

# OCMP5318: Assignment 1

## Image Classification of Street House View House Numbers

Written By: Ayra Islam



## Introduction

Reading text from natural images is a well-established challenge amongst data scientists and researchers. Accurate digit recognition has significant practical applications in navigation, mapping services, and optical character recognition (OCR). Enhancing model performance in this task contributes to broader advancements in automated reading systems and real-world computer vision applications. For instance, Google Street View in Google Maps can benefit from more accurate address mapping, which is relied on by millions to navigate to desired locations [17].

To investigate further into this problem, researchers from Stanford University introduced the Street View House Numbers (SVHN) dataset, which consists of real-world images of house numbers extracted from Google Street View [17]. The dataset presents challenges such as blur, distortion, illumination changes, and varying font styles, making it an ideal benchmark for simulating the current challenges of image recognition in a natural setting.

This report aims to evaluate and compare machine learning models assigned for digit recognition on the SVHN dataset. Inputs for this project are two cropped 32x32 Red Green Blue Digits (RGB) files saved in binary MATLAB files (.mat). These two files contain training and test data. Analysis and preprocessing techniques such as feature selection methods were applied to the files to enhance data quality, optimise computational efficiency and to ensure images are fit for model training. The expected output are results which demonstrate effectiveness of chosen model in classifying images and ultimately, the most effective model for classifying house digit numbers in complex real-world conditions.

While deep learning techniques are widely used for image recognition, this study focuses on traditional machine learning models to establish a baseline for performance evaluation. These findings provide insights into the dataset's underlying challenges and patterns, serving as a foundation for future research involving more advanced modeling techniques.

# Methodology

## Data Preprocessing

Data preprocessing is crucial for transforming raw data into a clean, structured format, directly influencing model performance and generalisability. Several challenges were identified in the SVHN dataset, including a skewed distribution towards lower digit classes (e.g., 1 and 2), high-dimensional image data with numerous pixel features and color channels, leading to long training times.

### Label Fixing

In the original dataset, some images containing the digit '0' were incorrectly labeled as '10'. This was corrected by reassigning such labels to '0' to ensure proper classification, particularly in cases where '0' was preceded by a digit other than one (e.g., the '2' in "204"). Correct labelling ensures images are accurately classified and interpreted as expected.

### Normalisation and Gray Scale Conversion

Normalisation was applied to rescale pixel values to a range between 0 and 1. This was achieved by dividing pixel features by 255, which is the maximum possible pixel intensity. This step reduces numerical instability and prevents large values from disproportionately influencing the model [3].

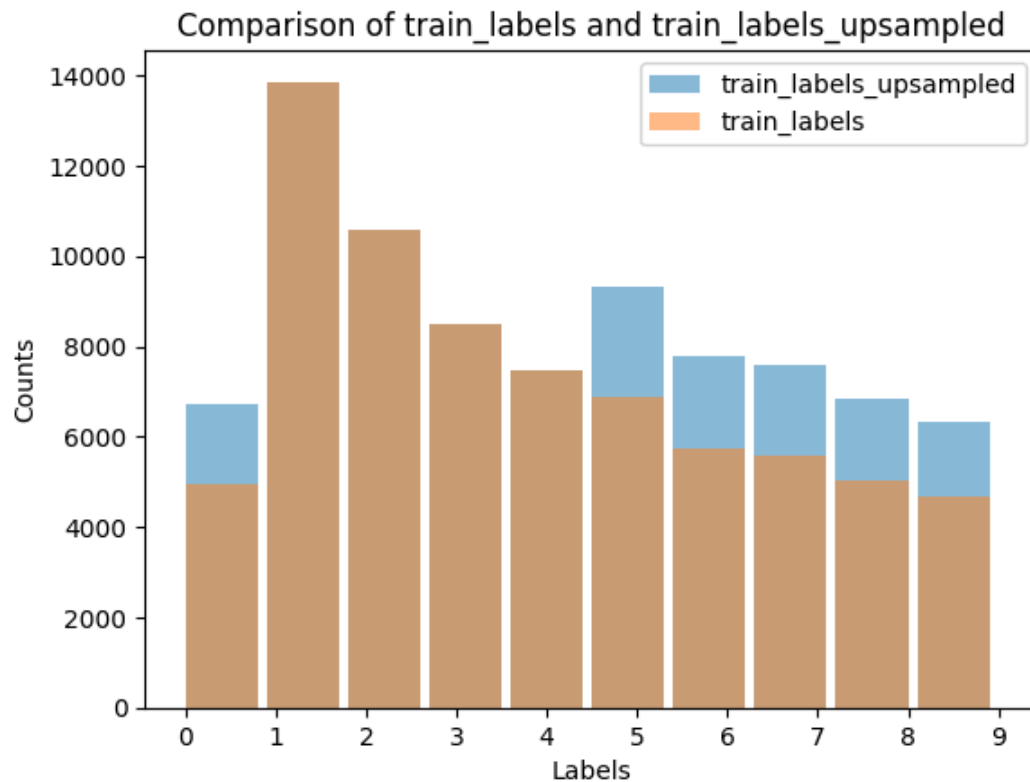
Grayscale conversion was achieved by transforming RGB images into a single-channel grayscale format. This conversion is a common practice in image processing to reduce complexity and improve computational efficiency [12].

