CITS5508 Machine Learning

Franco Meng 23370209

Outline:

This report is aimed at building a binary classification algorithm by using Logistic regression and KNN. The dataset was from Fashion-MNIST, where containing 60,000 examples in training set and 10,000 examples in test set. Each example is a 28 x 28 single channel grayscale images, associated with label from one of ten classes.

It is required to extract only 2 labels for this binary classification task. 'Sandal' and 'Sneaker', where 'Sandal' is set to be positive class (Y=1).

Deliverables:

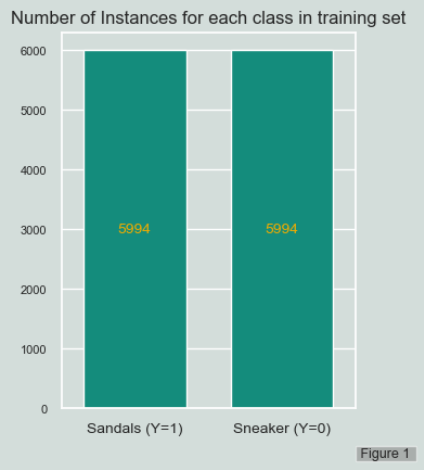
D1: Summarising the dataset:

- Answer:

	Total
The number of instances in Training:	11988
The number of instances in Test :	2000
The total number of instances :	13988

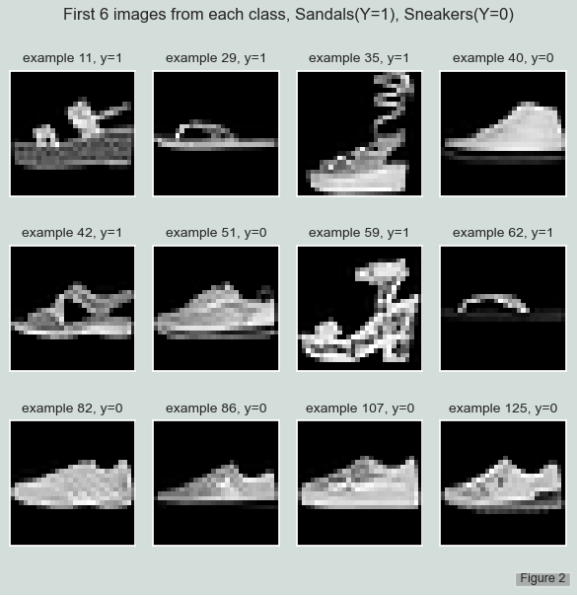
D2: Provide a bar plot showing the number of instances for each class.

- Answer: As figure 1 shows, we have a perfectly balanced training set.



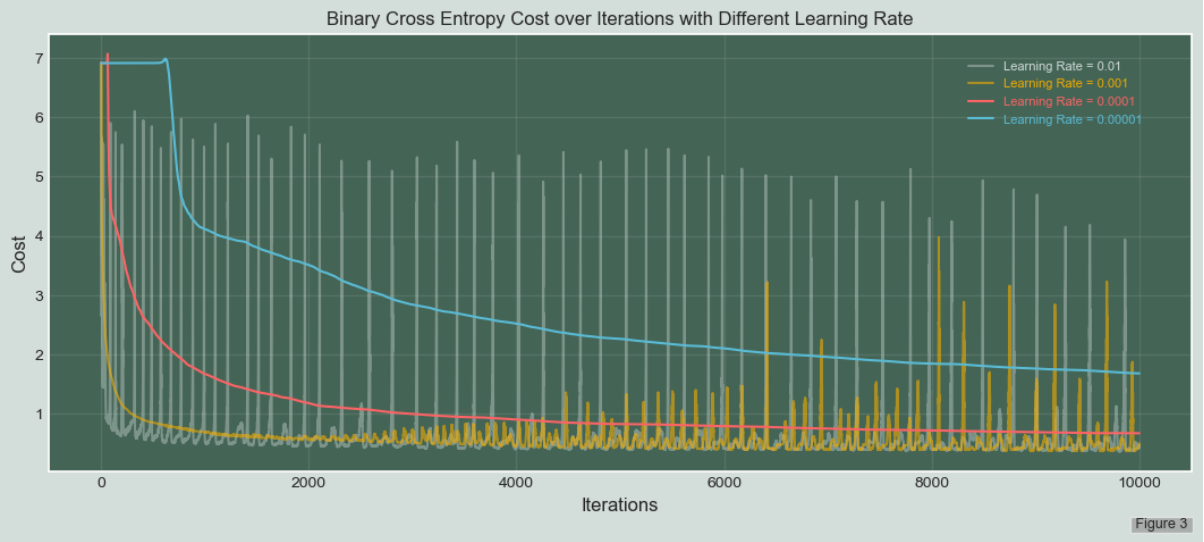
D3: Plot the first six images from each class with the corresponding example id and label listed.

- Answer : Figure 2.



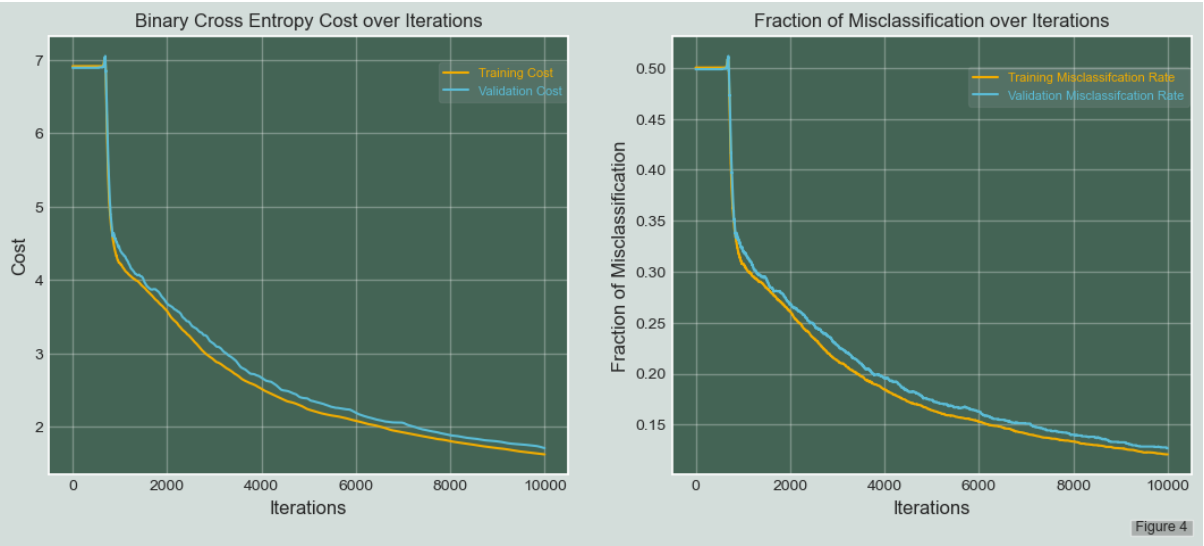
D4: Fitting my own implementation of logistic regression with neg log loss / binary cross entropy as cost function. Experience with different learning rate (η) in gradient descent.

- Answer: As shown in Figure 3, different learning rate affect the gradient descent distinctively.
  With a high η = 0.01, the cost is jumping around and not able to achieve global minimum, where the low η = 0.00001 showing a very slow reduction, and may need more than 10,000 iterations.
  The η=0.00001 is acceptable, showing a very fast convergence but if the iteration is more than 2,000, it will start to bounce.
  With η=0.0001 to be the most stable learning rate in 10,000 iterations.



D5. Set η = 0.00001, providing side by side plot, showing the cost (y-axis) for each iteration (x-axis) for the training and validation set. The right plot showing the fraction of misclassification (y-axis) for each iteration (x-axis).

- Answer : Figure 4,

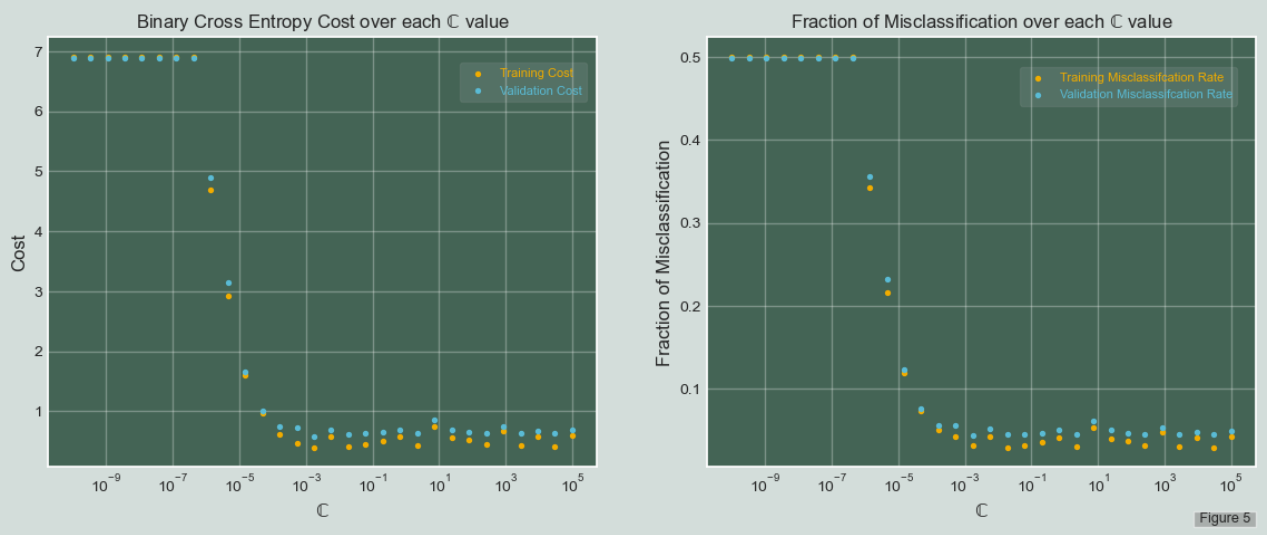


D6. Interpretation of Figure 4.

- Answer: As shown in figure 4, both the costs of training and validation are showing decreases along with the increases of the iterations. However clearly the cost on the validation set start to be constantly higher than the training cost, from around 1000 iteration mark. Meaning the model perform better on the training set but not as good as on the validation set. The model may have overfitting issue with high variance. If we increase the bias, the model may generalize better.

D7. Incorporate a L2 regularisation in the mode.

- Answer : Figure 5. Provide side by side plots showing the value of cost function for each C value, and the fraction of misclassifications for each C value.
  **Here we define: C_range = np.logspace (-10, 5, 30)*



- Discussion:
  In order to be comparable with the C hyperparameters from Sklearn packages for all the following deliverables.
  Here C is here defined as *"Inverse of regularization strength"*. SKlearn
  Therefore the implementation is C here in my own model.
  As we can see from the plot above, the small C has imposed a very strong regularization on the model, where reduced all the weights to nearly 0 (not equal to 0 like lasso regression). Therefore leading to the high loss, and the 50% misclassification rate as the model is predicting every instance to Y=0, sneakers.
  The optimal C may be between 10^{-5} and 10^{-3}. Where training set and validation set cost are closer to each other, but when penalization become very minimum along with the increase of C, the validation cost tend to be constantly higher.

- Extra discussion (not required in deliverables)

  In the usual l2 norm regression mathematic equation, the α, or sometimes λ, is a constant that multiply the regularization term, the higher the value, the stronger the regularization.

  - Cost function with L2 norm = cost + $\alpha \sum_{i=1}^{n} \theta_i^2$
    (Geron, 2019)

  But the C defined in SKLearn documentation. The hyperparameter has been added onto the log loss cost function. Hence the inverse effect.

  - As an optimization problem, binary class logistic regression with regularization term $r(w)$ minimizes the following cost function:

    (1)

    $$\min_w C \sum_{i=1}^{n} s_i \left(-y_i \log(\hat{p}(X_i)) - (1-y_i)\log(1-\hat{p}(X_i))\right) + r(w),$$

    where s_i corresponds to the weights assigned by the user to a specific training sample (the vector s is formed by element-wise multiplication of the class weights and sample weights).

