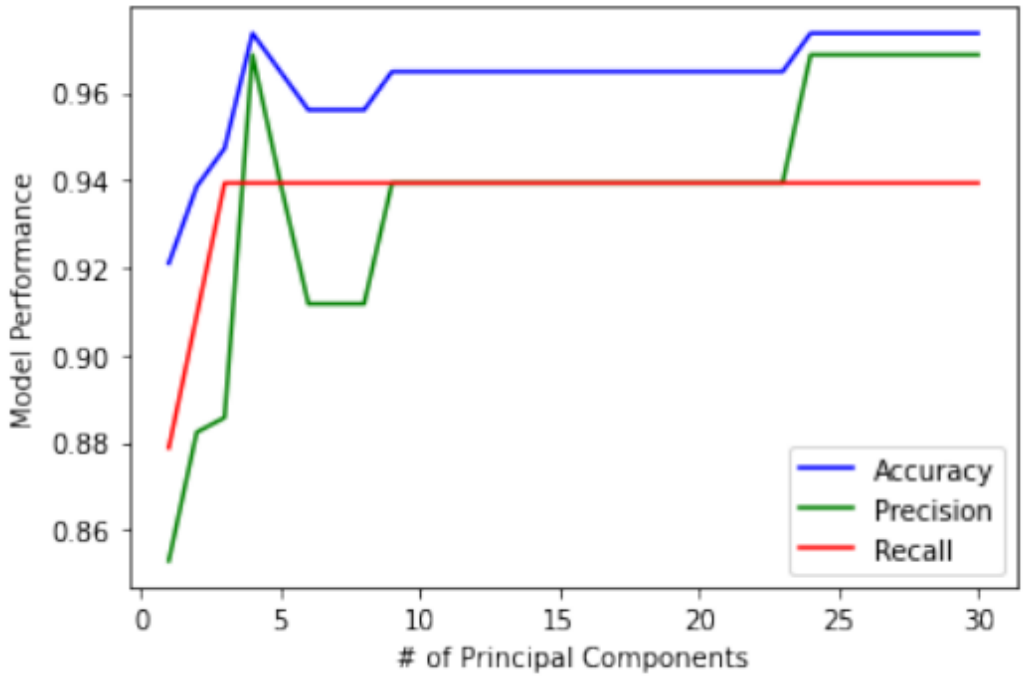
```
#Decide the number of PCA components
from sklearn.decomposition import PCA
pca = PCA(random_state=88)
pca.fit(X_train)
explained_variance = np.cumsum(pca.explained_variance_ratio_)
plt.vlines(x=9, ymax=1, ymin=0, colors='r', linestyle="--")
plt.hlines(y=0.95, xmax=120, xmin=0, colors="g", linestyles="--")
plt.plot(explained_variance)
#principalComponents = pca.fit_transform(x)
#principalDf = pd.DataFrame(data=principalComponents
#                           , columns=['principal component 1', 'principal component 2'])
```

```
[<matplotlib.lines.Line2D at 0x20963c7d2e0>]
```



For the case shown in the code PDF, 4 principal components gave the highest classification accuracy. The plot of the classification report will be provided here:

Stanly Gomes
Student ID: 801118166

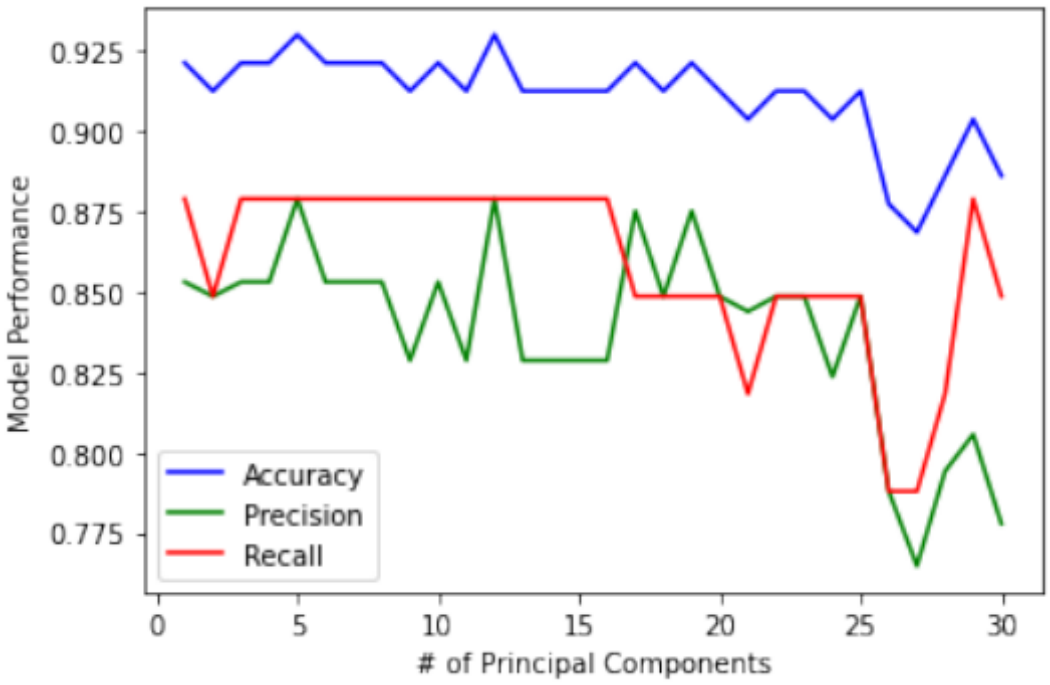


The accuracy is the highest statistic from the classification report, which is plausible due to more predicted values being able to seem accurate as discussed in lecture. We want our precision to be high due to the nature of the PCA feature extraction, which desires to keep as much information from the original set as possible.

Problem 3:

Similarly to Problem 2, the PCA feature extraction was used for training a Naïve Bayes model. The plot of the classification report as k (number of principal components) increase is shown here:

Stanly Gomes
Student ID: 801118166



Like the end of Problem 1, the Bayes classifier does not perform as well as the Logistic Regression classifier. This was to be expected as both datasets had changed equally since Problem 1. Since Problem 2 (Logistic Regression model) had higher statistics overall, this further reinforces that a Logistic Regression classifier is more accurate than a Naïve Bayes classifier.