Practicum 3 – Logistic Regression

Content

Note: All codes and additional comments can be found in the iPython notebook submitted in the zip file.

Abstract

The objective of the practicum is to predict the classification of the bank notes as genuine or forged using logistic regression, stochastic gradient descent (SGD) Classifier from Scikit learn library.

The table below summarized accuracy of each model for this practicum.

Model	Accuracy	Precision	Recall
SGD Classifier with optimized hyperparameters	0.99	0.98	0.99
SGD Classifier with optimized hyperparameters and L1 penalty	0.99	0.99	0.99
SGD Classifier with optimized hyperparameters and L2 penalty	0.99	0.98	0.99
SGD Classifier with optimized hyperparameters and L1&L2 penalty	0.99	0.98	0.99
KNN	1.00	1.00	1.00

As shown in the table above, the KNN algorithm achieves a 100% accuracy for the validation set whereas the SGD model achieves an accuracy of 99% (which is respectable but lower than the KNN model). Therefore, based on the result, the KNN model better fits the dataset where it is possible to accurately predict if the bank note is forged by analyzing the neighboring 5 data points of the test date.

Dataset

The dataset is downloaded from [UCI machine repository](). The class labeled as 0 and 1 is deemed to be genuine and forged respectively. There are a total of four (4) input features and a target output as follows.

- Variance of the wavelet transformed image (continuous)
- Skewness of the wavelet transformed image (continuous)
- Kurtosis of the wavelet transformed image (continuous)
- Entropy of image (continuous)
- [**Target output**] Class: 0 for genuine, 1 for forged

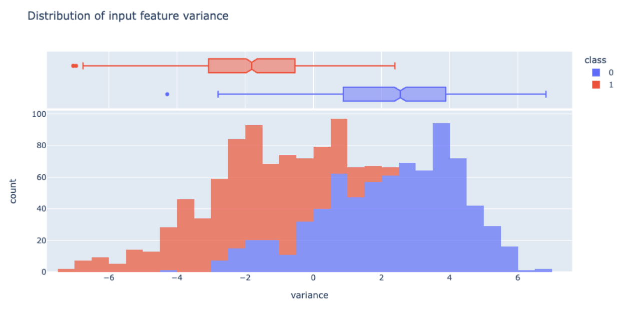
Task 1 – Preprocessing of dataset

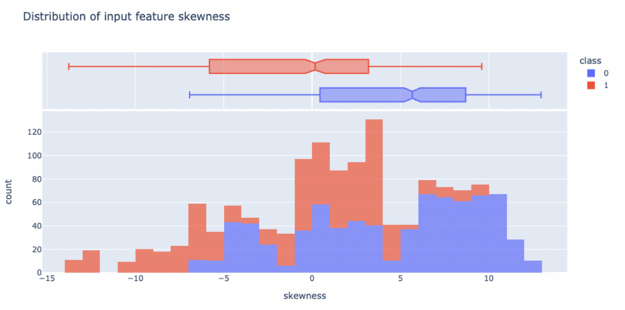
After loading the data and the required libraries in Jupyter notebook, a brief analysis was done. It was noted that there was no missing value, and the data type of the dataset was in order. There was a total of 1,372 records. No data imputation for missing values done for this dataset.

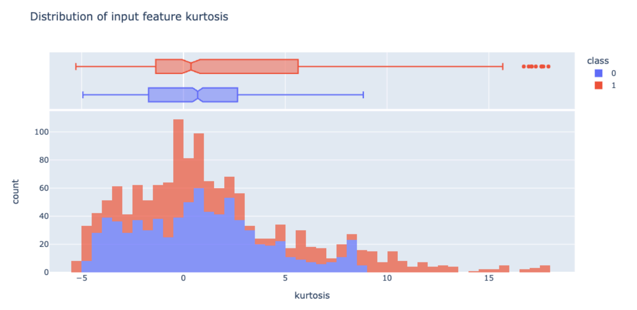
As all input features (variance, skewness, kurtosis, entropy) were numerical, encoding was also not required for the dataset.

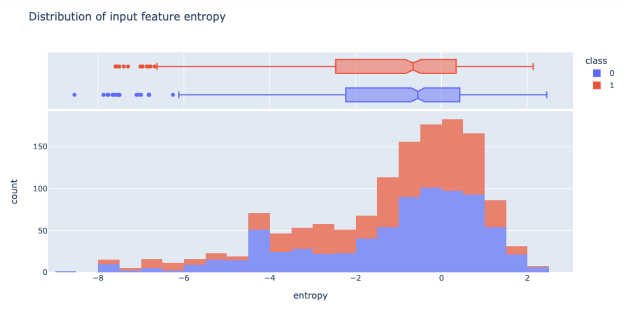
The dataset is deemed rather balanced; hence, sampling technique is also not adopted for the dataset.

Class	**Count**
0	762
1	610





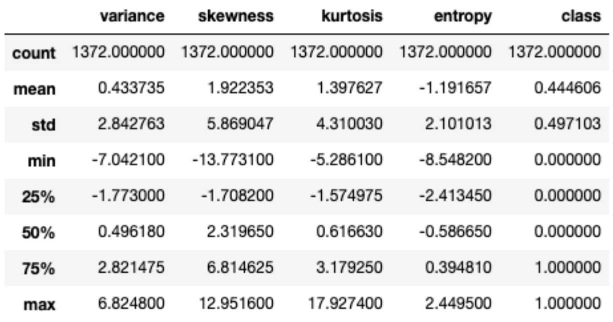




With reference to the distribution plots above, we note the following observations.

1. There were a number of outliers for the following features based on the marginal box plot for variance, kurtosis, and entropy. However, they may contribute in patterns to identify if a bank note is forged. Hence, outliers need not be remove.
2. We note that the variance and skewness of the forged notes have a rather different distribution as compared to the unforged notes in the first two histograms.
3. The forged notes have the long right tail for kurtosis as shown in the third histogram.

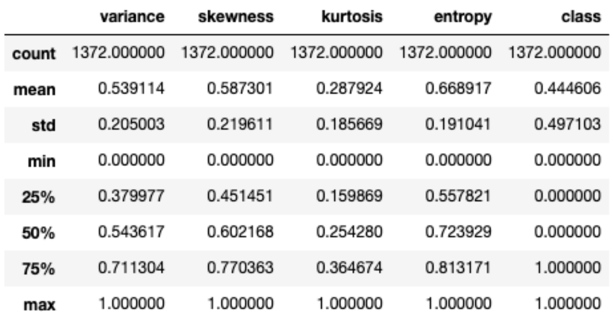
Task 2 – Feature Normalization



	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

Figure 1: Before normalization

As the input variables are of different scales as shown in the summary statistics above, normalization was conducted before model training to ensure equal importance or contribution of the features to the model. It may also help reduces the training time depending on the algorithm used to train the model. For this practicum, we use the MinMaxScaler() from Scikit Learn preprocessing module.



	variance	skewness	kurtosis	entropy	class
count	1372.000000	1372.000000	1372.000000	1372.000000	1372.000000
mean	0.539114	0.587301	0.287924	0.668917	0.444606
std	0.205003	0.219611	0.185669	0.191041	0.497103
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.379977	0.451451	0.159869	0.557821	0.000000
50%	0.543617	0.602168	0.254280	0.723929	0.000000
75%	0.711304	0.770363	0.364674	0.813171	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

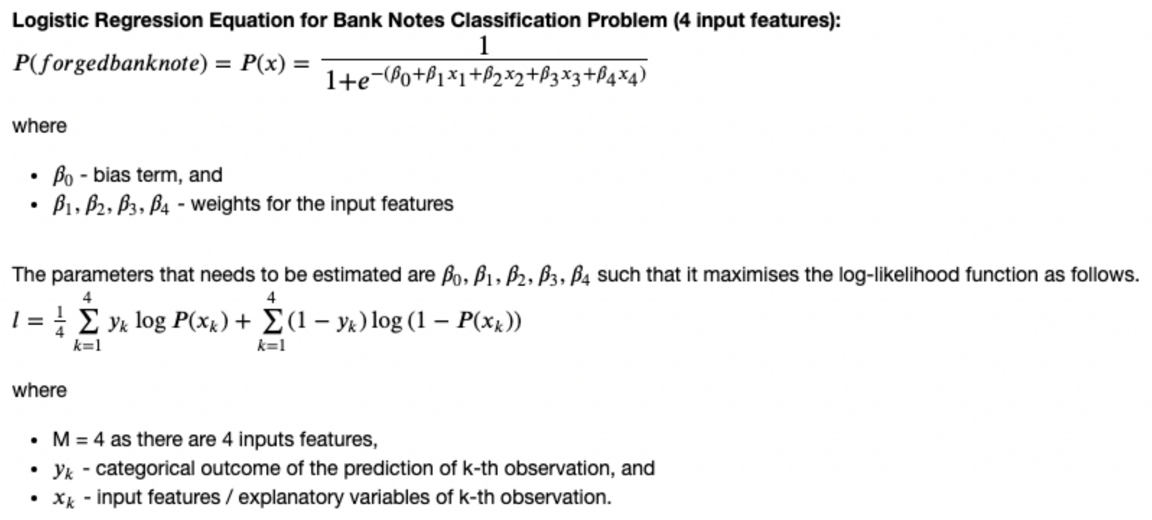
Figure 2: After normalization

Task 3 – Logistic Regression Equation

Logistic regression is a statistical method for predicting binary (or two) classes. The target variable is dichotomous in nature. In this practicum, there are only two possible classes, whether the bank note is genuine or forged.

The logistic regression graph is in the form of a sigmoid function such that the prediction of bank note with a probability 0.5 and above would be classified as forged. While those with probability below 0.5 would be classified as genuine. The threshold probability of 0.5 can be changed depending on the problem.

The logistic equations for the practicum problem can be found in the image below.



Logistic Regression Equation for Bank Notes Classification Problem (4 input features):

$$P(forgedbanknote) = P(x) = \frac{1}{1+e^{-(\beta_0+\beta_1x_1+\beta_2x_2+\beta_3x_3+\beta_4x_4)}}$$

where

- β_0 - bias term, and
- $\beta_1, \beta_2, \beta_3, \beta_4$ - weights for the input features

The parameters that needs to be estimated are $\beta_0, \beta_1, \beta_2, \beta_3, \beta_4$ such that it maximises the log-likelihood function as follows.

$$l = \frac{1}{4}\sum_{k=1}^{4} y_k \log P(x_k) + \sum_{k=1}^{4}(1-y_k)\log(1-P(x_k))$$

where

- M = 4 as there are 4 inputs features,
- y_k - categorical outcome of the prediction of k-th observation, and
- x_k - input features / explanatory variables of k-th observation.

Task 4 – Split dataset into train and test

The train_test_split function from the Scikit learn library model selection module is used to split the dataset into train and test sets with a ratio of 70% and 30% respectively. The random state parameter is used to ensure replicability of the result in subsequent runs.

Task 5 – SGD Classifier

The SGD classifier from Scikit learn library would be imported and used to train the model with the following optimized hyperparameters:

- **Alpha**: Constant which multiplies with the regularization term and used to compute the learning rate when learning rate is set to 'optimal'. The higher the value, the stronger the regularization.
- **Learning_rate**: Value that determine the step of the algorithm takes to optimize the loss function. Learning rate schedule includes the four options: 'constant', 'optimal', 'invscaling' and 'adaptive'.
- **Max_iter**: Maximum number of passes over the training data.
- **Tol**: Stopping criterion for the training. If none is set, then the training will stop when loss is less than (best loss – tol).

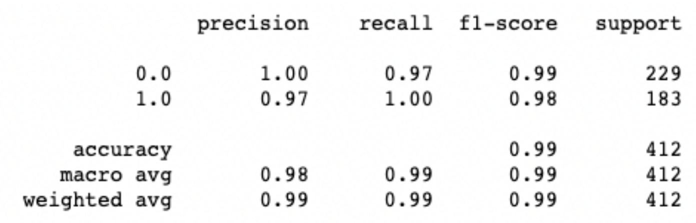
In order for us to find the optimized hyperparameters, a dictionary was created to store each parameter and respective list of values to search through. In this practicum, GridSearchCV algorithm is used to perform an exhaustive search throughout the search space to find the best parameters that returns the highest accuracy score. The search for intentionally performed twice; first, to find the best parameters and second, to lower the range values to search based on the first result to verify if it is possible to get better accuracy score. The finalized values of the hyperparameters are as follows.

Hyperparameter	Value
Alpha	2e-05

Learning_rate	‘optimal’
Max_iter	50
Tol	0.00275

Using the finalized results, we will build and train the model to predict if the bank note is forged.

Task 6 – Classification report of SGD Classifier



```
              precision    recall  f1-score   support

         0.0       1.00      0.97      0.99       229
         1.0       0.97      1.00      0.98       183

    accuracy                           0.99       412
   macro avg       0.98      0.99      0.99       412
weighted avg       0.99      0.99      0.99       412
```

Figure 3: Classification report for SGD Classifier

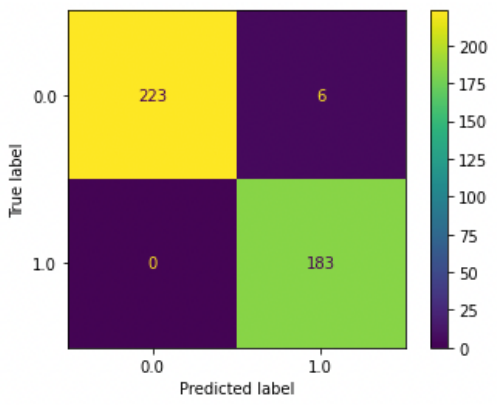


Figure 4: Confusion matrix for SGD Classifier

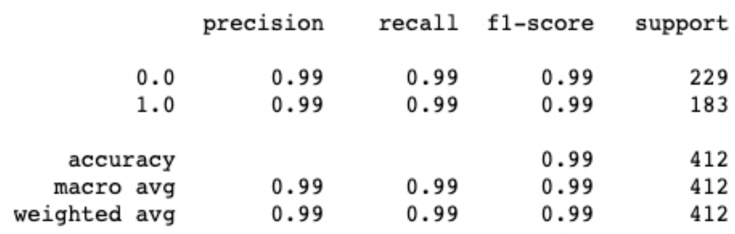
The trained model has a high accuracy score of 99%. There were only 6 misclassified bank notes which were not forged but predicted as forged. Given that the impact of type II error (i.e., false negative) is greater than type I error due to the potential loss of profit and revenue by not identifying the forged bank notes, the impact of misclassification deemed not detrimental.

Task 7 & 8 – Regularization and classification reports

In this section, we will explore regularization and compare the accuracy with the original model.

<u>L1-norm: Lasso Regression</u>

By adding a penalty parameter of l_1 (which represents lasso regression), we can retrain the model with the penalty term. The classification report is as follows.



```
              precision    recall  f1-score   support

         0.0       0.99      0.99      0.99       229
         1.0       0.99      0.99      0.99       183

    accuracy                           0.99       412
   macro avg       0.99      0.99      0.99       412
weighted avg       0.99      0.99      0.99       412
```

Figure 5: Classification of SGD Classifier with L1 penalty

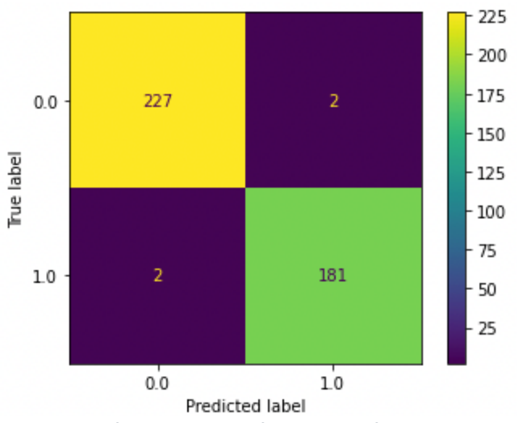
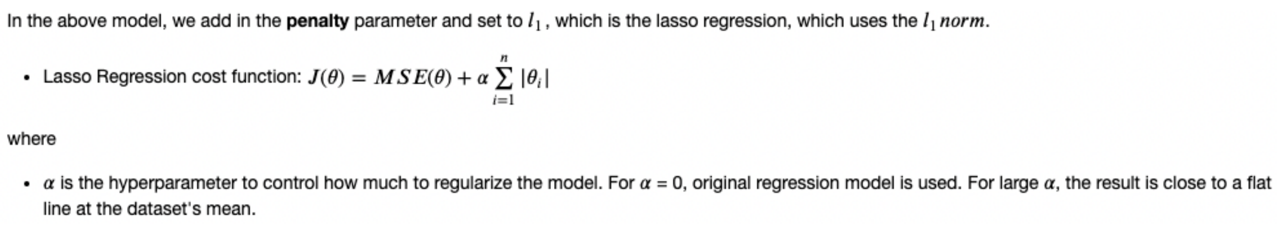


Figure 6: Confusion matric of SGD Classifier with L1 penalty



In the above model, we add in the **penalty** parameter and set to l_1, which is the lasso regression, which uses the $l_1\, norm$.

- Lasso Regression cost function: $J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i|$

where

- α is the hyperparameter to control how much to regularize the model. For $\alpha = 0$, original regression model is used. For large α, the result is close to a flat line at the dataset's mean.

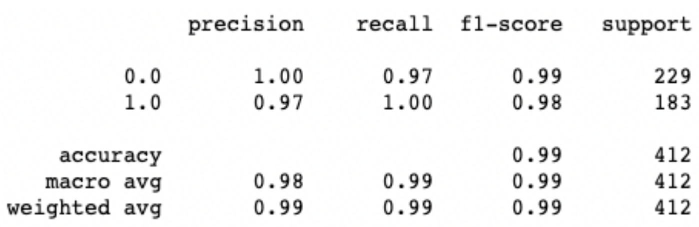
Lasso regression tends to eliminate the weight of insignificant or non-important features by setting the weights of these features to zero. In other words, it performs feature selection automatically and return a sparse model with only the features deemed as important.

Based on the above classification and confusion matrix, we note an improvement in accuracy as the number of misclassification decreases from 6 to 4. There were two forged bank notes predicted as genuine; and two genuine bank notes predicted as forged.

This prediction may not be ideal for this particular problem due to the comparatively more detrimental impact of type II error as highlighted in the above section.

L2-norm: Ridge Regression

By adding a penalty parameter of l_2 (which represents ridge regression), we can retrain the model with the penalty term. The classification report is as follows.



```
              precision    recall  f1-score   support

         0.0       1.00      0.97      0.99       229
         1.0       0.97      1.00      0.98       183

    accuracy                           0.99       412
   macro avg       0.98      0.99      0.99       412
weighted avg       0.99      0.99      0.99       412
```

Figure 7: Classification report for SGD classifier with L2 penalty

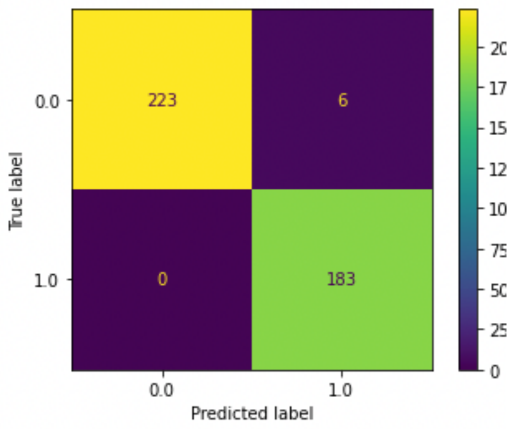


Figure 8: Confusion matric for SGD classifier with L2 penalty

For ridge regression, we add the **penalty** parameter and set it to l_2, which uses the $l_2 \, norm$.

- Ridge regression cost function: $J(\theta) = MSE(\theta) + \alpha(\frac{1}{2}\sum_{i=1}^{n}\theta_i^2)$

where

- α is the hyperparameter to control how much to regularize the model. For α = 0, original regression model is used. For large α, the result is close to a flat line at the dataset's mean.

Ridge regression restricts the weights of the features that are not important by assigning values that are close the 0. All features will still be used in the final model. Feature selection is not conducted automatically for ridge regression.

Based on the above classification report and confusion matrix, there is no difference from the original SGD classifier model in task 5 and 6.

L1-norm & L2-norm: Elastic Net

By adding a penalty parameter of ‘elasticnet’ (which represents elastic net regularization), we can retrain the model with the penalty term. Elastic net is a combination of lasso and ridge regression. The classification report is as follows.

```
              precision    recall  f1-score   support

         0.0       1.00      0.97      0.99       229
         1.0       0.97      1.00      0.98       183

    accuracy                           0.99       412
   macro avg       0.98      0.99      0.99       412
weighted avg       0.99      0.99      0.99       412
```

Figure 9: Classification report for SGD classifier with L1 and L2 penalty

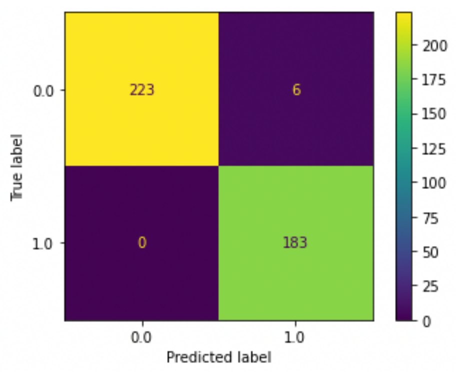


Figure 10: Confusion matrix for SGD classifier with L1 and L2 penalty

For elastic net, we add the **penalty** parameter and set it to `elasticnet`, which uses both $l_1\, norm$ and $l_2\, norm$.

- Elastic net cost function: $J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} |\theta_i| + \alpha(\frac{1}{2} \sum_{i=1}^{n} \theta_i^2)$

Based on the classification report and confusion matrix above, we note that there is not difference from the original SGD classifier model in task 5 and 6.

Task 9 – K-Nearest Neighbor (KNN)

In this section, we build a KNN model with the default hyperparameters. The classification report is as follows.

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       229
         1.0       1.00      1.00      1.00       183

    accuracy                           1.00       412
   macro avg       1.00      1.00      1.00       412
weighted avg       1.00      1.00      1.00       412
```
Figure 11: Classification report for KNN

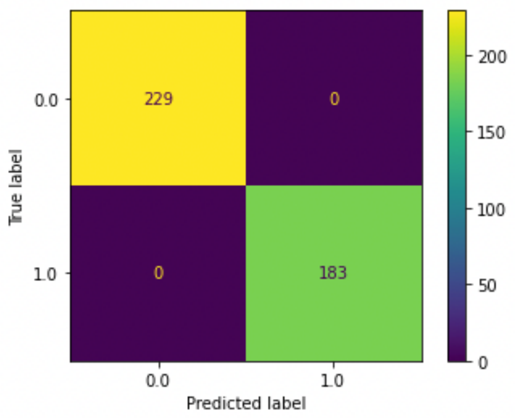


Figure 12: Confusion matrix for KNN

As shown in the above classification report, KNN model achieves a 100% accuracy for the validation whereas SGD model has an accuracy of 99%. We can infer that the KNN model fits the dataset better than SGD. It is possible to accurately predict whether the bank note is genuine or forged by analyzing the 5 nearest neighbors and take the most common class of the 5. However, the accuracy for SGD is considered respectable as well.

References

1. Wikipedia: Logistic Regression - https://en.wikipedia.org/wiki/Logistic_regression
2. Machine Learning Mastery: Hyperparameter optimization - https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/
3. Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011 - https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html