Capstone Project **Anurag Agarwal**
Machine Learning Engineer nanodegree April 26, 2017

Identification of Fake Currency Notes

1. Definition

1.1. Project Overview

Every major economy in the present times has to deal with the menace of counterfeit currency. Counterfeit currency is the imitation currency produced without the legal sanction of the state or government. Some of its ill-effects on society include a reduction in the value of real money, and increase in prices (inflation) due to more money getting circulated in the economy. This is the reason that fake currency has been used by governments to wage economic wars against each other.

Examples include a high-profile counterfeit scandal which came to light in Hungary in 1926, when several people were arrested in the Netherlands while attempting to procure 10 million francs' worth of fake French 1000-franc bills which had been produced in Hungary [1]. According to the US Department of Treasury, an estimated $70 million in counterfeit bills are in circulation in America [2].

For this reason, we require to adopt counter-measures which can help in preventing this menace. With the use of computers and technology, it is possible to make high quality counterfeit bills that are hard to distinguish from genuine notes. In fact, many counterfeit notes were seized which mimicked many of the security features which are usually built in genuine currency notes. Therefore, we need to innovate new techniques to help people in more accurate identification of fake currency notes in a convenient way.

Machine learning techniques can be utilized to build tools which can help in this task. We can make our computers learn patterns or features to help distinguish between fake and genuine currency. These machine learning models will utilize image processing and pattern recognition techniques to capture and learn the distinguishing features. Using these patterns and information that these models can learn, they will be able to classify a new currency bill as either fake or genuine. Given the high quality of fake currency bills, which are being developed these days with the aim to dupe people and adversely affect the economy of a country, it becomes pertinent to utilize computers to aid in distinguishing them.

Therefore, the goal of this project is to develop such a model which can be adequately trained with the help of relevant data, and then, subsequently used for identifying counterfeit bills with minimal classification errors. Thus, this will be a classification model which will be trained with the help of labeled data. This data will have attributes and associated labels for both genuine and fake currency bills. Our model will learn these attribute values and, with the help of associated labels, attempt to identify decision boundaries which separate the two classes of samples.

The dataset for this problem is taken from UCI Machine Learning Repository [3]. Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.

The features extracted after wavelet transformation have following attribute information:

1. Variance of wavelet transformed image (continuous)
2. Skewness of wavelet transformed image (continuous)
3. Kurtosis of wavelet transformed image (continuous)
4. Entropy of image (continuous)
5. Class of the currency (integer)

The first four features are continuous and define the characteristics of a currency note, while the fifth feature is the actual label of the note, which is 1 for real one, and 0 for fake one. This dataset has 1372 samples, out of which 762 samples belong to fake notes, while 610 samples belong to genuine notes.

In this project, our goal is to build a trained model which can classify new input images based on features extracted after wavelet transformation. The component of processing the image and doing wavelet transformation is not part of the project. This component can be developed as another project. Our goal in this project is to emphasize on the machine learning aspect of the problem. Hence, we are working with the transformed images. To test the performance of our model, we will holdout a test set which will also be labeled, but not visible to our model. We expect to obtain high accuracy for our model as the problem is financial in nature.

1.2. Problem Statement

The problem statement is formally defined as:

“Given a dataset containing features extracted after wavelet transformation of images of currency notes, use the features available in the dataset to design a supervised learning model which can identify whether a currency note is real or fake.”

We begin our investigation by analyzing the dataset for any missing values or any outliers. Since, the four features that we are going to use to train our models are continuous in nature, we will normalize them in the range of 0 to 1. Normalization or scaling ensures that all features are treated equally when applying supervised learners. We would also look if any feature is skewed or has outliers, i.e., it has a sample with vastly different value than those in other samples. In case any feature is skewed we can apply log transform before scaling the feature.

The next step will be to establish a benchmark model that we can use to make performance comparisons. For this problem, we define our benchmark model to be a naïve classifier, which classifies all banknotes as fake. We will define few metrics to evaluate each model including our naïve classifier. We aim that our designed model should perform better than the naïve predictor. Since, this naïve prediction model does not consider any information to substantiate its claim, it helps establish a benchmark for whether a model is performing well. That been said, using a naïve prediction would be pointless as will predict all notes counterfeit and would not identify any note as genuine.

Once our data is in good form, we will split our data into training and test sets. Thereafter, we will investigate different supervised learning algorithms and determine which is best at modeling the data. The various algorithms that we will consider for evaluation are, Gaussian Naïve Bayes, Decision Tree, Support Vector Machines. We will compare each algorithm with naïve predictor to evaluate its performance. Further, we will select one algorithm and fine-tune the chosen model on few of its important parameters using grid search. This will give us our final optimized and fine-tuned model.

1.3. Metrics

We also require to define evaluation metrics that we can use to quantify the performance of our solution, as well as, the benchmark model. The metrics that we will use for our problem are accuracy and f-score. Accuracy is defined as the number of samples correctly classified to the total number of samples. This can be a good metric, but suppose, if it's more detrimental for us if we are not able to correctly classify a fake note, than a real one. Then, the model's ability to recall all fake currency notes is more important than the model's ability to make precise prediction. In this scenario, we can use, F_β score with β = 2, as it weighs recall more than precision. The general form is:

$$F_\beta = (1+\beta^2)\,.\frac{precision\,.\,recall}{(\beta^2\,.\,precision)+recall},$$

Where, Precision is the number of samples correctly classified as positive (as fake notes) to the total number of samples classified as positive, and Recall is number of samples correctly classified as positive to the total number of samples actually labeled as positive. So, Recall is more important for us as we want to correctly classify as many fake notes as possible.