    We currently provide four choices for the regularization term $r(w)$ via the `penalty` argument:

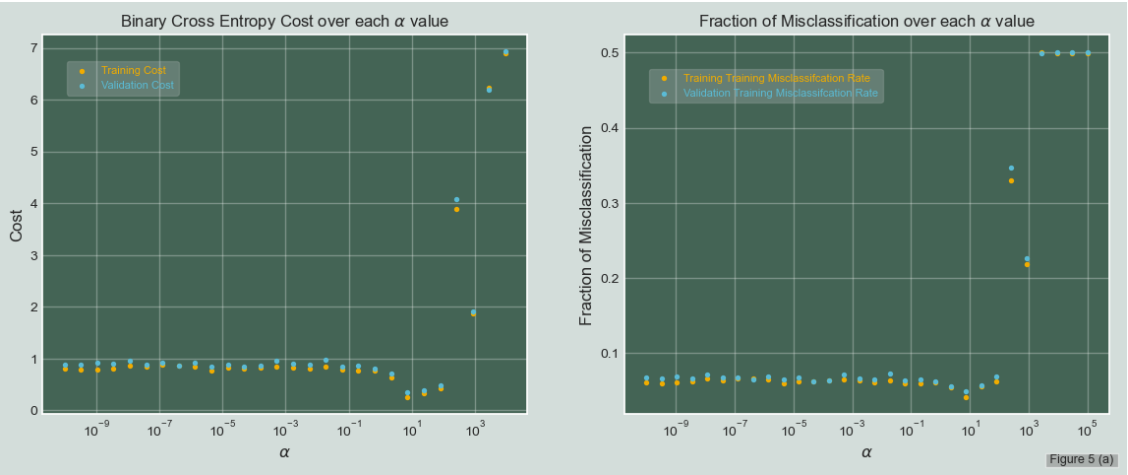
penalty	$r(w)$
None	0
ℓ_1	$\|w\|_1$
ℓ_2	$\frac{1}{2}\|w\|_2^2 = \frac{1}{2}w^T w$
ElasticNet	$\frac{1-\rho}{2}w^T w + \rho\|w\|_1$

    (scikit-learn, 2007-2024)

  Therefore in order to complete task *"Your task is to select an optimal value for the regularisation hyperparameter C (the hyperparameter α in the textbook) "*. Here the relationship between α and C to be :

C * α * m (number of instances in the training set) = 1

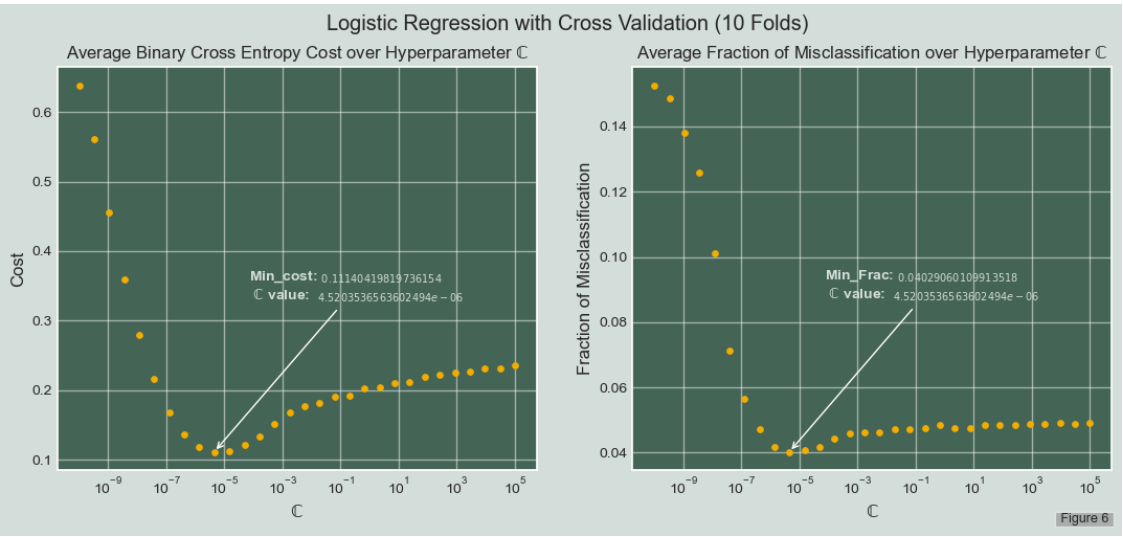
I have further provided a following figure 5 (a), where showing the effect of α implementation for my logistic regression.



As we can see the large alpha leads to heavy penalization, same effects with the small C as above.

