Detection of Fake Currency using Machinelearning

Manohar Venkat G
dept. Cyber Security in Systems and Networks
Essentials of Cyber Security
Mysore, India
manumw21@gmail.com

***Abstract*—This document presents the problem of solving detection of fake bank notes using machine learning. The algorithms used are Support Vector Machines (SVM), Random Forrest (RF) and Decision Tree (DT). For Analysis of the three model I used metrics like accuracy, False Positive rate (FPR), False Negative Rate (FNR), precision, Matthew's Correlation Coefficient (MCC), Specificity, confusion matrix and recall. To further analyse I used learning and validation curves and to compare my models I used AUC_ROC score and ROC curve which all of these lead me to decide on a model and what must be done to solve the given problem.**

***Index Terms*—Machine Learning, Fake Bank Notes.**

I. Introduction

For any transactions to happen in market we use currency (banknotes) to trade for products. It is an important asset for any nation but there are also fake notes which disrupt this flow of trade and create chaos which could lead to a huge amount of loss in money [6]. Fake currency can be identified in two ways, one is using chemical methods and another is physical that is by identifying the secret codes embedded in the bank notes. Machine learning is definitely suited for this problem as rather than manually checking for bank notes and determine whether its fake or real, we could just take the photo of the currency and convert it using wavelet transform tool and feed this input to our model as an input to determine whether currency is fake or real.

Problem Statement: To predict whether the banknote is forged or genuine with the data set obtained .

This is clearly a classification problem so I will use machine learning algorithms, Support Vector machine (SVM), Decision Tree (DT) and Random Forest (RF). I will compare their learning curves and determine which is more accurate.

II. Related Work

In the past many have worked on this data set [3]. One such paper was from Anju Yadav and her team [14] where they worked on this problem using multiple algorithms like DF, DT, SVM, KNN and Naive Bayes and some metrics were used to evaluate their model. The data set was not scaled in their work and also they haven't mentioned which cross vlaidator they used. I used KFold cross validator and standard scaler to scale my data. Another work done by Anurag [1] has used the same data set and trained on th same models mentioned. His benchmark model was Naive Bayes. I did not choose naive bayes as my benchmark as what if it declares every data as forged data and went with a random guesser, dummy classifier. Many other work [5] have used similar method to solve the problem. Main thing that I observed was that this being classification problem it was guaranteed that Decision and Random Forest would come up.

III. Methodology

A. Dataset

The data set is taken from UCI machine learning repository for 'banknote authentication Data Set' [3]. The data set comprises of 5 features that are "Variance of Wavelet Transformed image, Skewness of Wavelet Transformed image, Kurtosis of Wavelet Transformed image, Entropy of image, Class". All features are important and therefore no columns can be discarded. The data set comprises of physical attributes of the bank notes scanned. The images were taken from bank notes and forged bank notes. These images were generally taken from industrial camera. Using wavelet transform tool, the images were transformed . Number of rows: 1372. '0' denotes genuine and '1' represents fake. There are 762 genuine bank notes (0) and 610 forged bank notes (1) [3].

The data set does not have any missing values and outliers, which means we do not have to modify our data set. I still had to scale the data so that our features are treated equally when running our algorithms. I used standard scaler [9] because if any feature has high variance that are larger than other features, then it would make our estimator not learn from other features correctly if standard scaler not used, hence I used standard scaler to scale our features.

B. Experiments Run

I ran 3 algorithms: SVM, RF, DT, and a dummy classifier as two different train-test splits.

- Dummy classifier: I have used dummy classifier as my benchmark. If my scores produced by the other three algorithms used couldn't do better than this dummy classifier, then I would have had to reject the underperforming model. But All the 3 algorithm used by me had very good scores [12].
- SVM: In the SVM I used the Support Vector Classifier(SVC) to train the training data set. There are two strategies that I used, linear and rbf. I used linear strategy because I am dealing with two dimensional space. There are other strategies present like poly, sigmond and

precomputed, which are not a good match for our dataset because I don't know our gamma value, that is I don't know how much does our single training examples affects in training. So I will use a default strategy 'rbf' with its default gamma value to know how well its getting trained [10].

- Random Forest: I used RF to to train the model and got very good results. I had to vary the hyper parameters to get a proper working model. The hyper parameters were max_depth = 100, min_samples_leaf = 50 and min_samples_split = 50 which gave me a proper result. Varying on the hyper parameters, if I decrease the value of min_samples_split and min_samples_leaf it caused under fitting and on increase of those hyper parameters it caused over fitting. The varying I had done was in range of 20, 40, 60...600. The hyper parameter max_depth did not have much impact after 50 and gave the same learning and validation curve points. So the most important hyper parameters to be considered here were min_samples_split and min_samples_leaf and also max_depth is important, but not as much as the other two [8].
- Decision Tree: They have the same hyper parameter as that of Random Forest, that is max_depth, min_samples_split, min_samples_leaf. The hyper parameters values I chose were max_depth=10, min_samples_split=10, min_samples_leaf=20 which gave me a proper result. On decreasing the values created under fit and increase in their values created an over fit also the performance had decreased too, dropped to less than 0.7. which was very bad. So I choose these values (max_depth=10, min_samples_split=10, min_samples_leaf=20) for my hyper parameter after analysing their learning and validation curve [7].

C. Metrics

The metrics I used to evaluate my models in the order of importance were 'False Positive Rate(FPR)' > 'Specificity' > 'False Negative Rate(FNR)' > 'Recall' > 'Precision' > 'Accuracy' > 'Confusion Matrix' > 'F1' > 'Matthews Correlation Coefficient(MCC)'

- Accuracy: since its a binary classification problem, finding the accuracy would be a good choice. In general number of correct predictions / total number of predictions [2].
- Recall: I wanted the recall score to be perfect mainly because I did not wanted to miss a single forged bank note to be judged as real bank notes [4].
- Precision: To know the correctness in our predication of positives.
- F1-measure (score)[4].
- Matthews correlation coefficient (MCC) [13].
- Confusion matrix: Confusion Matrix gave me a very good idea on my False Positives (FP) and False Negatives (FN) which helped on choosing a better a model [4].
- Specificity [11]
- False Positive Rate (FPR) [11].

Ratio 80:20	RF	DT	SVM Linear	SVM RBF	Dummy
Accuracy	1.000	0.939	0.990	1.000	0.553
Recall	1.000	0.888	0.989	1.000	0.000
F1	1.000	0.920	0.987	1.000	0.000
Precision	1.000	0.954	0.984	1.000	0.000
FPR	0.036	0.036	0.036	0.000	0.036
FNR	0.000	0.112	0.011	0.000	1.00
MCC	1.000	0.859	0.976	1.000	0.000
Specificity	0.964	0.859	0.964	1.000	0.964

TABLE I
FOR SPLIT TRAIN-TEST RATIO 80:20

Ratio 70:30	RF	DT	SVM Linear	SVM RBF	Dummy
Accuracy	1.000	0.938	0.986	1.000	0.551
Recall	1.000	0.942	0.979	1.000	0.000
F1	1.000	0.923	0.989	1.000	0.000
Precision	1.000	0.904	1.000	1.000	0.000
FPR	0.000	0.086	0.000	0.000	0.000
FNR	0.000	0.058	0.021	0.000	1.000
MCC	1.000	0.855	0.981	1.000	0.000
Specificity	1.000	0.914	1.000	1.000	1.000

TABLE II
FOR SPLIT TRAIN-TEST RATIO 70:30

- False Negative Rate (FNR) [11].

As to why I wanted FPR to be the most important metrics is because I did not want even a single forged bank note to get mixed up in the large validated genuine note. Its better to detect the genuine note as forged and do manual screening on the rejected bank notes. So I gave more importance to FPR and Specificity.

IV. Analysis and Result

A. Scores

To understand how each of my models performed with the above mentioned metrics I scored my models on the these metrics.