2. Analysis

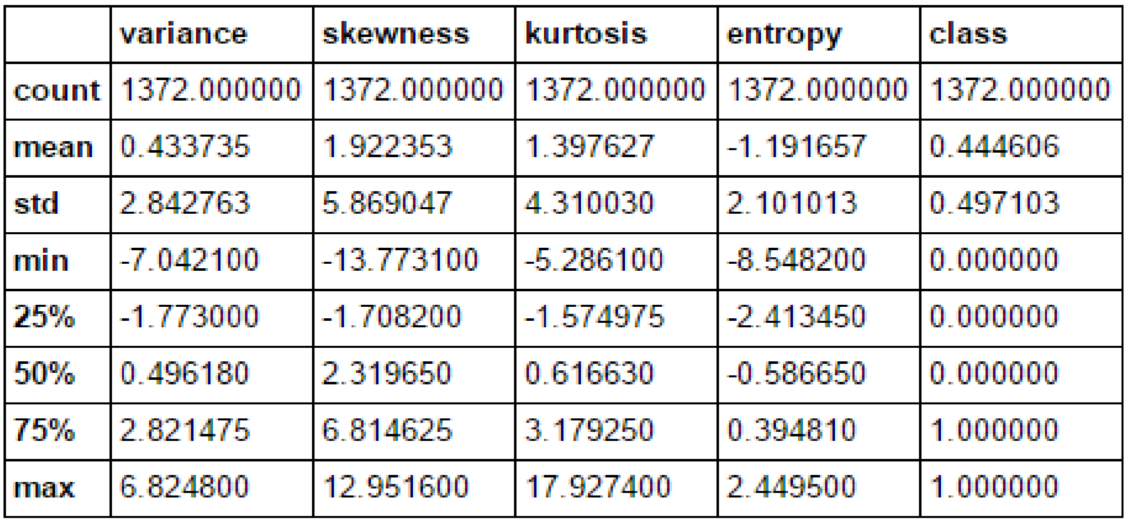
2.1. Data Exploration

As discussed, the banknote authentication dataset consists of five attributes, viz., variance, skewness, kurtosis, entropy and the class. First four features, which were extracted from the wavelet transformation of the image of a currency note, are continuous and will be used for training our model. The fifth attribute is the class label, which in this case can either be, 1 (genuine) or 0 (fake). This dataset consists of 1372 samples, out of which 762 correspond to fake notes, while 610 correspond to real notes. The dataset downloaded from UCI Machine Learning Repository does not have titles for its five attributes, which are added manually to have a better interpretation. A tabular representation of the first few records is shown in Table 1.

	variance	**skewness**	**kurtosis**	**entropy**	**Class**
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Table 1. First five records of the dataset

A brief description of the dataset, including parameters like mean, min, max for each column is given in table 2.



	variance	skewness	kurtosis	entropy	class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.433735	1.922353	1.397627	-1.191657	0.444606
std	2.842763	5.869047	4.310030	2.101013	0.497103
min	-7.042100	-13.773100	-5.286100	-8.548200	0.000000
25%	-1.773000	-1.708200	-1.574975	-2.413450	0.000000
50%	0.496180	2.319650	0.616630	-0.586650	0.000000
75%	2.821475	6.814625	3.179250	0.394810	1.000000
max	6.824800	12.951600	17.927400	2.449500	1.000000

Table 2: A description of the features of dataset

We note following points from the above description:

1. All features, except our target feature 'class', are continuous and have different ranges. They also have different mean and standard deviation.
2. There is large difference in the median value of four continuous valued input features.
3. Therefore, we need to perform scaling to get better results and avoid any type of bias.
4. Our target feature 'class' is binary valued, with value '0' for fake, and '1' for real note.
5. We also observe from the exploration of data that it **does not** have any missing value.

2.2. Exploratory Visualization

A dataset may sometimes contain at least one feature whose values tend to lie near a single number, but will also have a non-trivial number of vastly larger or smaller values than that single number. Algorithms can be sensitive to such distributions of values and can underperform if the range is not properly normalized. We plot a histogram for each feature to identify the presence of any such outliers. The histograms are shown in Figure 1.

We note following points from these distributions:

1. There are no features in this dataset for which a sample has very large or very small value than the values in the rest of the samples.
2. Since, there are no such outliers, we do not need to perform a log transformation.
3. 'Variance' and 'skewness' attributes have quite evenly spread distribution.
4. 'Kurtosis' has a positively skewed distribution, and most of the records have kurtosis value less than 3. Distributions with kurtosis less than 3 are said to be *platykurtic*, which means that the distribution produces fewer and less extreme outliers than does the normal distribution which has kurtosis value 3. It means that the corresponding images should have less noise.
5. Entropy has a negatively skewed distribution, which means more samples have higher entropy, which further implies that more images have higher contrast.
6. Scaling is needed to bring all features in the same range.

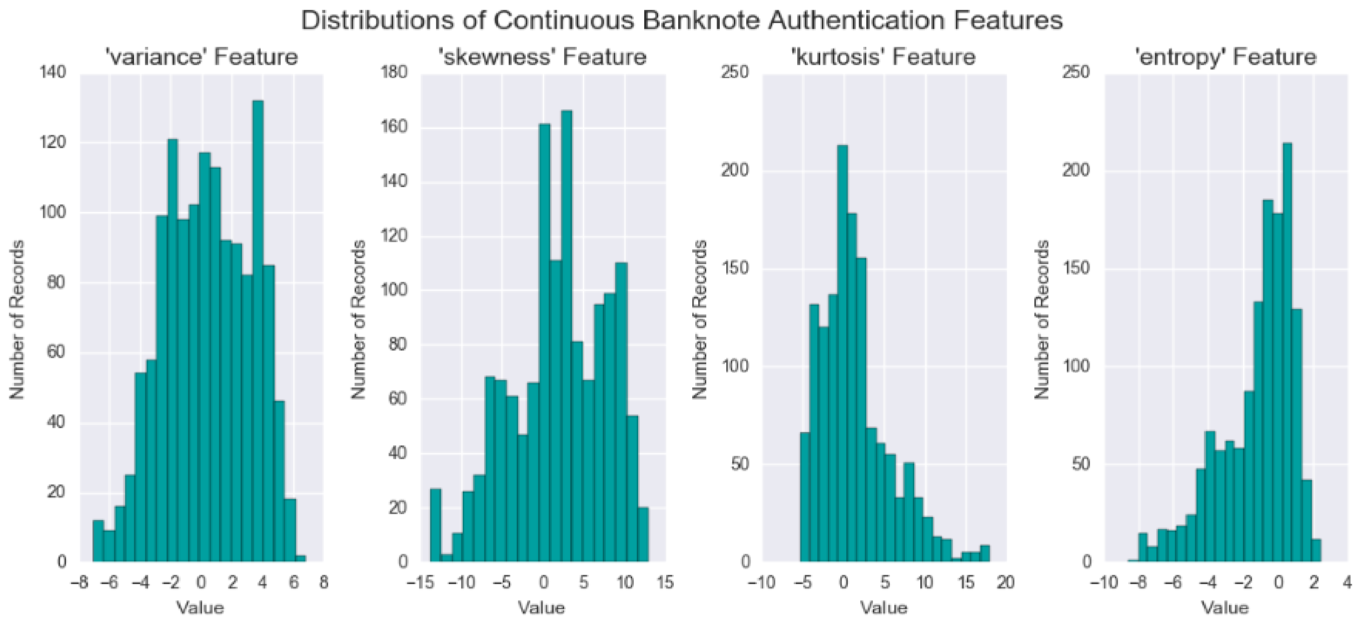


Figure 1: Distributions of features in the banknote authentication dataset

The next analysis that we would like to perform is to create six classification scatter plots, using two of the four numerical attributes for each display, and use different colors for objects belonging to class 1 and class 0. This would help us identify the pair of features which separate the two classes more strongly. In these scatter plots, displayed on the following page, class 1 (real note) is colored green, while class 0 (fake note) is colored red.

We note the following points from the scatter plots:

1. Skewness, Variance pair most strongly separates the two classes of notes, and a piecewise linear boundary is going to produce good results.
2. Entropy, Kurtosis pair least strongly separates the two classes of notes, and algorithms will have tough time in doing good classification if we rely only on these two features.
3. Variance is a prominent feature that distinguishes genuine and fake notes. Real currency bills predominantly have low variance, while fake bills have high variance.
4. Entropy is the feature which is least important in terms of class-separability.

As we see in the scatter plots, there is a good separability between the two classes for many pairs of features. So, it looks that our features are very good in distinguishing two classes of banknotes, and we should get high prediction accuracy with an appropriate algorithm.

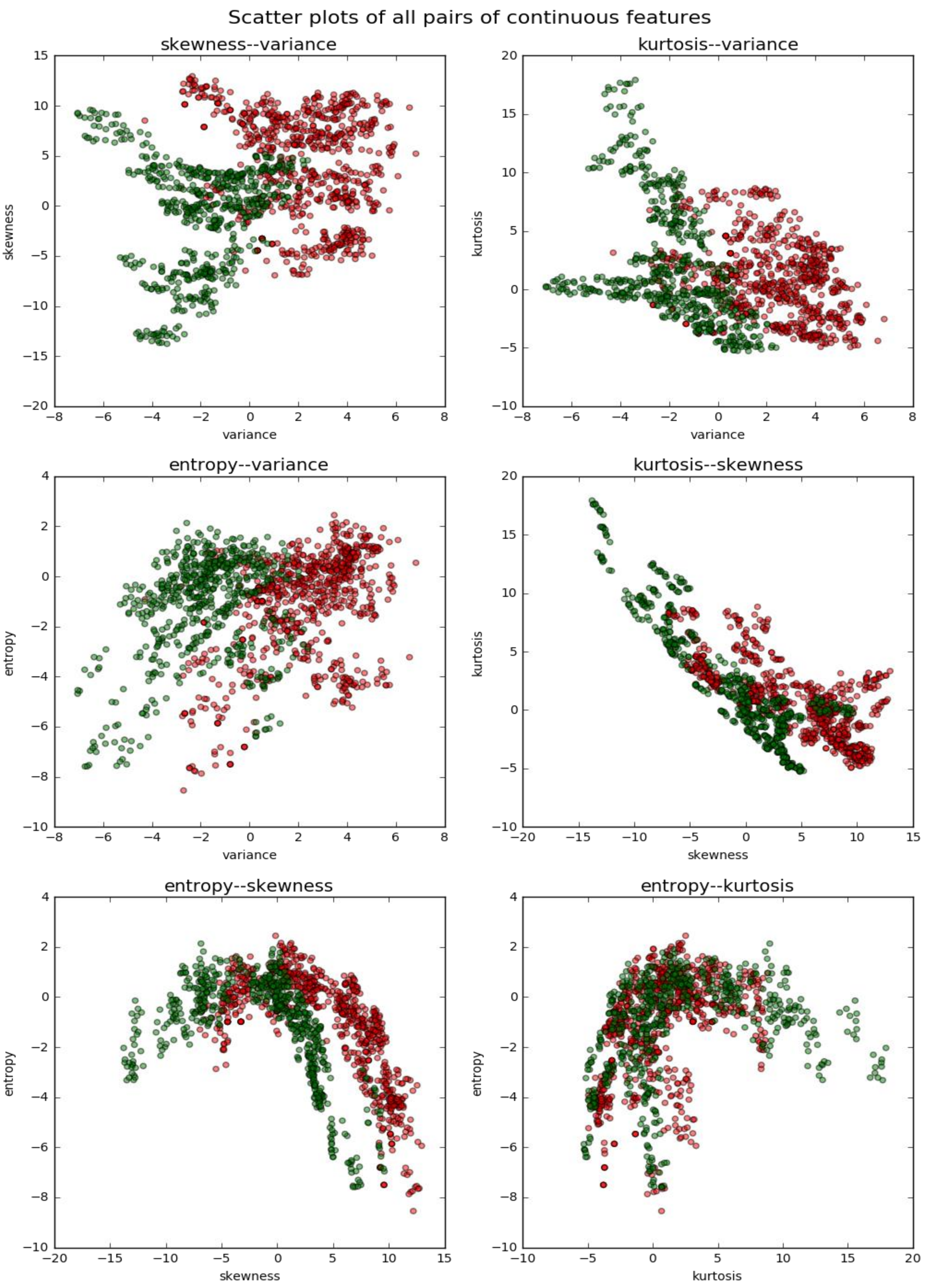


Figure 2: Scatter plots of all pairs of continuous features

2.3. Algorithms and Techniques

Three supervised learning approaches are selected for this problem. These algorithms are chosen such that their approaches are fundamentally different from each other, so that we can cover a wide spectrum of possible approaches. We consider following three algorithms for analysis, and would compare their performance with our benchmark model.

1. Support Vector Classifier
2. Gradient Boosting Classifier
3. K-Nearest Neighbors Classifier

Before we proceed further, we would like to discuss these algorithms in brief.

2.3.1. Support Vector Classifier:

Support Vector Machine is a supervised learning technique that constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier. Some of the various parameters that it manipulates, include:

- kernel: It defines the type of kernel to be used, like, 'linear', 'poly', 'rbf', 'sigmoid'.
- C: Penalty parameter of the error term.
- degree: Degree of the polynomial kernel function ('poly'). Default is 3, ignored by other kernels

Advantages:

- Effective in high dimensional spaces.
- Performs well with non-linear decision boundaries if appropriate kernel is used.
- Also effective where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different kernel functions can be specified for the decision function.

Disadvantages:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- Expensive to train.

Support Vector Classifier is going to work for the current problem, since, it is a binary classification problem and the number of features is quite small than the number of samples. Moreover, in the scatter plots that we made, even with two features, we were able to get good classification. So, after projecting to higher dimensional space, we should get even better classification.

2.3.2. Gradient Boosting Classifier

Gradient Boosting model is an ensemble of weak learners such that this aggregation eventually provides a very robust model. A weak learner is basically a simple prediction model which is defined as one whose performance is at least slightly better than random chance. Thus, it produces a very accurate prediction rule by combining rough and moderately inaccurate rules. This weak learner or prediction rule is in the form of decision trees, which are essentially a flowchart of Yes/No questions. The weak learners are added sequentially, one in each subsequent step, and by weighting the observations, so as to put more weight on difficult to classify instances and less on those already handled well.

Advantages:

- Easy to interpret and explain.
- It can effectively capture complex non-linear function dependencies.
- Extremely flexible and can easily be customized to different practical needs.
- Aggregates the output of many weak learners to build a more robust model.

Disadvantages:

- GBMs consume a lot of memory when the number of boosting iterations for learning become extensively large.
- Using the fitted GBM model to obtain predictions can be time-consuming, when the ensemble is considerably large, since all weak learners are necessarily consulted.
- The learning procedure is essentially sequential and has problems with parallelization by design, which makes it on average slower to learn.

Gradient Boosting Classifier is going to work for the current problem, as the data is structured in a way that it is in appropriate form to ask layered questions like in decsion trees. Also, with small number of features, a boosted ensemble model should not take much time during training or testing phase. So, GBM is going to provide us a robust model using an ensemble of weak learners.

2.3.3. K-Nearest Neighbors Classifier

KNN is a class of lazy learners which classifies a given data point by looking at its neighbors and assigning weights to them in such a way that the closest neighbors have a greater say in determining the class. Here, distance can be Euclidean, Minkowski, etc. KNN is said to a lazy learner as it does not build a model on training data and classifies a point in test set only by looking at its nearest neighbour.

Some of the hyperparameters that KNN manipulates includes:

1. n_neighbors: Numbers of neighbors to be used for classification
2. weights: degree of influence each data points possesses.
3. algorithm: Algorithm used to compute the nearest neighbors (ball_tree, kd_tree)

Advantages:

1. Simple to implement.
2. Flexible with regard to feature or distance choices.
3. Naturally handles multi-class cases.

Disadvantages:

1. All heavy computational work happens during testing.

K-Nearest Neighbor Classifier is going to work for the current problem as our dataset is not very huge. So, making predictions using KNN would not require much time. Also, since, KNN is non-parametric, we do not need to make any assumption about data distribution, and the decision boundary can take on any form. So, it should be good applying KNN in the present context.

2.4. Benchmark

A benchmark model is defined to provide a threshold and compare performance of the solution obtained by the trained model. For the Banknote Identification problem, we will define our benchmark to be a naïve classifier, which classifies all currency notes as fake. The accuracy of this naïve classifier will be the number of samples of fake banknotes to the total number of samples in the data. Our model should perform far better than the naïve approach to be any worthy for intended use. Since, this naïve prediction model does not consider any information to substantiate its claim, it helps establish a benchmark for whether a model is performing well. That been said, using a naïve prediction would be pointless as it will predict all notes as counterfeit and would not identify any sample as genuine.

We calculate the accuracy of the naïve predictor model as follows:

$$Accuracy = \frac{Samples\ correctly\ classified}{Total\ number\ of\ samples} = \frac{Number\ of\ fake\ banknotes}{Total\ number\ of\ samples} = \frac{762}{1372} = 0.5554$$

F-beta score is calculated from the precision and recall. Precision here is same as accuracy and value of recall is 1. We take beta as 2 to put more weight on recall. Thus F-beta score is:

$$F_\beta = (1+\beta^2)\ .\ \frac{precision\ .\ recall}{(\beta^2\ .\ precision) + recall} = (1+2^2).\frac{0.5554 * 1}{(2^2 * 0.5554) + 1} = 0.8620$$

So, for the benchmark model, accuracy is 0.5554, and F_β score is 0.8620 with β value of 2.

3. Methodology

3.1. Data Preprocessing

Data Preprocessing is necessary to bring data into good shape, remove any abnormalities, or modify any characteristics, before feeding it into an algorithm. From the data visualization section, we noted following points regarding our data.