### Handling of imbalanced datasets

A key challenge of the SVHN dataset was class imbalance, with positive skewness toward lower digit classes, in particular class '1'. Data augmentation was applied to upsample underrepresented classes ('0' and '6' through '9'). This method was chosen to increase volume and diversity of the dataset, preventing overfitting and improving generalisability across real-world variations [14].

The python package *Albumentations* is frequently used to augment images and upsample training data. This library was chosen in particular for its extensive repository of image transformation techniques, which are widely used in neural network-based image preprocessing [7]. Images from minority classes were picked and augmented randomly by one or more transformations. These transformations included horizontal and vertical flipping, random shifting, scaling, rotating, blurring and varying lighting conditions. Such transformations were chosen and modulated to ensure digits were still visible but still simulated real world image variation.

However, upsampling was applied with constraints to prevent excessive data expansion. The total sample count per class was limited to 86,000 to maintain reasonable training times and computational efficiency. While this restriction resulted in a moderate reduction in the global imbalance ratio (from 2.12 to 1.71), implementation of ensemble methods Random Forest and Gradient Boosting were inherently more robust to imbalanced class distributions .



*The orange bars refer to pre-sampled counts and blue bars refer to the additional augmented images added for the targeted minority classes.*

### **Standardisation and Principal Component Analysis (PCA) decomposition**

Pixel features were standardised prior to PCA decomposition to ensure all features contributed equally and avoid disproportionate variance in magnitude that would bias results.

PCA was applied to significantly reduce the dimensionality of the SVHN dataset, making the data more computationally efficient to process. PCA is a dimensionality reduction method that compresses features into principal components which hold the most important information in a dataset. The technique is frequently used for compressing images into an acceptable format, thus ensuring images can be efficiently analysed by machine learning algorithms. After applying PCA, features were reduced from 1024 to 48 features.

## Choice of Predictive Machine Learning Techniques

Four models were selected based on their effectiveness, computational efficiency, and feasibility within the constraints of this project. The chosen models include two ensemble methods (Random Forest and XGBoost), a probabilistic classifier (Naïve Bayes), and a distance-based algorithm (K-Nearest Neighbors). These methods provide a diverse set of approaches for evaluating digit classification in the SVHN dataset.

### Random Forest

Random Forest is an ensemble method that constructs multiple decision trees using bootstrapped samples and a random subset of features. Decision trees are independently trained, with each tree learning to make decisions based on the best available attribute for splitting the data at each node. The goal is to split the data such that the resulting subsets are as pure as possible, meaning the instances in each subset should ideally belong to the same class. To measure this "purity," common metrics like **Gini Impurity** or **Entropy** are used.

For classification problems, **Entropy** can be defined as

$$H(S) = - \sum_{i=1}^C p_i \log_2 p_i$$

- $C$  refers to the number of classes,
- $p_i$  is the proportion of examples that belong to class  $i$ .

Random Forest was chosen for several reasons including robustness to overfitting and handling of large imbalanced datasets which do not require much preprocessing beforehand. They are also able to capture non-linear relationships in images affected by unpredictable factors. Although Random Forest is not traditionally a first choice for image classification tasks, it has been shown to handle non-linear relationships with pixel data effectively and generalize well to unseen images [6] [17].

### XGBoostClassifier (Extreme Gradient Boosting)

XGBoostClassifier is an ensemble technique based on gradient boosting, designed for scalability and efficiency [4]. Decision trees are built sequentially, with gradient descent used to minimise a loss function to correct the residual errors of the previous tree.

Predictions are iteratively updated as follows:

$$F_t(x) = F_{t-1}(x) + \alpha \cdot h_t(x)$$

- $F_t(x)$  is the current model prediction and  $F_{t-1}(x)$  is the previous,
- $\alpha$  is the learning rate and  $h_t(x)$  is the new tree trained on residuals.

XGBoost was favoured due to its higher predictive power compared to traditional ensemble methods, regularisation mechanisms to handle noisiness within datasets and has been optimised for scalability. The classifier has been applied for many image classification cases

and has been frequently used in partnership with neural networks to optimise classification methods [11].

### Naive Bayes Classifier

Naive Bayes is a probabilistic technique that classifies data under the assumption features are conditionally independent and are of equal importance for each class.

Predictions are based on Bayes theorem:

Given a hypothesis  $H$  and evidence  $E$  for this hypothesis, then the probability of  $H$  given  $E$  is:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)}$$

- $P(E|H)$  is the posterior probability of evidence  $E$  given some knowledge of  $H$
- $P(H)$  is the prior probability of  $H$
- $P(E)$  is the probability of the evidence, which is computed as a sum over all classes.

Naive Bayes was chosen for its computational efficiency, implementation ease and ability to perform reasonably well in image classification tasks. In particular, it has been able to produce reasonable accuracy in digit recognition on the MNIST dataset [5] and is frequently used in document and text classification [10]. Furthermore, including a simple probabilistic model helped to assess the effectiveness of more complex methods such as Random Forest and XGBoost Classifier.

### K-Nearest Neighbours Classifier

K-Nearest Neighbours (KNN) Classifier is a non-parametric, lazy learning algorithm that requires no training prior to making real time predictions. New data points are classified based on the majority class of the  $k$  nearest dot points. The choice of  $k$  is important, however nearness can be determined by several distance metrics, with the popular one being Euclidean distance.

Suppose  $A$  and  $B$  are data points with attribute values  $a_1, a_2, \dots, a_n$  and  $b_1, b_2, \dots, b_n$ . Euclidean distance between two points are:

$$D(A, B) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

KNN was chosen as no explicit model training is required to classify new data points. Furthermore, it is flexible with the choice of distance metrics which can significantly affect results. KNN is not as robust as ensemble methods on high dimensional, imbalanced data, therefore making it a useful baseline to compare against its more advanced counterparts.

## Choice of Evaluation Metrics

F1-score was the primary evaluation metric for hyperparameter tuning, as it balances precision and recall, making it well-suited for imbalanced datasets. Additional metrics, including precision, recall, and AUC-ROC, were used for a more comprehensive evaluation of model performance on the test set. Accuracy was deliberately avoided due to its bias towards the majority class, which can mask poor classification performance for minority classes. By using alternative metrics, it was possible to gain a more nuanced understanding of the models' ability to correctly classify minority class instances [14].

## Hyperparameter Tuning Methods

A base model was initially trained for each classifier to establish a performance baseline before hyperparameter tuning. Grid Search with 5-fold cross-validation was used to optimize hyperparameters, with parameters tuned individually rather than simultaneously to reduce computational cost.

### Random Forest

Six key hyperparameters were tuned: `n_estimators`, `max_depth`, `min_samples_split`, `min_samples_leaf`, `max_features`, and `criterion`. Initial tuning was performed using Grid Search, and the best values were selected based on cross-validation F1-score. However, the model exhibited signs of overfitting, particularly in the majority class. To mitigate this, `min_samples_split`, `min_samples_leaf` and `max_features` were reduced, leading to slight improvements in minority class prediction.

### XGBoost

XGBoost hyperparameter tuning followed a similar approach, with training conducted on GPU to accelerate computation. The parameters tuned included `n_estimators`, `max_depth`, `min_child_weight`, `subsample`, and `learning_rate`. The initial model with all optimized parameters resulted in a lower F1-score than the base model, indicating overfitting. To address this, regularization parameters `reg_alpha` and `reg_lambda` were introduced, along with `colsample_bytree` to limit the fraction of features used per tree [13]. Despite these adjustments, the base model still outperformed the tuned model, and therefore, the default configuration was retained.

### Naive Bayes & K-Nearest Neighbors (KNN)

Naive Bayes required tuning for variance smoothing and priors, while KNN tuning focused on number of neighbours, algorithm to execute distance metric, choice of distance metric, and weights. Since KNN lacks built-in mechanisms for handling imbalanced data, the weights parameter was set to "distance", ensuring that closer neighbors had a greater influence on classification decisions.

# Evaluation and Comparison

## Experimental Setting

After applying data preprocessing steps, models were trained on 84,997 samples from the training dataset and evaluated on 26,032 samples from the test dataset. The Scikit-Learn library was used to implement Random Forest, KNN, and Gaussian Naïve Bayes classifiers, while the XGBoost library was used for the XGBoost classifier.

### Class Imbalance Handling

Class imbalance was addressed by adjusting class weights in Random Forest and XGBoost, ensuring that the models did not overly favor majority classes.

### Evaluation Metrics

Models were evaluated using the weighted F1-score, which provides a more balanced assessment by accounting for both false positives and false negatives across all classes.

### Hyperparameter Tuning

Hyperparameters were tuned using Grid Search with 5-fold cross-validation. To reduce excessive training time, parameters were optimized one at a time rather than searching all at once. CPU was used throughout tuning, however GPU was used solely for tuning XGBoost parameters.

### Hyperparameters and Best Values

Base models were built for all four techniques to a baseline to compare against after all parameters were tuned. Further improvements were made after tuning hyperparameters since testing attributes individually did not ascertain optimal performance when all parameters were used to evaluate model performance based on test data.

### Random Forest Classifier

Hyperparameters Tuned	Best Value Found
Number of Estimators	300
Max Depth	30
Min Samples Split	10



Min Samples Leaf	7
Max Features	sqrt
Criterion	Entropy

### **XGBoost Classifier**

Hyperparameters Tuned	Best Value Found
Number of Estimators	300
Max Depth	10
Learning Rate	0.2
Min Child Weight	7
Subsample	0.8

### **Naive Bayes Classifier**

Hyperparameters Tuned	Best Value Found
Variance Smoothing	1e-9
Priors	Uniform Priors (0.1) for each test label

### **KNN Classifier**

Hyperparameters Tuned	Best Value Found
Number of Neighbours	20
Weights	Distance
Algorithm	auto
Metric	Manhattan

## Experimental Results

Model	F1 Score	Precision	Recall	AUC	Time Taken
Random Forest	0.168	0.167	0.171	0.566	17 minutes
XGBoost	0.170	0.173	0.168	0.562	43 seconds
Naive Bayes	0.138	0.151	0.170	0.545	0.08 seconds
K Nearest Neighbours	0.165	0.160	0.192	0.565	0.01 seconds

Upon observation, XGBoost performed the best in terms of F1-Score, suggesting it balanced precision and recall better than other models. Additionally, it had the highest precision, meaning it had made fewer false positive predictions compared to others. On the other hand, KNN scored the highest recall, meaning it was better at identifying actual positive instances. However, its precision was lower, indicating a higher false positive rate. Furthermore, all AUC values were near 0.5, suggesting that all predictions were no better than just mere guessing.

In terms of computational efficiency, XGBoost had a major advantage over Random Forest (43 secs vs 17 minutes) thus being a viable option for accuracy and speed. Naive Bayes and KNN classifiers were faster however this came at the expense of predictive performance. Naive Bayes performed the worst in terms of F1 score, which is expected given patterns in image datasets cannot be fully captured by the assumption all features are equally important and independent.

For all models, correct predictions were driven by digit class 1, resulting from class imbalance in the dataset (see Appendix 1). KNN had the highest correct instances for digit 1, suggesting it heavily relied on nearest neighbours without adjusting for class imbalance. XGBoost, while having the lowest correct predictions for class 1, spread its predictions more evenly across other classes, which might indicate a more balanced classification strategy.

## Discussion

In this project, PCA was experimented to reduce dimensionality, however may have led to an over-reduction and thus compromised predictive accuracy. Instead, a more effective approach would have been to select the most important pixel regions to retain as much information as possible. An experiment for this project was conducted involving the cropping of images to size 54x54 pixels to replicate similar research on SVHN dataset, but this approach led to prolonged training times [16]. Thus, justifying the importance of focalising most important pixel regions to retain as much information as possible.

Several feature selection methods could also enhance the most crucial aspect of images. For instance, Histogram of Gradients is a popular and well-established method for highlighting the edges and points that define an image, minimising focus on less important regions [2].

Another major challenge was class imbalance, which negatively impacted all models. Predictions were skewed towards the majority class (digit "1"), leading to poor generalization across other digits. Although upsampling was applied to mitigate this, it was not extensive enough to create a truly balanced dataset. One approach to address this issue would be to generate additional training data, but experimentation with this method came at a computational cost, significantly increasing training time. A more effective alternative would be to apply advanced upsampling techniques, such as Synthetic Minority Oversampling Technique (SMOTE) which generates synthetic data points by interpolating between existing samples of the minority class, thereby increasing class representation without simply duplicating data [14]. Another approach is Generative Adversarial Networks (GANs), which learn the underlying distribution of the dataset and generate realistic synthetic images, providing a more natural data augmentation approach [1].

Finally, better models could have been used. Support Vector Machines are typically better suited for non-linear relationships and might have outperformed both Random Forest and XGBoost, given that digit classification often involves complex decision boundaries. Furthermore, stronger baseline models compared to KNN and Naives Bayes could have been used. For instance, decision trees allow for a cost sensitive learning approach, ensuring class imbalance is considered in model prediction. Whilst this study was restricted to traditional machine learning methods, a Convolution Neural Network (CNN) would have likely outperformed all models. This is because they are able to extract hierarchical representations of image content, making them better suited for digit recognition tasks [8].

## Conclusion

XGBoost provided the best balance between precision and recall while also being computationally efficient. However, all models struggled due to severe class imbalance and feature limitations. Future work could focus on more advanced upsampling techniques, feature selection methods such as Histogram of Gradients and deep learning approaches to improve classification performance.

# Appendix

## Information about code


All code was run using Google Colab using an Apple M1 Macbook Air Laptop with 8GB.  
All code was run using CPU; however for training XGBoost Classifier, GPU was used.

## Instructions for running code

1. Include `train_32x32.mat` and `test_32x32.mat` into Google Drive
2. Proceed running code from section 2.3 Load and analyse the data. Please change directory to the location where the dataset has been saved.

## ✓ 2.3 Load and analyze the data

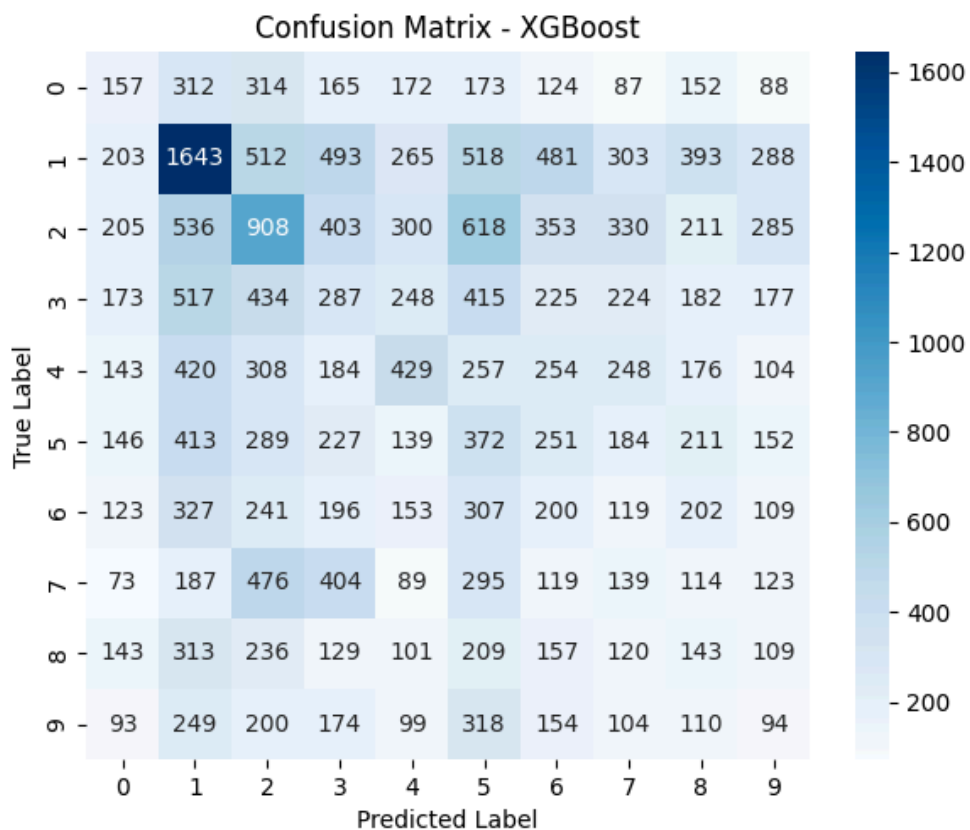
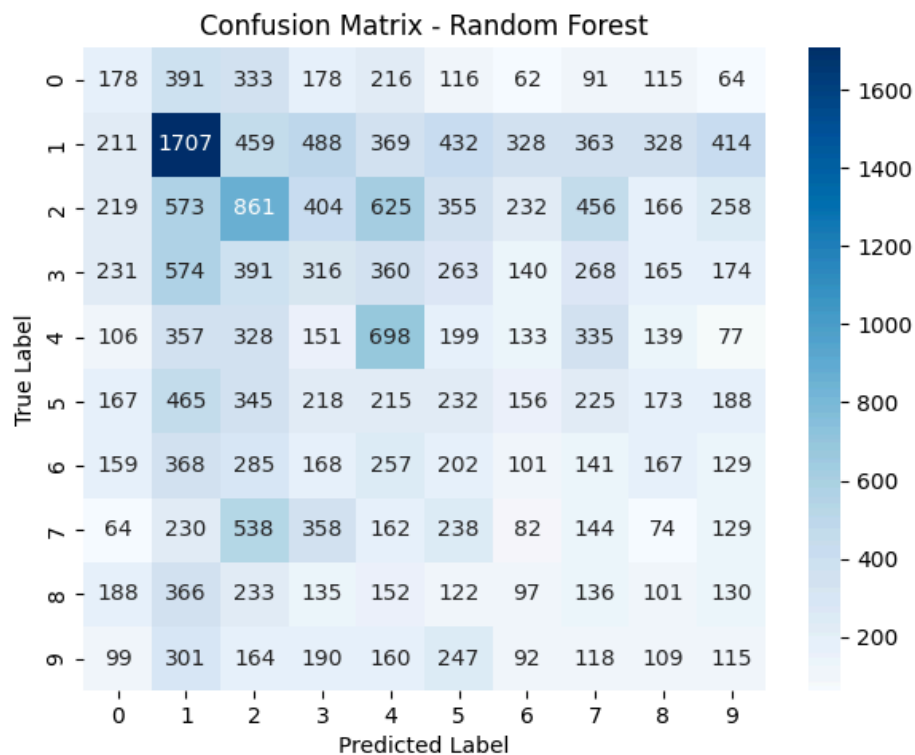
```
[ ] # Set directory
    from google.colab import drive
    drive.mount('/content/drive')
```

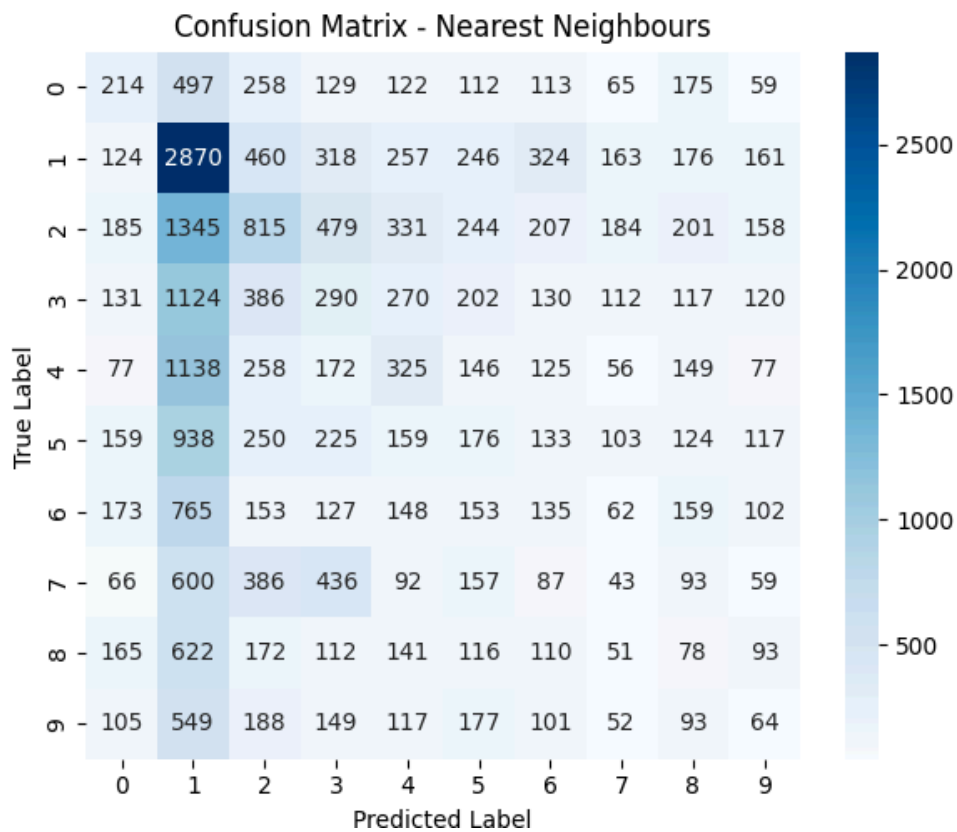
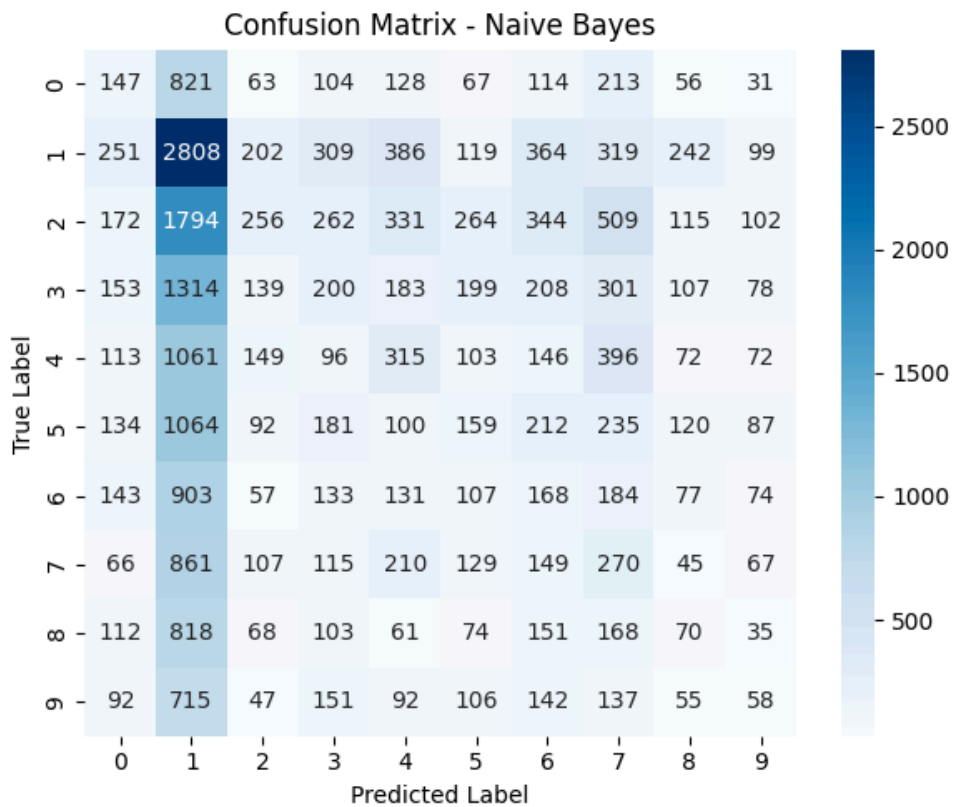
 Mounted at /content/drive

```
[ ] data_path = "/content/drive/My Drive/data_asg1"
```

*(see next page for Appendix 1)*

Appendix 1 - Correlation matrices for final models





## References

- [1] Advancing Analytics. (2023, February 2). *Image classification: Dealing with imbalance in datasets*. Retrieved March 23, 2025, from <https://www.advancinganalytics.co.uk/blog/2023/2/2/image-classification-dealing-with-imbalance-in-datasets>
- [2] Analytics Vidhya. (n.d.). *A gentle introduction to the histogram of oriented gradients (HOG)*. Retrieved March 23, 2025, from <https://medium.com/analytics-vidhya/a-gentle-introduction-into-the-histogram-of-oriented-gradients-fdee9ed8f2aa>
- [3] Analytics Vidhya. (n.d.). *Why should we normalize image pixel values (divide by 255)?* Retrieved March 20, 2025, from <https://medium.com/analytics-vidhya/a-tip-a-day-python-tips-8-why-should-we-normalize-image-pixel-values-or-divide-by-255-4608ac5cd26a>
- [4] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. <https://doi.org/10.1145/2939672.2939785>
- [5] Diellorhoxhaj. (2023, July 27). *Building a Naïve Bayes Classifier from Scratch and Classification of handwritten digits (MNIST dataset)*. Medium. Retrieved March 18, 2025 from <https://medium.com/@diellorhoxhaj/building-a-na%C3%AFve-bayes-classifier-from-scratch-and-classification-of-handwritten-digits-mnist-50440d5d110c>
- [6] Gautam, R., Chandran, S., & Hormese, J. (2020). Classification of buildings and vehicles in Google Map satellite images using random forest classifier. In P. K. Mallick, P. Meher, A. Majumder, & S. K. Das (Eds.), *Electronic Systems and Intelligent Computing* (Vol. 686). Springer. [https://doi.org/10.1007/978-981-15-7031-5\\_102](https://doi.org/10.1007/978-981-15-7031-5_102)
- [7] GeeksforGeeks. (n.d.). *Enhancing deep learning models with Albumentations: A guide to image augmentation*. Retrieved March 18, 2025, from <https://www.geeksforgeeks.org/enhancing-deep-learning-models-with-albumentations-a-guide-to-image-augmentation/>
- [8] Google Developers. (n.d.). *Convolutional neural networks (CNNs) for image classification*. Retrieved March 23, 2025, from <https://developers.google.com/machine-learning/practica/image-classification/convolutional-neural-networks>
- [10] IBM. (n.d.). *Naïve Bayes classification*. Retrieved March 21, 2025, from <https://www.ibm.com/think/topics/naive-bayes>
- [11] Jonaac. (n.d.). *Deep XGBoost*. Retrieved March 20, 2025, from <https://jonaac.github.io/works/deepxgboost.html>
- [12] Kanan, C., & Cottrell, G. W. (2012). *Color-to-Grayscale: Does the Method Matter in Image Recognition?* *PLoS ONE*, 7(1), e29740. <https://doi.org/10.1371/journal.pone.0029740>
- [13] *Notes on Parameter Tuning — xgboost 1.6.0 documentation*. (n.d.). Xgboost.readthedocs.io. [https://xgboost.readthedocs.io/en/stable/tutorials/param\\_tuning.html](https://xgboost.readthedocs.io/en/stable/tutorials/param_tuning.html)

- [14] Okeshakarunarithne. (2024, September 10). *Handling Class Imbalance in Image Classification: Techniques and Best Practices*. Medium.  
<https://medium.com/@okeshakarunarithne/handling-class-imbalance-in-image-classification-techniques-and-best-practices-c539214440b0>
- [15] Picsellia. (n.d.). *Improve imbalanced datasets in computer vision*. Retrieved March 18, 2025, from <https://www.picsellia.com/post/improve-imbalanced-datasets-in-computer-vision#5-Class-Weighting>
- [16] R. T. Younesi, J. Tanha, S. Namvar and S. H. Mostafaei, (2024). A Deep Learning-Based Model for House Number Detection and Recognition. *2024 32nd International Conference on Electrical Engineering (ICEE)*, Tehran, Iran, Islamic Republic of, 2024, pp. 1-5,  
<https://doi.org/10.1109/icee63041.2024.10668257>
- [17] Springer. (n.d.). *Classification of buildings and vehicles in Google Map satellite images using random forest classifier*. Retrieved March 19, 2025, from [https://link.springer.com/chapter/10.1007/978-981-15-7031-5\\_102](https://link.springer.com/chapter/10.1007/978-981-15-7031-5_102)
- [18] Stanford University. (2011). *Street view house numbers dataset (SVHN)*. Retrieved March 16, 2025, from [http://ufldl.stanford.edu/housenumbers/nips2011\\_housenumbers.pdf](http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf)