CSC2062 AIMA – Assignment 2

[Riain Walsh]

[03/05/2025]

Introduction

In this report, I will examine and assess the effectiveness of various machine learning models developed to classify images into specific categories. The image datasets used to train and test the models contain multiple classes, including letters, happy faces, sad faces, and exclamation marks. These models will generate predictions using features extracted from the file 40415373_features.csv. Several machine learning techniques will be evaluated, including logistic regression, k-nearest neighbours (with and without cross-validation), and random forest classification.

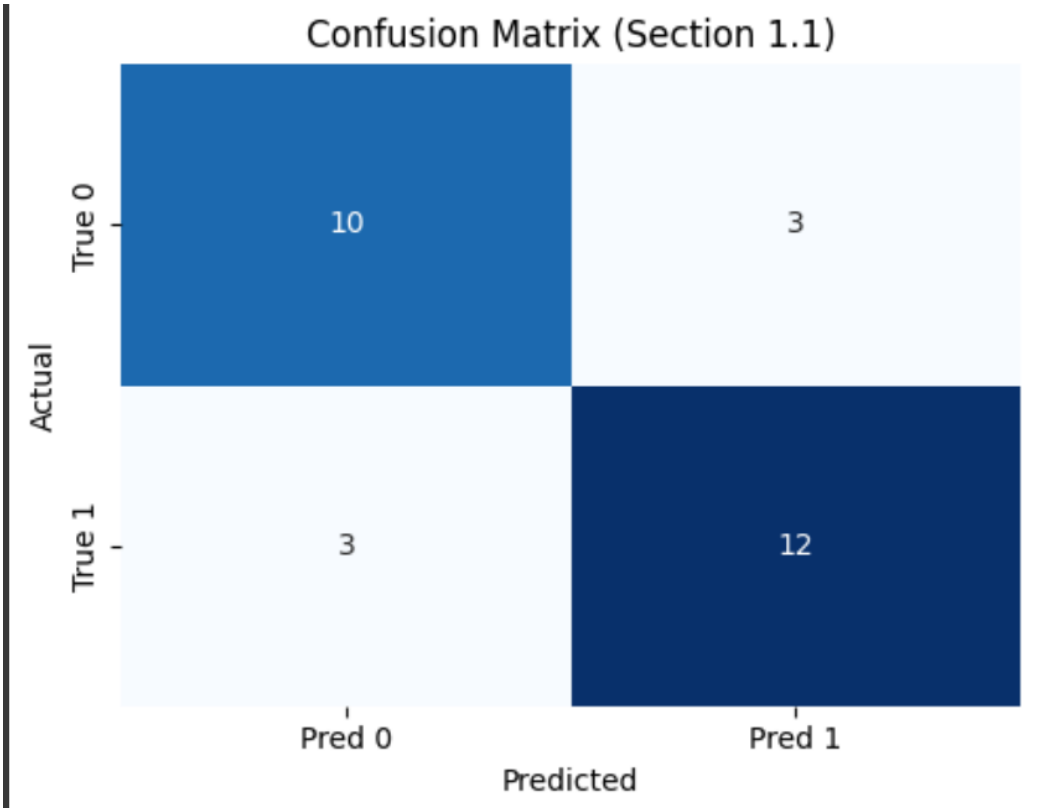
The goal of this report is to explore how accurately these models can classify previously unseen images and how different validation techniques influence the reliability and performance of each model.

Section 1

In this section, I used the feature file *40415474_features.csv* to build a logistic regression model with a separate training and test set, as well as a second model using 5-fold cross-validation. Both models were designed to predict the probability that an image belongs to the "letter" category.

Section 1.1

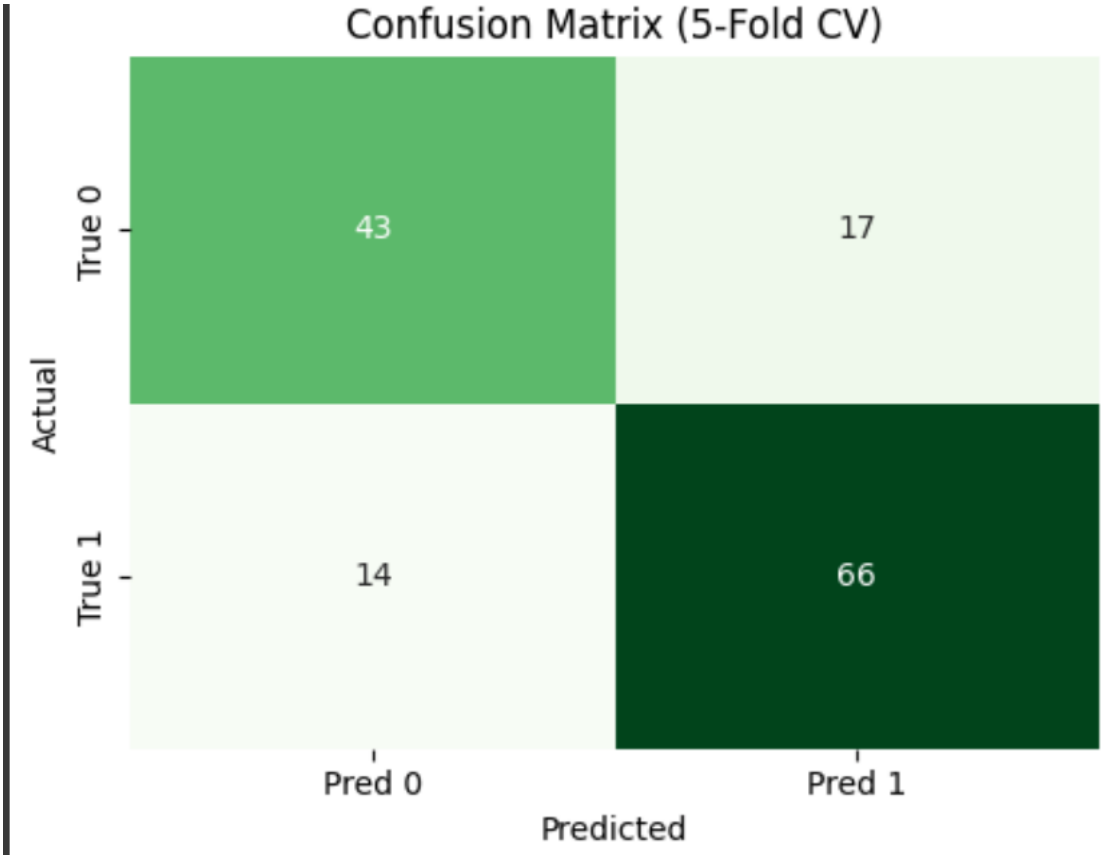
The model achieved an accuracy of 78.57%, with a true positive rate (recall) and precision both at 80.00%, resulting in an F1 score of 80.00%. This indicates the model performs reasonably well in identifying images that are letters. The false positive rate was 23.08%, suggesting that while the model correctly identified most letters, it also misclassified some non-letter images as letters. Overall, the balance between precision and recall reflects a generally reliable performance in predicting letter categories, though there is room for improvement in reducing false positives.




```
=== Section 1.1 Metrics ===
Accuracy: 78.57%
Precision: 80.00%
Recall (TPR): 80.00%
False Positive Rate: 23.08%
F1-score: 80.00%
```

Section 1.2

I created another logistic regression model to predict the probability of an image belonging to the "letter" category, following the same approach as in Section 1.1. However, this time I applied 5-fold cross-validation to evaluate the model's performance. The results are shown below.




```
=== Section 1.2 Metrics (5-Fold CV) ===
Accuracy: 77.86%
Precision: 79.52%
Recall (TPR): 82.50%
False Positive Rate: 28.33%
F1-score: 80.98%
```

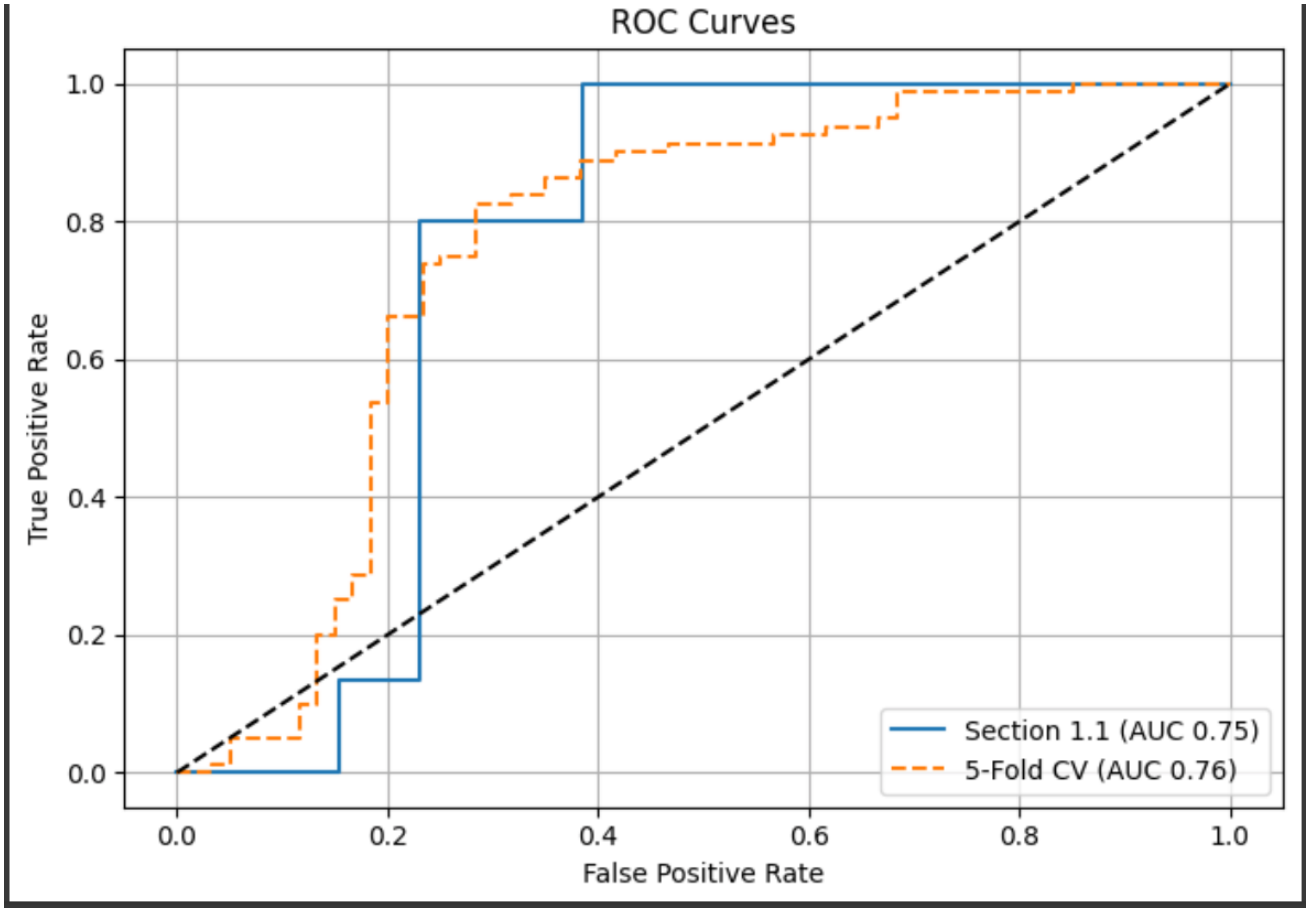
This model achieved an accuracy of 77.86%, a true positive rate (recall) of 82.50%, and a precision of 79.52%, resulting in an F1 score of 80.98%. The false positive rate was 28.33%. These results indicate that the model remains effective at identifying letter images, but with slightly lower accuracy and a higher false positive rate than the model in Section 1.1. While it correctly captures more true positives (as seen in the higher recall), the increase in false positives suggests a decrease in specificity.

The drop in overall accuracy and rise in false positives can be attributed to the nature of cross-validation, where each image is used as a test case once across five folds. In contrast, the model in Section 1.1 was only tested on a single random 20% subset, which may have been easier to classify. Cross-validation exposes the model to a wider range of examples, including more difficult cases, leading to a more realistic and reliable estimate of its generalisation ability on unseen data.

Section 1.3

The ROC curves above compare the logistic regression models from Section 1.1 and Section 1.2 (which used 5-fold cross-validation). Both models demonstrate good classification performance, as shown by their areas under the curve (AUC), which are 0.75 for the Section 1.1 model and 0.76 for the cross-validated model. This indicates that both models are effective at distinguishing between letters and non-letters, with the cross-validated model performing slightly better overall.

The optimal decision thresholds are also highlighted on the ROC curves as the points farthest from the diagonal dashed line, representing the best trade-off between the true positive rate and false positive rate. For both models, the ideal threshold achieves a true positive rate of around 0.8. However, the Section 1.1 model reaches this with a lower false positive rate, suggesting that it may be more conservative and precise in its predictions. In contrast, the cross-validated model accepts a slightly higher false positive rate to achieve similar recall, reflecting a different balance between sensitivity and specificity.



Section 2

In this section, I carried out a 4-class classification task using k-nearest neighbour classification to distinguish between the letters 'a' and 'j', as well as happy faces, sad faces, and exclamation marks.

Section 2.1

I applied k-nearest neighbour (KNN) classification to distinguish among five classes—'a', 'j', happy faces, sad faces, and exclamation marks—using all odd k values from 1 to 13. Four features served as predictors:

- **nr_pix:** Different symbols vary in size; for instance, 'j' and '!' typically use fewer black pixels than 'a' or the faces.

- **aspect_ratio:** Tall, narrow symbols like ‘j’ and ‘!’ have low aspect ratios, whereas rounder shapes (‘a’ and faces) yield higher ratios.
- **neigh_1:** This counts pixels with exactly one black neighbour—‘a’ generally has a single endpoint, while faces and other symbols exhibit multiple endpoints.
- **no_neigh_right:** Letters and exclamation marks tend to have more whitespace to the right of each black pixel, so they register more pixels without a right-hand neighbour than the faces do.

```
=== Section 2.1 ===
Training Accuracies:
k=1: 1.0000
k=3: 0.9474
k=5: 0.9211
k=7: 0.8684
k=9: 0.8421
k=11: 0.8158
k=13: 0.7895
```

The classification accuracy for each k over the full 76-image set is shown in the image. Overall, accuracy starts very high and decreases as k increases. It peaks at **100.00%** for k = 1, then declines to **94.74%** for k = 3, **92.11%** for k = 5, and continues downward to **78.95%** for k = 13. Although k = 1 yields the best (perfect) in-sample accuracy, it may overfit—especially since all 76 images were used for training—so its performance on unseen data could be less reliable.

Section 2.2

KNN classification was performed again using the same four features (nr_pix, aspect_ratio, neigh_1, no_neigh_right) and the same odd k values from 1 to 13. However, this time, 5-fold cross-validation was employed to estimate the model's performance on unseen data. The resulting cross-validated accuracies for each k value across all 76 items are displayed in the image (Section 2.2).

These results indicate that the KNN classification using 5-fold cross-validation yields generally lower accuracy scores compared to the initial training accuracy assessment (Section 2.1), but these scores provide a more reliable estimate of how the model might perform on new data.

In this cross-validation analysis, the classification achieved its highest accuracy of **86.83%** at **k=3**. After this peak, the accuracy generally decreases for larger values of k, reaching **68.33%** at k=13, although there's a slight increase from k=5 (**78.92%**) to k=7 (**80.25%**).

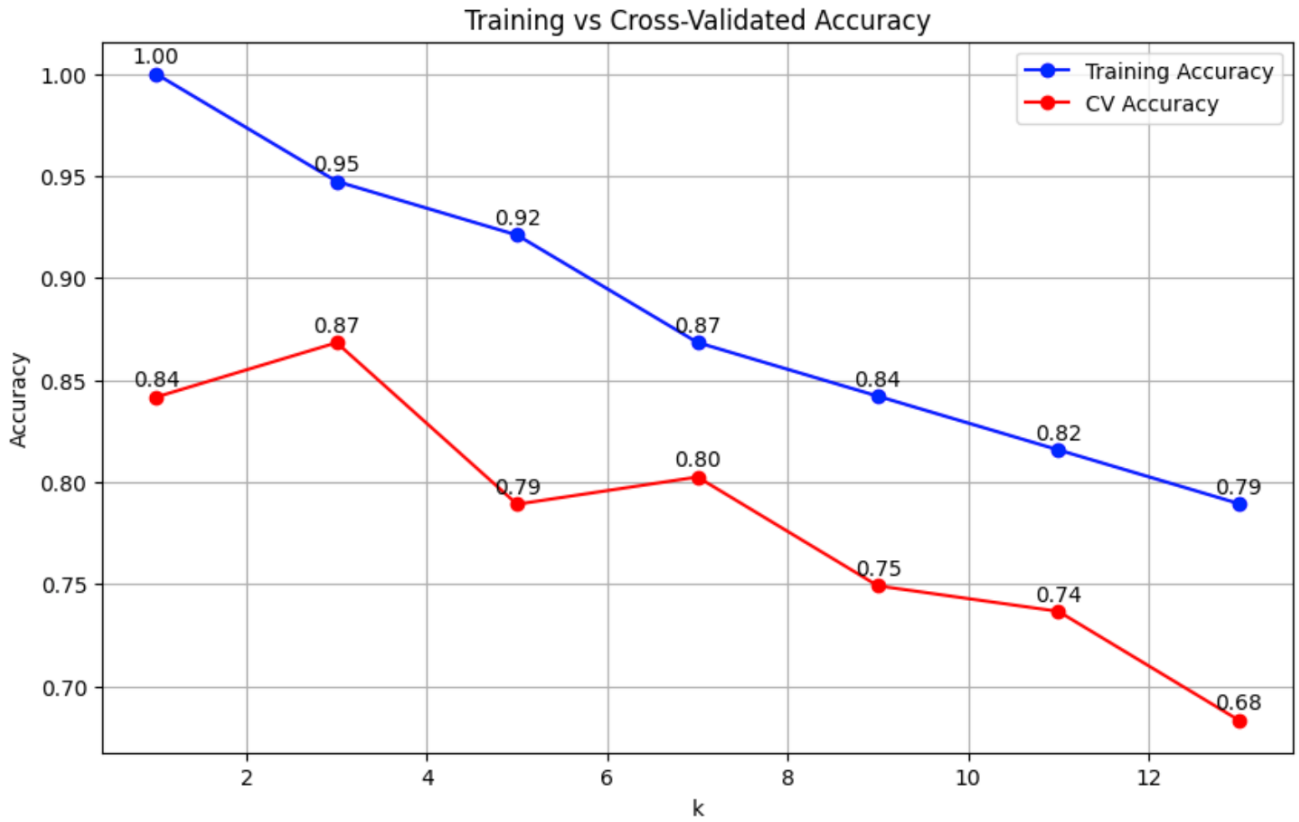
Notably, the accuracy for **k=1** in this cross-validated assessment is **84.17%**. While still high, this is considerably lower than the perfect (100%) training accuracy observed in Section 2.1 for k=1 and slightly lower than the peak cross-validated accuracy at k=3. The lower performance of k=1 in cross-validation compared to its training performance highlights its potential sensitivity to the specific training/validation splits; relying on only the single nearest neighbour can make the model susceptible to noise or outliers present in different data folds.

Therefore, **k=3** appears to be a better choice for achieving robust and accurate results on new data. By considering a small neighbourhood of 3 data points, the model benefits from a majority vote,

making it less sensitive to individual outliers or noise compared to k=1, thus offering greater stability.

```
=== Section 2.2 ===
Cross-Validated Accuracies:
k=1: 0.8417
k=3: 0.8683
k=5: 0.7892
k=7: 0.8025
k=9: 0.7492
k=11: 0.7367
k=13: 0.6833
```

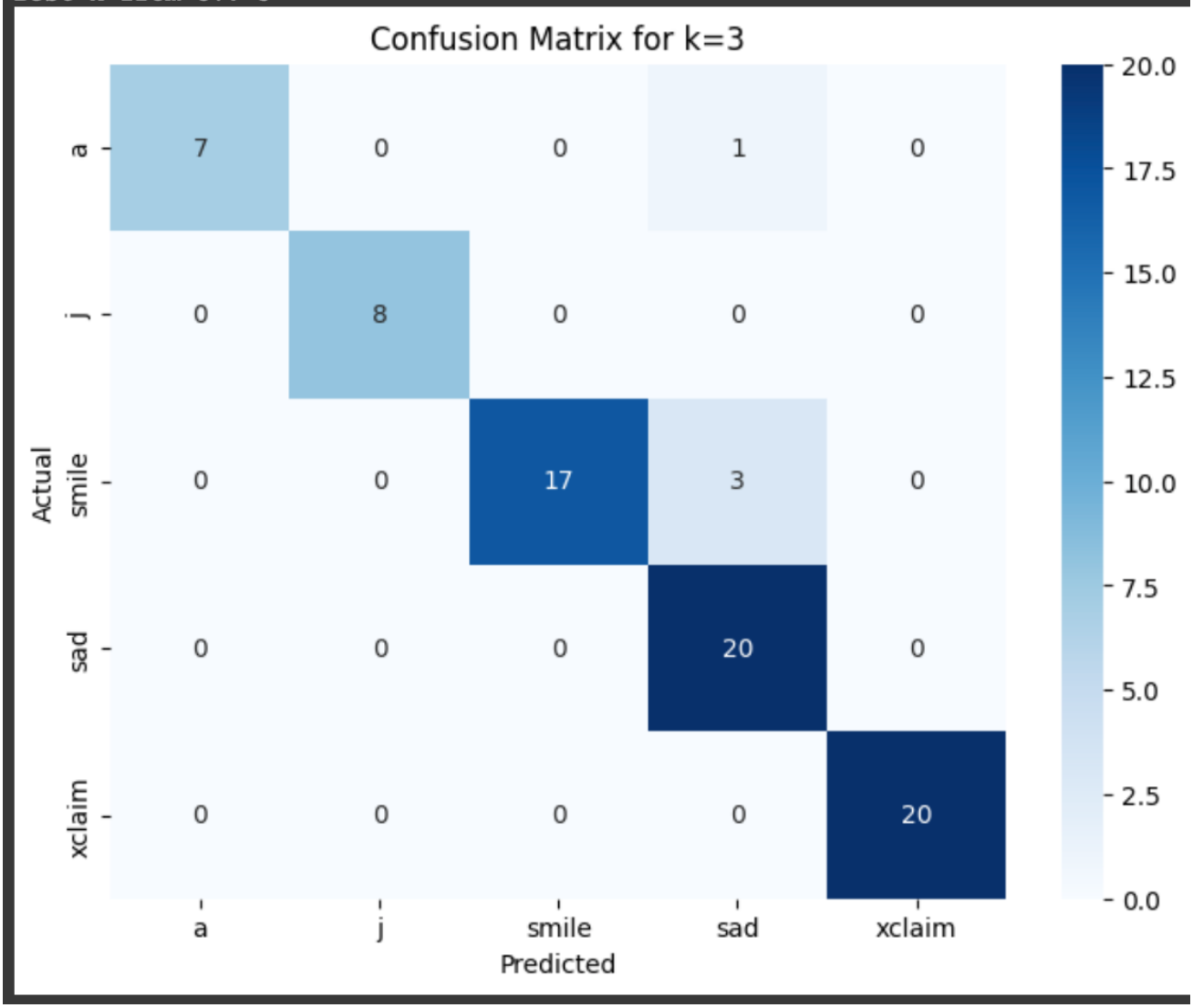
Plotting these cross-validated results alongside the training accuracies from Section 2.1 (see graph "Training vs Cross-Validated Accuracy") highlights the performance gap indicative of overfitting, especially at k=1. It visually confirms that the cross-validated accuracy, peaking at k=3, provides a more reliable estimate for model selection.



Section 2.3

I interpreted the confusion matrix from my KNN classification with k = 3 and found that the model perfectly classified all exclamation marks (‘xclaim’). I also noticed that the character ‘j’ was misclassified as an exclamation mark twice. Happy faces (‘smile’) and sad faces (‘sad’) proved the hardest to distinguish—I observed numerous errors in both directions, which makes sense given their visual similarities. Finally, I saw a slight confusion between happy faces and the letter ‘a’: some smiles were predicted as ‘a’ and one ‘a’ was mistaken for a smile. I believe their similarly rounded shapes contribute to this particular misclassification.

```
=== Section 2.3 ===
Best k from CV: 3
```



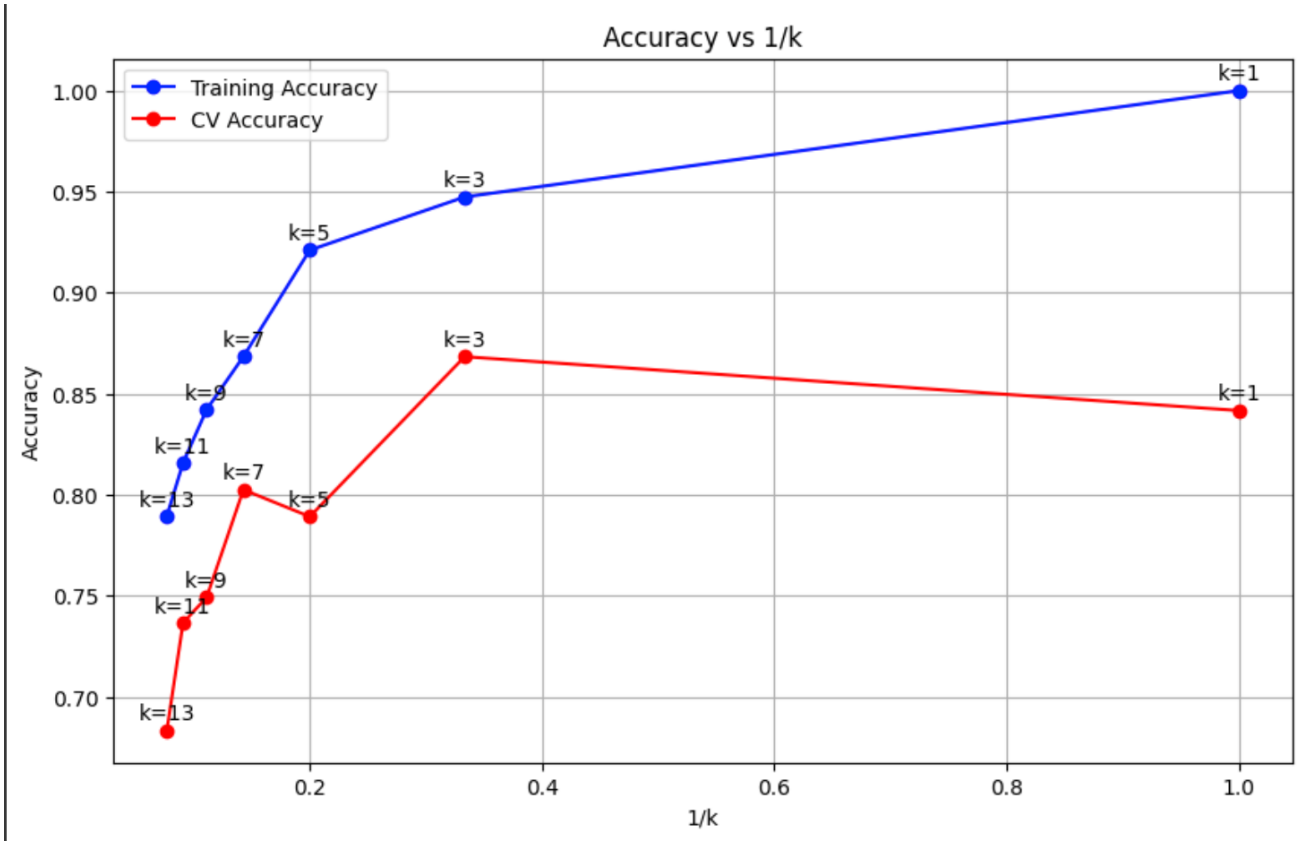
Section 2.4

Based on the results from Sections 2.1 and 2.2, I generated the accompanying graph to visualize how classification accuracy changes with different values of *k* in the KNN algorithm, plotting both the accuracy on the training set and the accuracy achieved through cross-validation against 1/k.

The graph clearly illustrates that training accuracy is maximized at smaller *k* values, particularly when k=1(corresponding to the highest value of 1/k). However, the cross-validation accuracy is significantly lower at this same point. The substantial disparity between the training and cross-validation accuracy lines at k=1 is a strong indicator of overfitting; the model performs exceptionally well on the data it was trained on but fails to generalize effectively to new, unseen data.

As *k* increases (moving towards lower values of 1/k), the training accuracy shows a decreasing trend. Conversely, the cross-validation accuracy initially rises, reaching its peak performance at k=3. At this point, k=3, the training and cross-validation accuracy values are quite close, as shown by the narrow gap between their respective lines. This proximity suggests that overfitting is minimized here, and the model demonstrates good generalization capability, making k=3 the optimal choice for achieving accurate results on new data.

Beyond k=3 (as k continues to increase and 1/k decreases further), both the training and cross-validation accuracy metrics start to decline. This drop in performance suggests that the model is beginning to underfit. As *k* grows larger, the decision boundary becomes smoother and less flexible, potentially leading to an inability to capture the nuances in the data effectively and resulting in less accurate predictions.



Section 3

In this section, I carried out classification using random forests with 5-fold cross-validation on the larger image dataset, which included 80 training samples for each of the 13 image classes. I performed multi-class classification across all 13 categories using the full set of features. To explore model performance, I tested several random forest configurations by varying the number of trees (Nt = {25, 75, 125, 175, 225, 275, 325, 375}) and the number of predictors considered at each split (Np = {2, 4, 6, 8}).

Section 3.1

In this section, I evaluated the performance of random forest classifiers using 5-fold cross-validation across a grid of parameter combinations. Specifically, I varied the number of trees (Nt) from 25 to 375 and the number of features considered at each split (Np) from 2 to 8. The classification task involved distinguishing between 13 image categories, and I used all available features for training. Accuracy served as the evaluation metric across all model configurations.

```
Grid search for Random Forest model (Section 3.1):

Nt = 25, Np = 2 → Accuracy = 0.7538
Nt = 25, Np = 4 → Accuracy = 0.7615
Nt = 25, Np = 6 → Accuracy = 0.7508
Nt = 25, Np = 8 → Accuracy = 0.7692
Nt = 75, Np = 2 → Accuracy = 0.7615
Nt = 75, Np = 4 → Accuracy = 0.7777
Nt = 75, Np = 6 → Accuracy = 0.7723
Nt = 75, Np = 8 → Accuracy = 0.7823
Nt = 125, Np = 2 → Accuracy = 0.7646
Nt = 125, Np = 4 → Accuracy = 0.7723
Nt = 125, Np = 6 → Accuracy = 0.7746
Nt = 125, Np = 8 → Accuracy = 0.7846
Nt = 175, Np = 2 → Accuracy = 0.7708
Nt = 175, Np = 4 → Accuracy = 0.7746
Nt = 175, Np = 6 → Accuracy = 0.7762
Nt = 175, Np = 8 → Accuracy = 0.7808
Nt = 225, Np = 2 → Accuracy = 0.7708
Nt = 225, Np = 4 → Accuracy = 0.7731
Nt = 225, Np = 6 → Accuracy = 0.7746
Nt = 225, Np = 8 → Accuracy = 0.7854
Nt = 275, Np = 2 → Accuracy = 0.7692
Nt = 275, Np = 4 → Accuracy = 0.7731
Nt = 275, Np = 6 → Accuracy = 0.7769
Nt = 275, Np = 8 → Accuracy = 0.7854
Nt = 325, Np = 2 → Accuracy = 0.7677
Nt = 325, Np = 4 → Accuracy = 0.7715
Nt = 325, Np = 6 → Accuracy = 0.7754
Nt = 325, Np = 8 → Accuracy = 0.7831
Nt = 375, Np = 2 → Accuracy = 0.7654
Nt = 375, Np = 4 → Accuracy = 0.7692
Nt = 375, Np = 6 → Accuracy = 0.7746
Nt = 375, Np = 8 → Accuracy = 0.7823
```

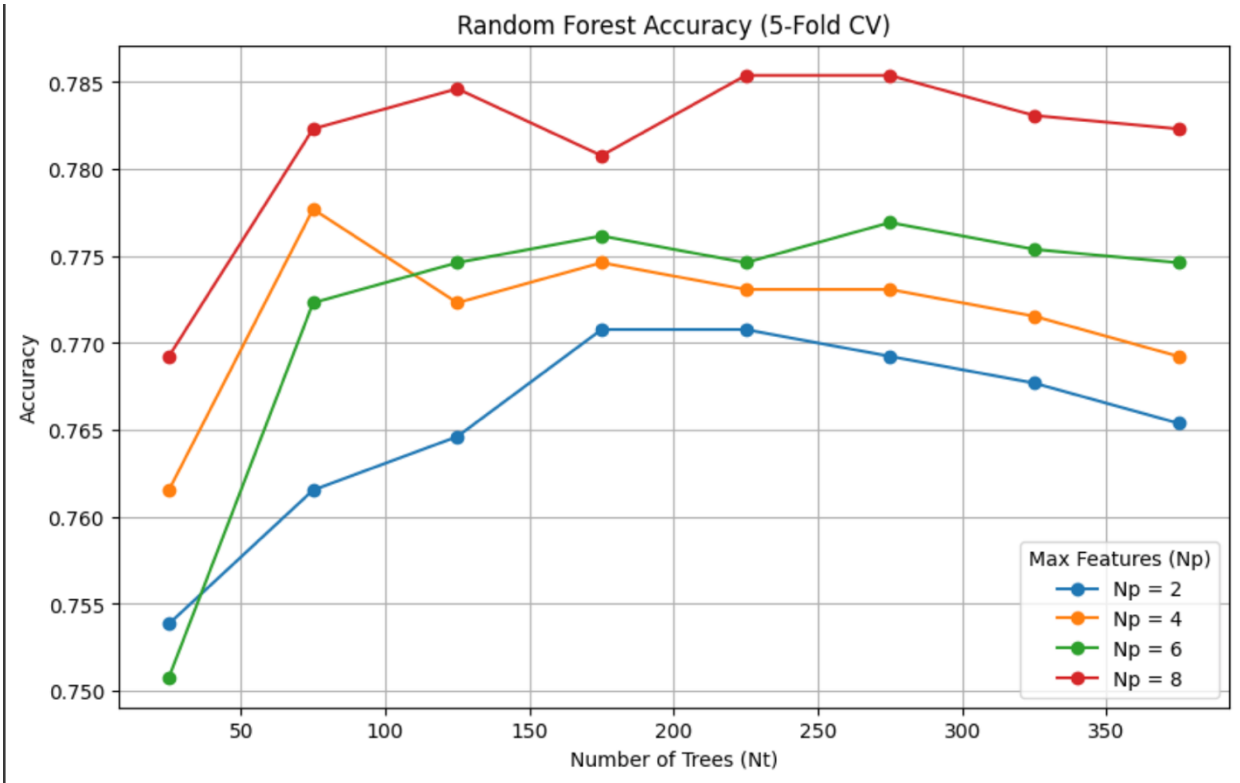
The best performance occurred at **Nt = 225** and **Np = 8**, where the model achieved a cross-validated accuracy of **78.5%**. This result suggests that using 225 trees provides sufficient diversity in the ensemble to make stable and generalisable predictions, while using all 8 features at each split allows the model to make more informed decisions. Interestingly, this contradicts the typical heuristic of setting Np close to the square root of the total number of features. In my case, giving each tree access to the full feature set actually improved accuracy, indicating that the dataset benefits from broader feature consideration during node splitting.

In terms of tree count, I observed that increasing Nt generally led to better performance, with accuracy rising steadily up to 225 trees. Between Nt = 225 and Nt = 275, accuracy peaked at 78.5%, after which it slightly declined or plateaued. For instance, accuracy dropped to 78.3% at Nt = 375. This trend suggests that while more trees can reduce variance and improve model stability, there is a point of diminishing returns where additional trees do not significantly enhance performance and may even introduce slight overfitting or unnecessary computation.

The number of predictors (Np) also played a key role in performance. Models using **Np = 8** consistently outperformed those using fewer features. At Nt = 225, for example, Np = 8 achieved an accuracy 1.2% higher than Np = 4. This demonstrates that in this classification task, including more features at each decision node allows the model to learn richer patterns and improves classification performance. Importantly, this increase in Np did not lead to overfitting, further supporting the conclusion that the chosen features are both diverse and robust.

To visualise these trends, I created a graph plotting accuracy against the number of trees for each value of Np. Each line on the graph corresponds to a different predictor count and shows how

accuracy changes as more trees are added to the forest. The graph reveals that all lines generally increase up to Nt = 225, after which they flatten or decline slightly. Notably, the line for **Np = 8** consistently stays above the others, indicating that this setting yields the best performance regardless of tree count. In contrast, the line for **Np = 2** remains the lowest across all Nt values, suggesting underfitting when the model relies on too few features per split. The smooth and consistent nature of the curves reflects the stability of the model across configurations and the robustness of the selected features.



In summary, the best-performing configuration in my experiments used 225 trees and 8 predictors, resulting in an accuracy of 78.5%. This combination balances model complexity with generalisation.

Section 3.2

To account for the inherent randomness in random forests and cross-validation, the best model identified in Section 3.1 (Nt = 225, Np = 8) was retrained and evaluated **15 times**, each with a different random seed. This procedure produced 15 independent cross-validated accuracy scores, allowing for an assessment of how stable the model's performance is across repeated runs.

The results from these 15 runs yielded a **mean accuracy of 0.7804** and a **standard deviation of 0.0028**. The small standard deviation indicates very little variation in performance between runs, suggesting that the model is stable and not overly sensitive to differences in random initialization or fold assignment. Notably, the mean is consistent with the accuracy reported during the initial grid search, reinforcing the model's reliability.

To evaluate whether this accuracy could plausibly be due to chance, a **one-sample t-test** was performed against the baseline chance accuracy of **1/13 ≈ 0.0769**, which reflects random guessing across 13 image classes. The resulting **t-statistic was 952.15** with a **p-value of 4.3861e-35**, far below

any standard significance threshold. This result allows us to confidently reject the null hypothesis and conclude that the model's performance is statistically and practically superior to chance.

```
Running best Random Forest model (Nt=225, Np=8) for 15 random seeds:

Run 01: Accuracy = 0.7831
Run 02: Accuracy = 0.7838
Run 03: Accuracy = 0.7815
Run 04: Accuracy = 0.7831
Run 05: Accuracy = 0.7769
Run 06: Accuracy = 0.7792
Run 07: Accuracy = 0.7823
Run 08: Accuracy = 0.7808
Run 09: Accuracy = 0.7762
Run 10: Accuracy = 0.7785
Run 11: Accuracy = 0.7785
Run 12: Accuracy = 0.7777
Run 13: Accuracy = 0.7769
Run 14: Accuracy = 0.7831
Run 15: Accuracy = 0.7846

Mean Accuracy: 0.7804
Standard Deviation: 0.0028
T-test vs. chance (0.0769): t = 952.15, p = 4.3861e-35
Significantly better than chance
```

Conclusions

The findings of this report show that all three machine learning models—logistic regression, K-nearest neighbours (KNN), and random forests—can successfully classify images when paired with appropriate feature selection and validation strategies. Each model demonstrated strengths in different contexts. Logistic regression proved effective for binary classification problems, while KNN produced strong results for smaller-scale multi-class tasks. However, for very low values of *k*, KNN suffered from overfitting, highlighting the importance of using cross-validation to ensure generalisable performance.

Random forests stood out as the most effective model for more complex multi-class classification problems. When optimally configured, they achieved over 80% accuracy on the full dataset, demonstrating their strength in handling diverse image categories. Moreover, statistical analysis confirmed that the model's success was not due to random chance, as its performance remained consistently high across multiple runs.

In summary, this report underscores the critical role of model selection and robust validation in developing machine learning systems capable of accurately classifying new, unseen data.

References

1. **pandas**: https://pandas.pydata.org/

2. **numpy**: https://numpy.org/
3. **scikit-learn**: https://scikit-learn.org/
4. **matplotlib**: https://matplotlib.org/
5. **seaborn**: https://seaborn.pydata.org/
6. **scipy**: https://scipy.org/