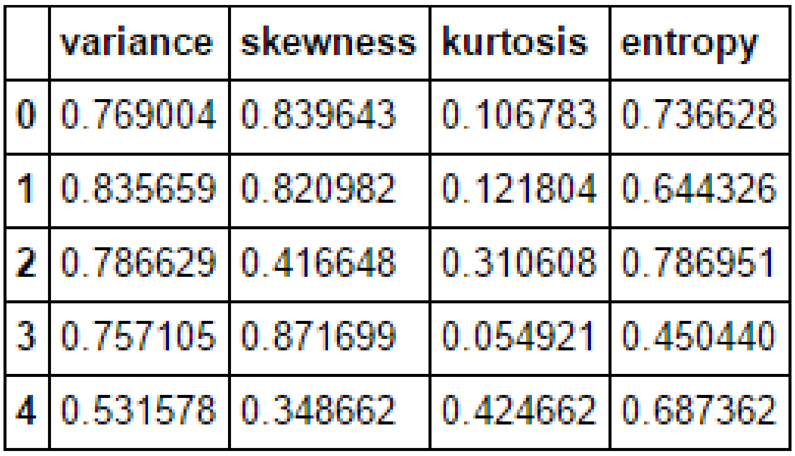
1. The dataset does not have any missing value.
2. All the input features are continuous and have different range, so we need to perform normalization to scale all features in the range of 0-1.

3.1.1. Normalization

From analyzing the data, we identify that we need to perform scaling on the four continuous features to bring them down in range of 0 to 1. This way, the classifier will treat all features equally. We use the following formula to normalize our features:

$$X_{norm} = \frac{X_{current} - X_{min}}{X_{max} - X_{min}}$$

We perform scaling so that the supervised learning algorithm is not biased towards any feature. The first five samples after normalization are reproduced here.



	variance	skewness	kurtosis	entropy
0	0.769004	0.839643	0.106783	0.736628
1	0.835659	0.820982	0.121804	0.644326
2	0.786629	0.416648	0.310608	0.786951
3	0.757105	0.871699	0.054921	0.450440
4	0.531578	0.348662	0.424662	0.687362

3.2. Implementation

Now that our data is in good shape after normalization, we should split it into training and test sets. Then we train our classifier with the training set, and then evaluate its performance on the test set. We will then compare the results with the benchmark model.

3.2.1. Splitting data

The reason we perform splitting is to test the trained model over samples which it has never seen before. This way, we make sure that the model has extracted classification patterns from the training samples and has not memorized them. For the given problem, to split our data, we use a function train_test_split available in cross_validation module of sklearn library. This function does two tasks which are important in the current context.

1. It shuffles the dataset so that both training and test sets have nearly equal number of samples from both classes.
2. After shuffling, it performs the split. We can specify what fraction of the total data to be included in the training or test set.

After splitting, the function returns four lists, comprising input features of the training set, input features of the test set, target labels for the training set, and target labels for test set. We make split such that training set has 60% of samples, while test set has 40% of samples.

3.2.2. Creating Training and Prediction Pipeline

We define a method called 'train_split' that takes as input the following parameters: learner, sample_size, X_train, y_train, X_test, y_test. It returns the accuracy and F-beta score on training and test set. The function fits the 'learner' on the training data of size defined by 'sample_size', and computes the time spent on training. Next it gets predictions on the test set, and 300 samples of training set, and compute the time spent on prediction. Finally, we compute the accuracy and f-beta scores on both training (300 samples) and test set. We also compute confusion matrix to get better visualization of the predictions.

We train the three classifiers that we have chosen, each with 5%, 20%, and 100% of the training data as sample size, so that it can be seen how performance varies with varying sizes of train set. The bar-graphs depicting the accuracy and f-scores for the three classifiers on training and test sets are depicted in Figure 3.

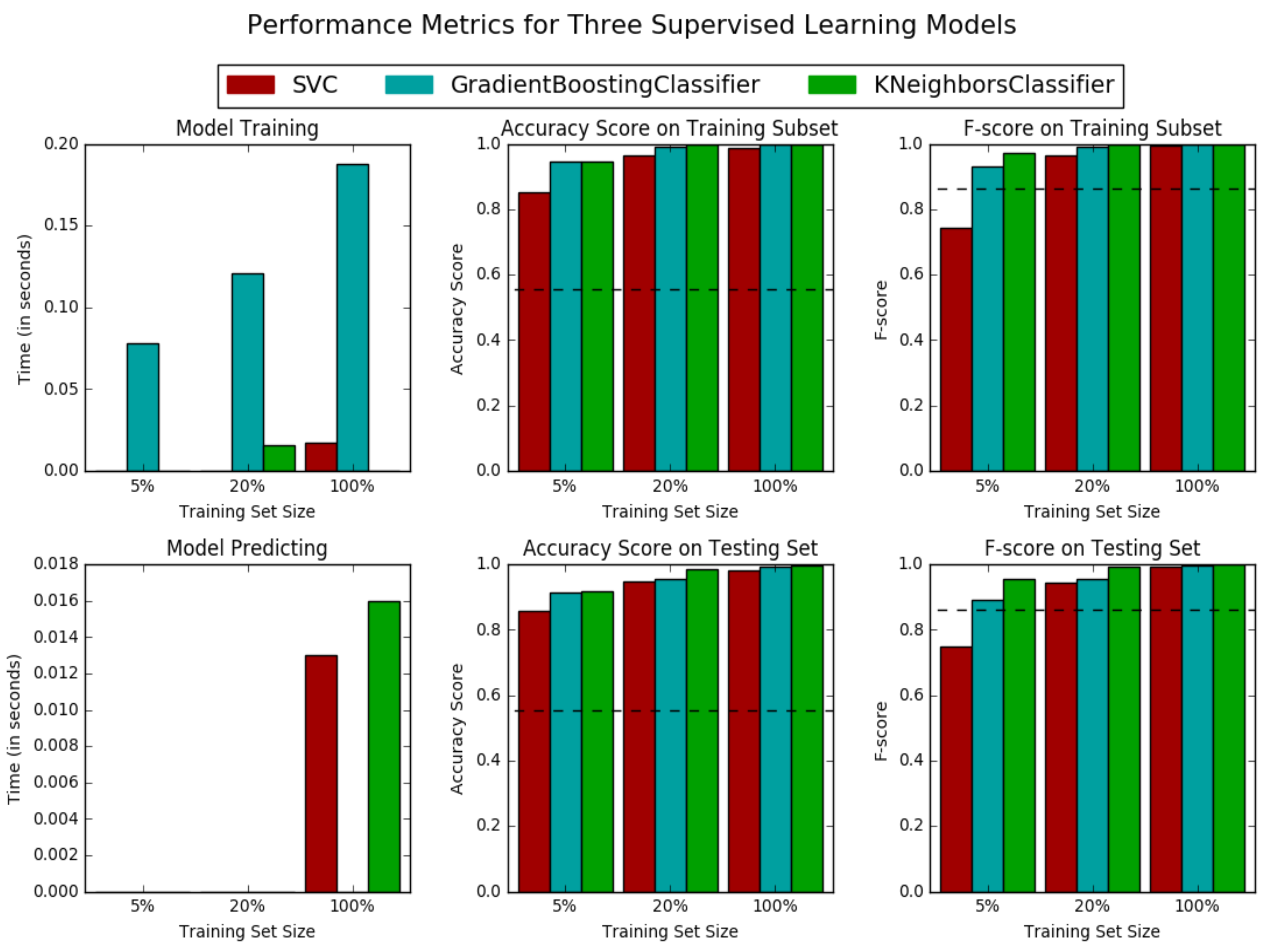


Figure 3: Performance Metrics for three supervised learning models

We get the following scores when entire train set is used for training.

	SVC	**Gradient Boost**	**K-Neighbors**
Train set accuracy	0.9867	1.0	1.0
Test set accuracy	0.98	0.9945	0.9982
Train set f-score	0.9941	1.0	1.0
Test set f-score	0.9914	0.9953	0.9992

The Confusion matrix of prediction of test data is also provided below for all three classifiers:

SVC	
284 (TN)	11 (FP)
0 (FN)	254 (TP)

Gradient Boosting	
293 (TN)	2 (FP)
1 (FN)	253 (TP)

K-Nearest Neighbors	
294 (TN)	1 (FP)
0 (FN)	254 (TP)

Following points are noted from the above visualization and metrics:

1. All three algorithms perform exceptionally well with the test set, which is required from an application that identifies fake currency notes.
2. SVC and KNN algorithms detect all fake currency notes, i.e., they have a recall of 1.0.
3. GBM and KNN have lowest false positives, and thus have highest precision.

4. KNN registers an accuracy of 0.9982, and F-score of 0.9992 on the test set. It classifies all test samples correctly except just one false positive. This is exceptionally good.
5. KNN Classifier outperforms other two algorithms on both training and test sets, for all sizes of training samples.
6. From the visualizations and the evaluation metrics, it is visible that KNN is a clear winner, though other algorithms need optimizations to perform equally well.
7. Though gradient boosting takes some time for training, and KNN takes time during testing, the overall time spent is very small, of the order of fraction of a second.
8. Such good results were anticipated from the scatter plots that we made earlier.

Since K-Nearest Neighbor Classifier has given the best results, and classified all fake currency notes correctly, it is the algorithm of our choice for solving this problem.

3.2.3. Refinement

Though we have got such good results with K-NN Classifier that there is no need of further refinement, but we will still perform a Grid Search and try to optimize on the parameters to verify that no improvement is possible by tweaking of parameters.

The parameters that we would like to optimize for KNN algorithm are:

1. **‘n_neighbors’**: number of neighbors to use. We consider six values [1, 2, 3, 4, 5, 6]
2. **‘weights’**: weight function used in prediction. We take two values [‘uniform’, ‘distance’]. ‘uniform’ value weighs each point in neighborhood equally, while ‘distance’ value puts more influence on closer neighbors.
3. **‘algorithm’:** algorithm used to compute the nearest neighbors. We take possible values as [‘ball_tree’, ‘kd_tree’]

With these parameters to optimize for, we perform a GridSearchCV to find the best combination of parameters. The GridSearchCV object takes following parameters as input:

1. An object of the classifier we want to optimize,
2. A dictionary of parameters and their possible values. Here, we have taken ‘n_neighbors’, ‘weights’, ‘algorithm’, to be the parameters to look out for.
3. A scoring function which will be used to evaluate a fitted model.
4. A cross-validated ‘StratifiedShuffleSplit’ object which gives stratified randomized folds, made by preserving the percentage of samples for each class.

Then GridSearchCV performs grid search on the classifier using the scoring method, and returns a grid search object, to which we fit our training data to find the optimal parameters.

Another thing that we do for refining the KNN Classifier is to separately iterate over number of neighbors to find the best value of ‘n_neighbors’. We do this to find how the accuracy and f-scores vary with increasing value of ‘n_neighbors’.

3.2.4. Feature Importance

The last thing that we will do is determining which features provide the most predictive power. By focusing on the relationship between only a few crucial features and the target label we simplify our understanding of the phenomenon. Though we only have a small set of features, and we have some rough idea from the scatter plots as well, we will now quantify this idea of determining feature importance. To do this, we choose a scikit-learn classifier (e.g., adaboost, random forests) that has a 'feature_importances_' attribute, which is a function that ranks the importance of features according to the chosen classifier. We choose two algorithms, viz., Random Forests and Gradient Boosting to rank the features and both provide similar results.

4. Results

We considered three classifier models for the given problem, viz., Support Vector Classifier, Gradient Boosting Classifier, and K-Nearest Neighbor Classifier. We evaluated the performance of these classifiers using accuracy and f-beta score evaluation metrics. We trained our classifier model on various sizes of training data, to analyze how it affects the prediction scores. Prediction scores were obtained for test data, and for a subset of training data. We also calculated the time that is taken for training and prediction. Finally, we also refined the selected model using grid search.

4.1. Model Evaluation and Validation

All three classifiers that we discussed produced good accuracy and f-beta scores, but KNN classifier stood out among them. So, it is our final choice of model as a solution to the given problem. We use following criteria to select KNN as the final model.

Firstly, the KNN classifier has better accuracy and f-beta scores than other two classifiers, on both training and test sets. The accuracy and f-beta score on training set is 1.0, and on test set 0.9982 and 0.9992, respectively. On the test set, it classified all fake notes correctly and misclassified only one genuine note.

Secondly, the model outperformed other two classifiers consistently on various training set sizes. Even when the size of training set was 5% of its actual size, prediction accuracy was more than 90% on both training and test data. This signifies the robustness of the algorithm.

Thirdly, the results have been so phenomenal with KNN, that even fine-tuning the parameters of model with grid search produced same results. A snapshot of results is shown in Figure 4.

On viewing the outcome of grid search, we find that the optimized model has parameter 'n_neighbors' = 1. 'n_neighbors' defines the number of closest neighbors that voted in the prediction outcome. But initially, we took the value of 'n_neighbors' to be 5. This motivated us to view how number of neighbors affects the performance of KNN in the context of this problem. The results are depicted in Figure 5. We observe that the accuracy and F-beta score are same for up to ten closest neighbors. This again highlights the robustness of KNN Classifier for the given problem.

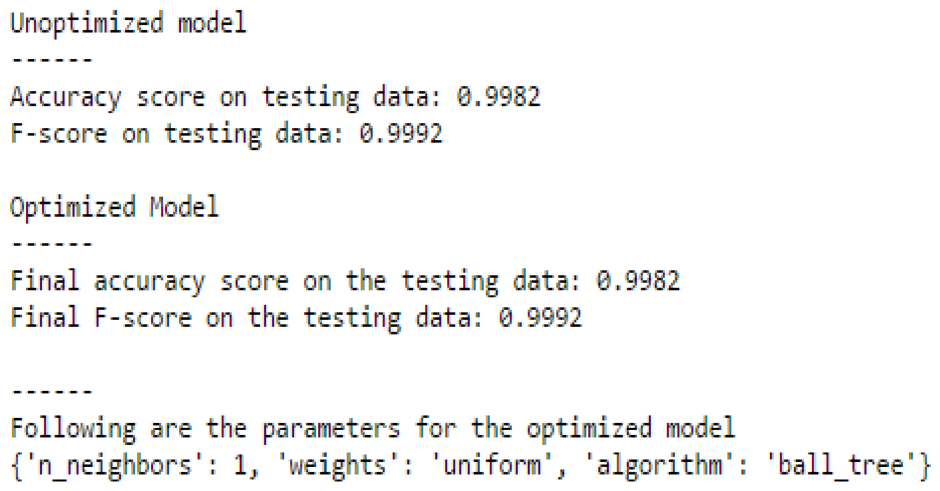
So, what value of 'n_neighbors' we should take in the final solution model? Value of 1 seems very simple, and makes our model very sensitive to the input data. A single noisy point or outlier may change the actual label of some samples. Value of 5 or more makes the model little complex. So, we should go with value of either 3 or 4. But 4 is an even number, so we will go with 3 as the value of 'n_neighbors' in our final model of KNN classifier.

So, we see that the model is quite robust as it gives surprisingly similar results on various training set sizes as well as on different values of its parameter 'n_neighbors'. The robustness of model further firms our belief on the choice of this model as a final solution.

The results of the model can be trusted on many accounts. Firstly, the test set was different from training set. So, the model was evaluated on unseen samples in the testing phase. Secondly, we also performed grid search with cross-validated randomized folds to negate any chances of overfitting and got similar results. Thirdly, we have got similar results with other models as well. So, yes, the results can be trusted.

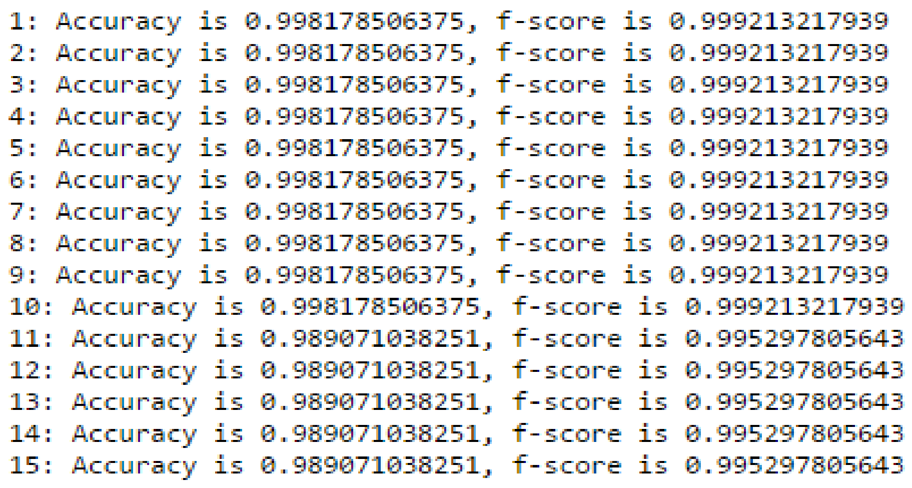
Finally, we believe that the model is reasonable and aligns with our expectations of a good solution as it performs exceptionally well and is quite robust. The model is also performing very fast and producing good classification results in fraction of a second.

As an aside, we also compute the predictive power of each feature, using two algorithms, viz., Random Forests and Gradient Boosting Classifier. These algorithms have 'feature_importances_' attribute that assigns a number to each feature in the range of 0-1, such that their sum is 1. The results are displayed in Figure 6. We can see that first three features, viz., variance, skewness and kurtosis have the most predictive power, while the influence of entropy is minimum.



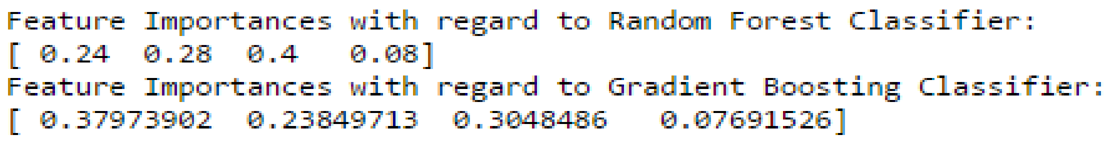
```
Unoptimized model
------
Accuracy score on testing data: 0.9982
F-score on testing data: 0.9992

Optimized Model
------
Final accuracy score on the testing data: 0.9982
Final F-score on the testing data: 0.9992

------
Following are the parameters for the optimized model
{'n_neighbors': 1, 'weights': 'uniform', 'algorithm': 'ball_tree'}
```

Figure 4: Outcome of grid search on KNN Classifier



```
1: Accuracy is 0.998178506375, f-score is 0.999213217939
2: Accuracy is 0.998178506375, f-score is 0.999213217939
3: Accuracy is 0.998178506375, f-score is 0.999213217939
4: Accuracy is 0.998178506375, f-score is 0.999213217939
5: Accuracy is 0.998178506375, f-score is 0.999213217939
6: Accuracy is 0.998178506375, f-score is 0.999213217939
7: Accuracy is 0.998178506375, f-score is 0.999213217939
8: Accuracy is 0.998178506375, f-score is 0.999213217939
9: Accuracy is 0.998178506375, f-score is 0.999213217939
10: Accuracy is 0.998178506375, f-score is 0.999213217939
11: Accuracy is 0.989071038251, f-score is 0.995297805643
12: Accuracy is 0.989071038251, f-score is 0.995297805643
13: Accuracy is 0.989071038251, f-score is 0.995297805643
14: Accuracy is 0.989071038251, f-score is 0.995297805643
15: Accuracy is 0.989071038251, f-score is 0.995297805643
```

Figure 5: Number of closest neighbors, and corresponding accuracy and F-score of KNN



```
Feature Importances with regard to Random Forest Classifier:
[ 0.24  0.28  0.4   0.08]
Feature Importances with regard to Gradient Boosting Classifier:
[ 0.37973902  0.23849713  0.3048486   0.07691526]
```

Figure 6: Predictive Power of the four features, in the order, variance, skewness, kurtosis, entropy

4.2. Justification

The benchmark model for the given problem is a naïve predictor which predicts every note as fake currency bill. The evaluation metrics for the benchmark was computed as follows:
Accuracy = 0.5554,   F-beta score = 0.8620 (β = 2)
The evaluation metrics for our solution model of KNN Classifier is as follows:
Accuracy = 0.9982   F-beta score = 0.9992 (β = 2)
This shows that our solution model way outperforms the benchmark in terms of evaluation metrics.

Moreover, the proposed solution correctly classifies all fake notes and most of the genuine notes. The solution model utilizes the information of input features to identify patterns that distinguishes the two classes. On the other hand, the benchmark model does not utilize any information to make the prediction. Besides, it classifies all notes as fake, and thus, cannot be put to any real use.

Another point to note is that the accuracy of the benchmark model is dependent on the percentage of samples of fake notes in the dataset. More is the percentage of fake notes, more is the accuracy. On the other hand, our proposed solution is robust in this regard, and has consistently performed well over various training sizes.

Therefore, our solution model is way stronger than the benchmark model, as it uses domain information to make informed predictions and is also quite robust in its performance. We believe the model can be adopted for real use, and even mobile apps can be developed for detection of fake notes [4]. Though, to work as a mobile app, the component for capturing images, removing noise, then applying wavelet transformation, and finally extracting features should also be developed.

5. Conclusion

5.1. Free-Form Visualization

We defined Confusion Matrix for each algorithm that was discussed to illustrate their performance on the test data. It is necessary for any model to correctly classify all counterfeit bills for the successful deployment of the application. It was visible that KNN-Classifier identified all fake notes correctly and misclassified only one genuine note. In the context of this problem, a few false positives are acceptable, but any false negative is not. So, we have been successful in achieving our goal with the model of our choice. Moreover, we also observed that the KNN Classifier is robust in its performance with regard to the training set size, and its parameters (like n_neighbors). This is necessary for any application that has financial implications.

5.2. Reflection

The project began with identifying an interesting dataset that could solve a real-world problem. Banknote Authentication dataset available on UCI Machine Learning Repository seemed to be a good choice. The first step for any problem like this is to analyze the dataset. We performed statistical analysis and reflected upon the features of the dataset. We then made some visualizations which reflected upon the relationship among various features, and their classification ability. We scaled the continuous features of the dataset in the range of 0-1.

Further, we split the data into training and test set. We took 40% of data to be in test set. We also made a benchmark model. A naïve predictor that classified all notes as fake was taken to be as benchmark. We defined metrics that we used to compare the performance of our model. We took accuracy and F-score to be the performance metrics.

Next, we chose three algorithms, SVM, Gradient Boosting and KNN, to train the dataset with, and compared their performances using accuracy and f-beta scores. We also made confusion matrix, which reflected upon the number of samples each model classified correctly. After analyzing the models, we chose KNN to be our final solution model. We performed Grid Search with K-fold cross-validation on KNN. We evaluated it for different values of nearest neighbors. We found that the model gave same performance for various values of 'n_neighbors', and different combination of hyperparameters. This reflected upon the robustness of the model.

It is quite interesting to note that a simple algorithm like KNN, would give such phenomenal results with such impressive robustness. The results are sufficient to firm my belief in the fact that it is the data, not the algorithms, that play the most decisive role. Algorithms are only tools in the hands of a data scientist, and there is no simple rule that defines which algorithm should work with which data.

Though I did not face any difficulty in executing the project, I would like to express my gratitude for the publisher of the dataset, without which it would have been almost impossible for me to carry out this project. The most difficult part for me seems to be capturing images, applying wavelet transformation, and then, extracting features, which was done by the owner of the dataset.

To finally conclude, I want to state that I am more than satisfied with the results, and feel that the final model has fulfilled my expectations for a highly accurate and robust solution.

5.3. Improvement

Though we have got exceptionally good results, there are still chances of improvement, as in a financial application like this, we always aim to be 100% accurate. Possibly, we can improve by having larger dataset which captures more distinguishing patterns, and exploring other algorithms like Neural Networks, which we did not touch in this project.

Another way to solve this problem can be through the use of deep learning. We can train a convolutional neural network, directly over the captured images, to build a classification model. Another possibility which can be explored is to use pre-trained deep neural networks like AlexNet or VGGNet. I strongly feel that these methods should be explored as potential solution candidate.

6. References

1. https://en.wikipedia.org/wiki/Counterfeit_money
2. https://en.wikipedia.org/wiki/Counterfeit_United_States_currency
3. https://archive.ics.uci.edu/ml/datasets/banknote+authentication
4. Lohweg, Volker, et al. "Banknote authentication with mobile devices." *IS&T/SPIE Electronic Imaging*. International Society for Optics and Photonics, 2013