D8. Incorporate a L2 regularisation with Cross Validation, by using sklearn.linear_model.LogisticRegressionCV

- Answer: Figure 6.



D9: Conclusion:

- Answer: Comparing between Figure 6 (with cross-validation) and Figure 5 (without cross-validation) , The optimal C value shifted to a smaller value, I would choose the C value after the cross validation as indicated above in the figure 6. The cost and misclassification rate are slightly improved. The average of all 10 folds of cross-validations on each C, would yield a better result. For the fixed validation set, the C may only be optimal for that particular training set - validation split, may not be optimised for generalization.

D10: Using SGDClassfier together with GridSearchCV to find the optimal regularisation parameter and learning rate.

- According to grid search:
  Again here, SGDClassfier using α not C as hyperparameters, so conversion has been conducted and recorded in the Jupyternote book clearly.

The optimal value of α (from SGDClassier with GridSearchCV):	18.453589348268654
Conversion into hyperparameter C:	*4.520353656360249e-06* *(same result with annotated in fig.6 when using logisticRegCV)*
The value of cost function in the training set for this optimal value:	0.09656777383905067
The value of cost function in the test set for this optimal value:	0.10285665156767708
The fraction of misclassifications in the training set:	0.034932221063607924
The fraction of misclassifications in the validation set:	0.03544620517097581

D11: Give a short interpretation of above result comparing with obtained in task D8.

- Answer: Comparing with D8 using the logisticRegressionCV, the grid search combining with SGDClassfier has returned an exact same optimal C value, in a form of α before conversion, It also returned an optimal learning rate.

- Both the cost and misclassification fraction on validation set have improved slightly in Gridsearch CV. The grid search has yield the best performance on the validation set in all the previous attempts.

D12: Train the model with optimal C, plot precision vs recall.

- Answer

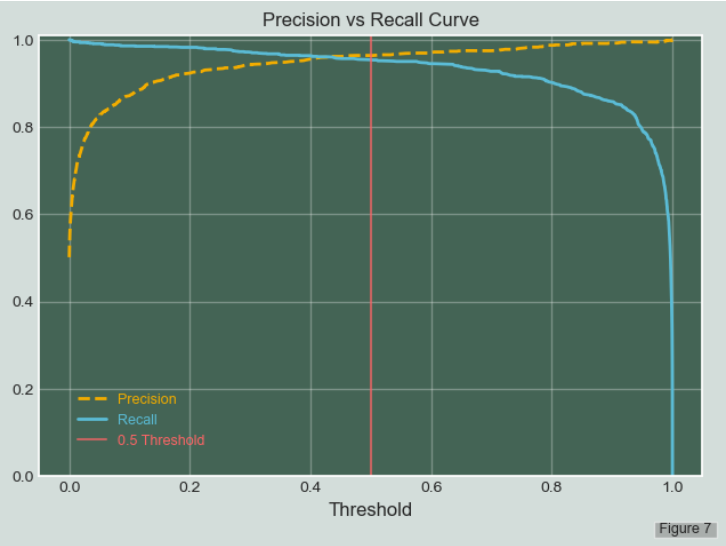


Figure 7

- Comment:
  - As we can see, when threshold is 0, the model predicts all the instances as Sandals (Class 1). Therefore, the recall is 1, It has predicted all the true Sandals as Sandales, where precision is 50%, as our relatively balanced dataset, all the Sneakers got predicted incorrectly. Similarly, when threshold is approaching to 1, the model predict has no false positive prediction, hence precision is 1, but false negative would be very high.
  - The red line indicates a 0.5 standard threshold, the figure shows if we choose 0.5, we could have higher precision than recall, meaning more false negative, less false positive, If model return the predicted class as 1 Sandales, less likely it is incorrect. But if model predict class as 0 Sneakers, more likely it should be Sandales instead.
  - I would choose a threshold between 0.4 and 0.5 to reach a balance of predicting both class correctly, no bias to Sandals or Sneakers.