The Table I and Table II are of different train-test split ratio 80:20 and 70:30 respectively. Since all three algorithms were performing better than my dummy classifier, it was safe to assume that my data set and the models trained were good enough and better than a random guesser. The scores present in the table are definitely good, many having scores above 90%. In table I and II from 80:20, model SV_RBF and from 80:20 RF and SVM_RBF seems too perfect, this can be a cause of over fitting.

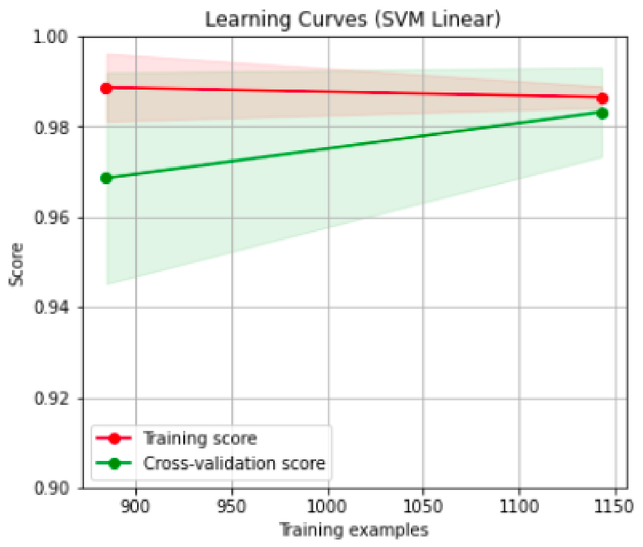
Having different train-test split ratios definitely had an impact in my models, the proof is the table; for example in 80:20 the FPR and Specificity is 0.036 and 0.964 whereas in 70:30 the FPR and Specificity is 0.000 and 1.000. Since my most important metrics were FPR and Specificity was clearly showing that random forest is the victor, I needed to confirm how my model is learning and how different is different from other models.

B. Curve Analysis

Since we can see that 80:20, model SV_RBF and from 80:20 RF and SVM_RBF are over fitting based on the scores obtained, because naturally the model is almost never perfect.

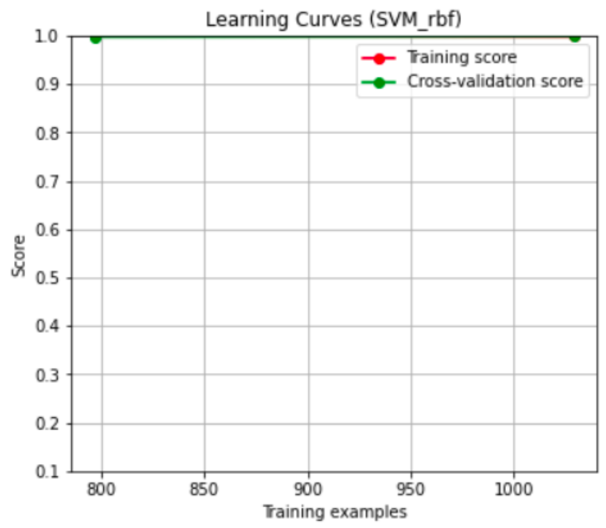
On that I had analyse, AUC_ROC_score, learning and validation curve of every model that I build in this project. The cross validation strategy changes for every model to not cause over fitting and under fitting. The Cross validation used in this project is KFold.

Fig. 1. learning and validation Curve for 80:20



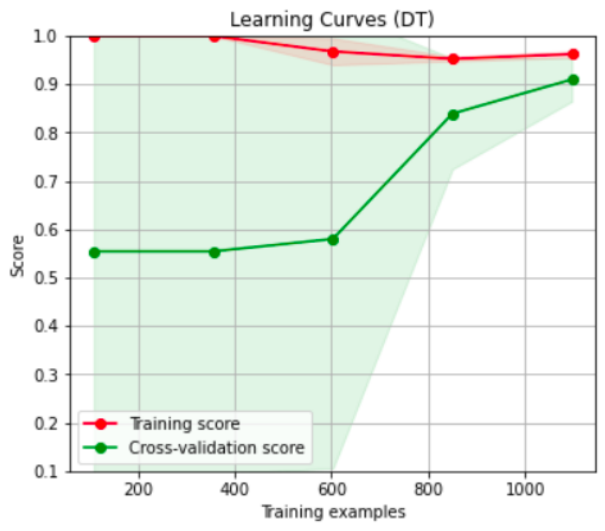
In the graph figure 1, the plotted lines clearly shows that they are converging and bound to meet also their area of their plotted lines are also overlapping. This concludes that the model is as good enough till 1150(or till they meet) training examples, But its not so bad as in figure 2. In figure 2 both our validation and learning curves are 1 from start till end, which is unrealistic and I can definitely say that the SVM_RBF model is not good.

Fig. 2. learning and validation Curve for 80:20



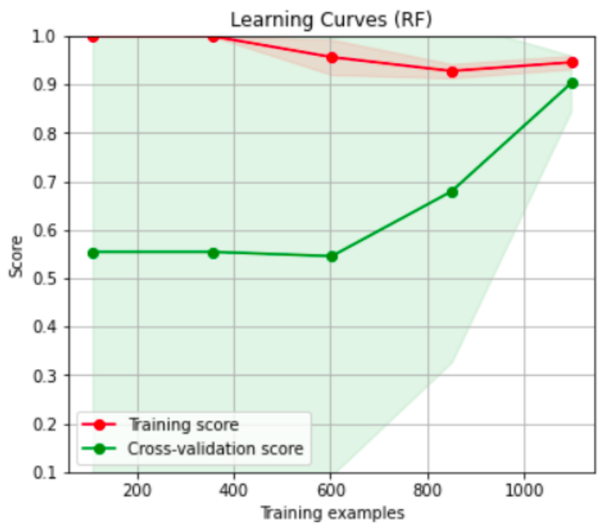
In figure 3 the learning curves and validation curve shows some promise. From 600 'Training examples' (x-axis), I could

Fig. 3. learning and validation Curve for 80:20



see that the complexity of our problem was increasing and that our training score was decreasing which is actually a good thing, it shows that our model is facing difficult problems and is not been able to perform as well as its previous 'Training examples'. As the gap between the curve decreases it creates over fitting hence its good to assume that between 600 and 800 'Training examples' we can consider our model well trained (better than the other two mentioned above).

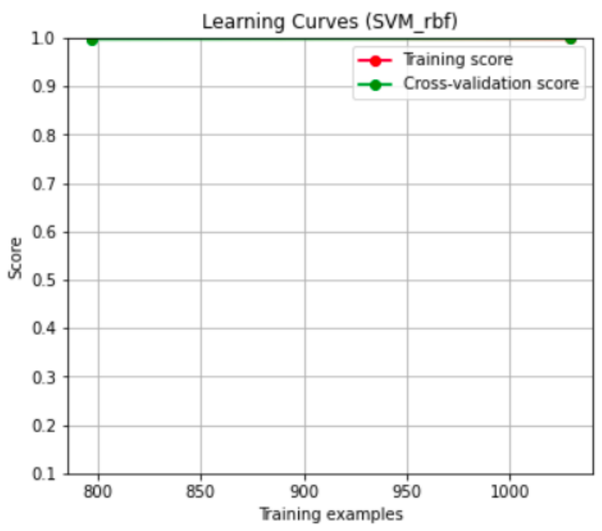
Fig. 4. learning and validation Curve for 80:20



In figure 4 we can see that its almost same as the decision tree graph, after 600 training examples the complexity increases and the training score decreases, which is good and realistic.

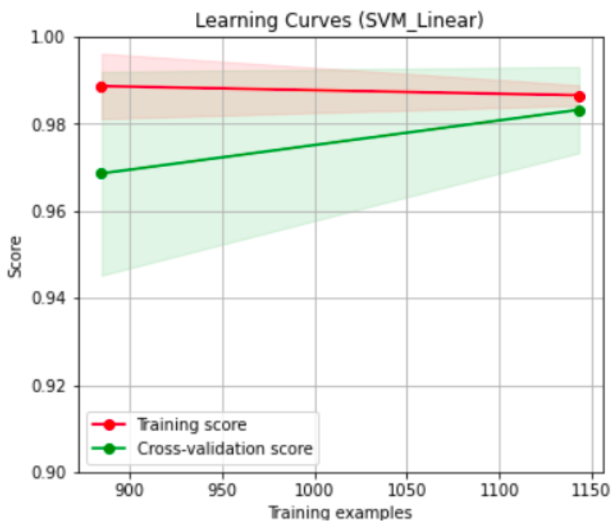
Now for models with train-test split 70:30, we can see the same trend in SVM_RBF figure 5 and figure 2. the conclusion

Fig. 5. learning and validation Curve for 70:30



is same as the SVM_RBF with train-test split 80:20 that its unrealistic and hence the model is to be rejected.

Fig. 6. learning and validation Curve for 70:30



In figure 6, figure 7 and figure 8 are similar to 80:20 train-test split graph figure 1, figure 3 and figure 4 respectively. Hence the conclusion drawn from the listed graphs is Random forest and Decision tree models are best suited for our problem. There is also difference in their score based on train-test split and their RUC_AUC score.

C. AUC_ROC Score

The figure 9 and figure 10 are ROC curves which is meant for model comparison. The graph are very similar to each other but they have slight difference in their scores. i am only considering two models now Random Forest and Decision Tree from both the train test split ratios. AUC_ROC score for 70:30 in RF = 0.987 and DT = 0.988 and for 80:20 in RF = 0.994 and DT = 0.988. On comparing them Random Forest with train test split 80:20 is the better model.

Fig. 7. learning and validation Curve for 70:30

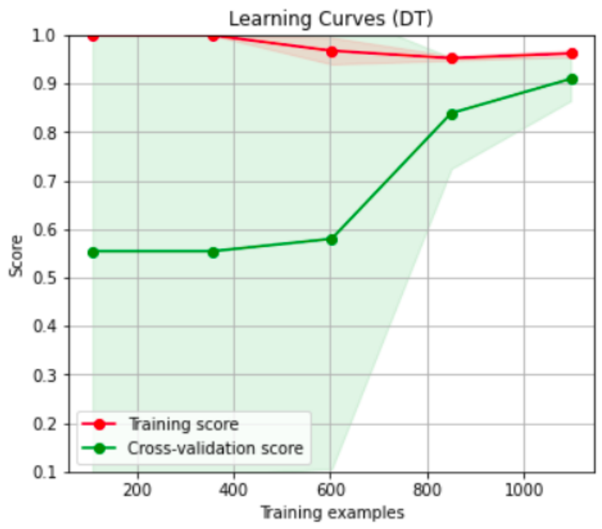
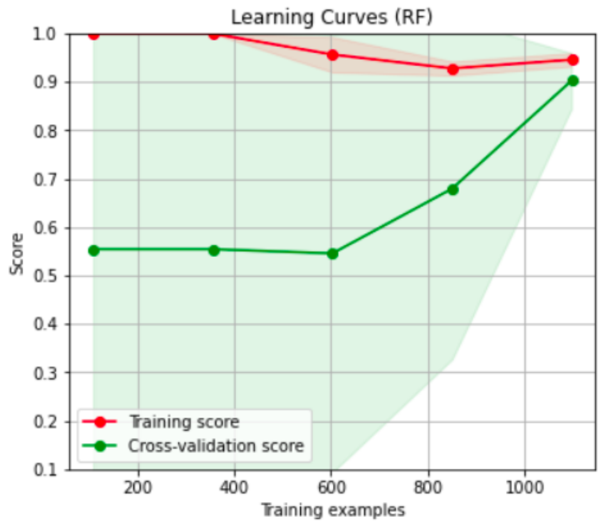


Fig. 8. learning and validation Curve for 70:30



V. Conclusion

The models that were trained gave really high scores in their metrics and better than baseline model. Comparing this to my baseline model [14], that is work done by their done, it was almost similar but different. By comparing my metrics scores and learning and validation curves with AUC_RoC scores. I can say that Random Forest has better edge than Decision Tree with a train-test split of 70:30 ratio and RF and DT are both better than SVM. But I also need more new and complex data to train my models. I did expect my FPR to be zero which was delivered to me by RF in 70:30train-test split ratio which is good whereas others didn't, except SVM. SVm can be completely ruled out in developing of this problem as it over fits.

Fig. 9. learning and validation Curve for 80:20

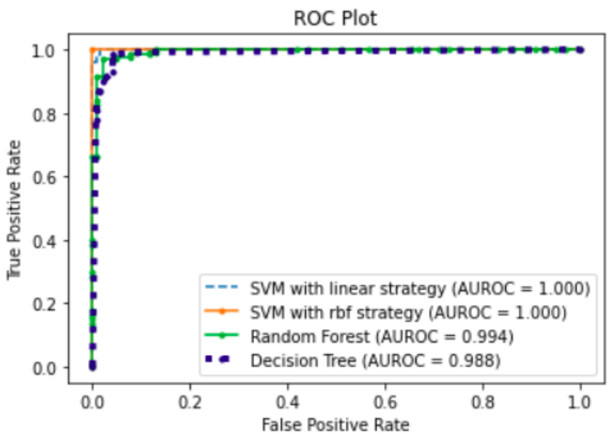
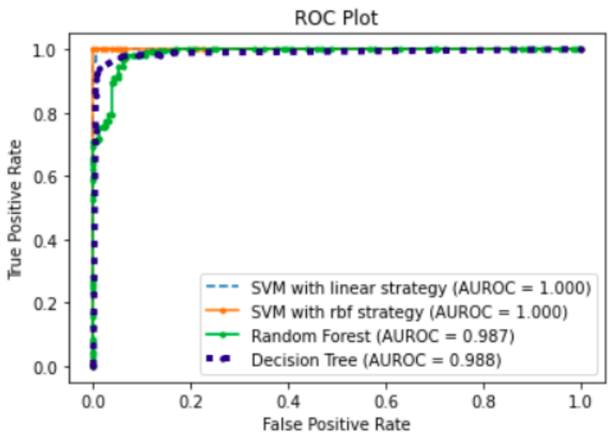


Fig. 10. learning and validation Curve for 70:30



REFERENCES

[1] Anurag Agarwak. *Fake Currency Identification*. URL: https://github.com/dudeanurag/Fake-Currency-Identification/blob/master/banknote_authentication.ipynb.

[2] Google Developer. *Accuracy*. URL: https://developers.google.com/machine-learning/crash-course/classification/accuracy.

[3] Volker Lohweg. *banknote authentication Data Set*. URL: https://archive.ics.uci.edu/ml/datasets/banknote+authentication.

[4] Pre-GA. *Evaluation Method*. URL: https://cloud.google.com/automl-tables/docs/evaluate.

[5] Bosubabu Sambana et al. "AN ARTIFICIAL INTELLIGENCE APPROACH TO FAKE CURRENCY DETECTION SYSTEM". In: *International Journal of Signal Processing, Image Processing and Pattern Recognition* 12.3 (2019), pp. 1–16.

[6] Ankit Sarkar, Robin Verma, and Gaurav Gupta. "Detecting counterfeit currency and identifying its source". In: *IFIP International Conference on Digital Forensics*. Springer. 2013, pp. 367–384.

[7] Scikit. *Decision Tree*. URL: https://scikit-learn.org/stable/modules/tree.html.

[8] Scikit. *Random Forest*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html.

[9] Scikit. *Standard Scaler*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html.

[10] Scikit. *Support Vector Machine*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html.

[11] Open Source Sklearn. *Confusion Matrix*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html.

[12] Open Source Sklearn. *Dummy Estimators*. URL: https://scikit-learn.org/stable/modules/model_evaluation.html.

[13] Open Source Sklearn. *Matthews correlation coefficient*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.matthews_corrcoef.html.

[14] Anju Yadav et al. "Evaluation of Machine Learning Algorithms for the Detection of Fake Bank Currency". In: *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE. 2021, pp. 810–815.