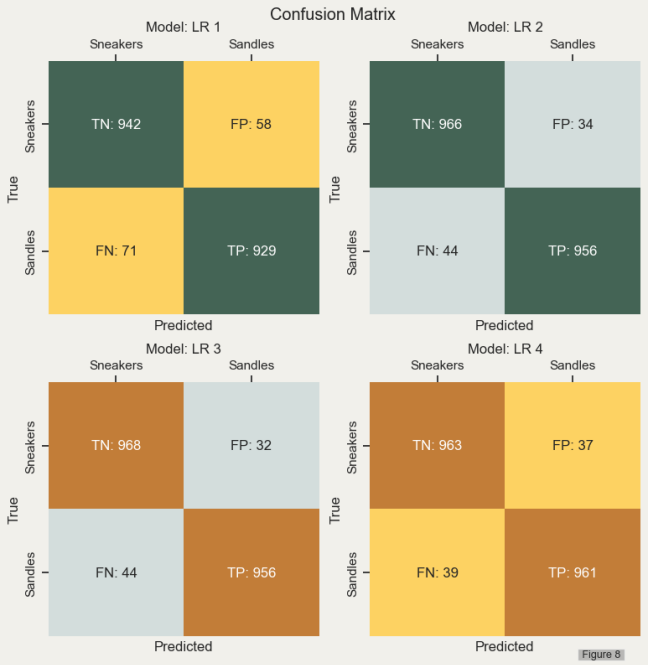
D13: Find and report the optimal threshold, comment comparing with reflection in D12.

- Answer:
  Optimal threshold : 0.42779265
  Fscore: 0.9608985

  Comparing with D12, this optimal threshold falling between 0.4 and 0.5, which agrees my previous choice, however in the real life scenario, the optimal threshold may be determined case by case, on the different task objectives and the main goal to achieve. (e.g. Spam email classification vs Cancer detection)

D14: Analysing the performance closer, provide model performance on the test set for LR1, LR2, LR3, LR4.

- Answer: Figure 8.



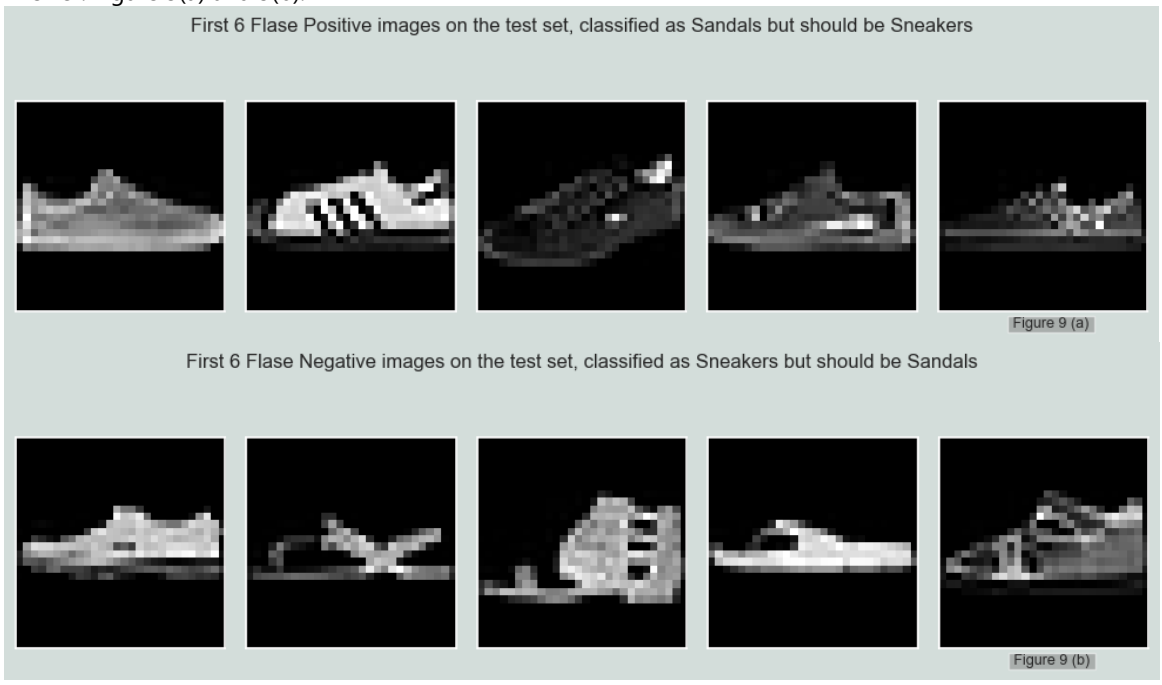
	Precision	Recall	False Positive Rate
LR 1	0.9412	0.929	0.071
LR 2	0.9657	0.956	0.044
LR 3	0.9676	0.956	0.044
LR 4	0.9629	0.961	0.039

D15. Conclusion:¶

- Answer: From above table and figure 8 Confusion matrix, we can see the LR1 model without penalisation generalize the worst on the testing set.
  Both LR2, and LR3 are quite similar with L2 norm penalisation implemented, in order to improved the model and distinctively reduced FP and FN.
- The LR4 has more balanced result. based on the best F1 score to pick the optimal threshold.

D16. Consider LR4 and show 5 images from FP and FN on the test set.

- Answer. Figure 9(a) and 9(b).

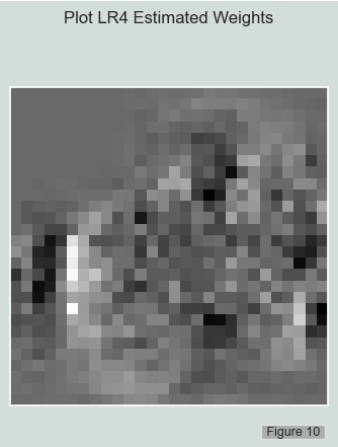


D17. Comment:

- Answer: The first 6 False Positive images show quite uniformed pixels, where model may not able to distinguish the class. The pixels have very low contract, however the Adidas sneakers maybe falsely classified as sandal straps.
- The second 6 False Negative images shows quite clustered pixels, where may more looks like sneakers rather than sandals. Even with human eyes it is a bit hard to tell a definite answer.

D18. Plot the LR4 estimated weights:

- Answer. Figure 10.

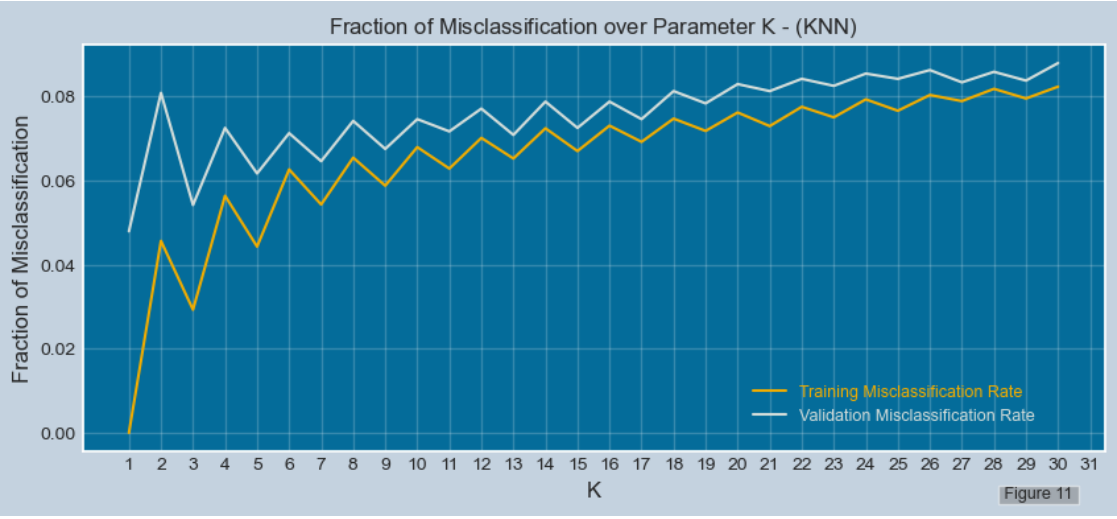


D19. Comment:

- Answer: The model LR4 has stronger weights around the front of the shoes, possibly if the shoes with enclosed front it may be a good indication to be sneakers, rather than sandals where it is more likely to be open at front.
  The model tends to detect the edge of the shape, rather than inner side.

D20. K-NN, plot showing the fraction of misclassification for each K.

- Answer: Figure 11.



D21. Comment:

- Answer: with the increase of the K, the model misclassification rate increases accordingly.
  It is really interesting to see the model perform better on the K being an odd number, compared to even number K.

  It is usually not recommended the choose K = 1 due to the high likelihood being overfitting, but based on this training and validation set, K=1 perform the best on the validation set. Comparing with the logistic regression in section 4.2, KNN perform also quite will with the mis classification rate being around 0.05. like the logistic regression.

D22. Providing KNN on test set for the K value decided in task D21.

- Answer: Figure 12.

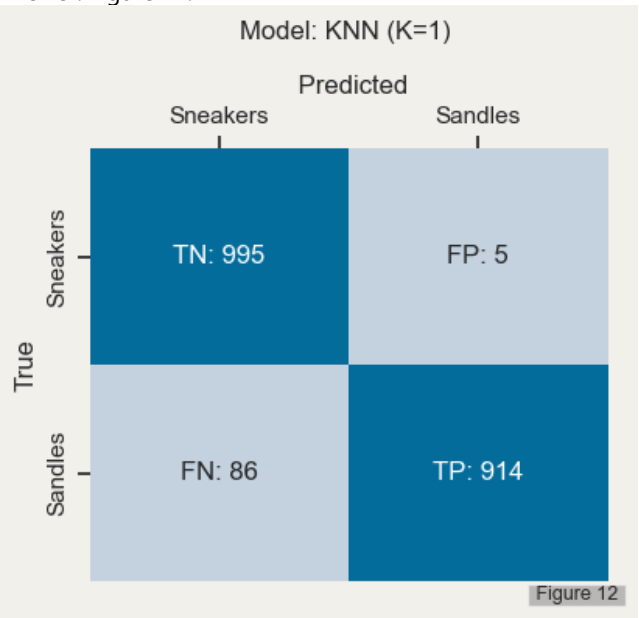


Figure 12

	Precision	Recall	False Positive Rate
KNN	0.9946	0.914	0.086

D23. Compare all models.

- Answer: As we can see from below

	Precision	Recall	False Positive Rate
LR 1	0.9412	0.929	0.071
LR 2	0.9657	0.956	0.044
LR 3	0.9676	0.956	0.044
LR 4	0.9629	0.961	0.039
KNN	0.9946	0.914	0.086

If our goal is precision score, which is, detect the true Sandals correctly but don't mind other classes being incorrectly classified as Sandals, then KNN is a great model for this purpose, as it is reaching highest 0.9946 accuracy score.

However if we wish to have no bias on detect different classes, where all classes should be detected correctly, then LR 4, logistic regression with optimal penalization and optimal threshold for Fscore, will lead us to the least misclassification rate.

D24. Improvements on the ML pipeline.

- Answer:
  Here in this task, we've got given all the dataset, with explicit instruction to train and model, and interpreting the result. There are a few things we could do to further improve the model.

  - Feature scaling.
    This a quite a crucial step for many ML algorithms, especially the distance based algorithms like KNN, as well as the penalization models, where a constant value has been multiplied to the weight vector.
    If there are vast differences in the scales in different feature, eg. Age vs population . Then these scale differences will influence the model differently, where larger scale feature may have more influence.

    However in our dataset, all features are representations of the grey scale of a pixel, which within the same range. Then it is still acceptable to perform model algorithms without feature scaling. Plus if all feature scaled to -1 to 1, the deliverable of D4 with learning rate to be 10^{-5} may be too slow for the model. And the subsequential plot may not be meaningful.
    After some experiment, if the features are scaled, the model will perform a bit better, but not vastly different.

  - Feature reduction/selection
    This is another important step to build a successful model. Maybe the 'white border pixel' outside every image are all 0 in our dataset, these may be not so useful and provide meaning information, can therefore been disregard.

Commonly for image data, it is also recommended to use 'max pooling' etc techniques, transform and reduce the number of features for the model. Or even increasing the size of the dataset through rotation/flipping, may also help. The edge/outline of an image may be quite important for classification/categorization.

- Balanced data
  This is also important for classification, as imbalanced data may cause some assumption on a good model but in fact is not. However, in our case, we have a perfectly balanced dataset.

- Lots of other skills including cross validation, hyper parameter tuning etc have been applied in our case.

Bibliography

Geron, A. (2019). *Hands On Machine Learning with Scikit Learn Keras and Tensorflow.* Sebastopol: OReilly Media.

scikit-learn, d. (2007-2024). *Sklearn*. Retrieved from Sklearn API: https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression