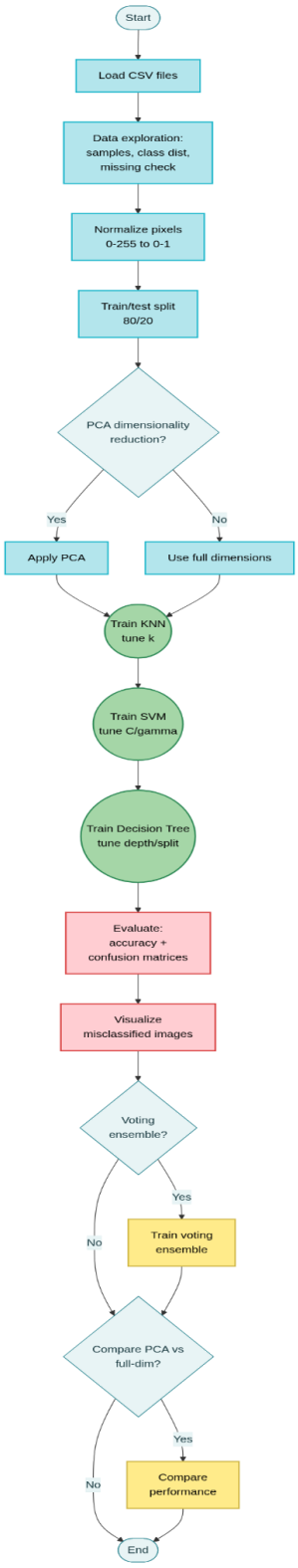
Assignment Report (Virtualyyst assignment)

Execution flow



- First step included downloading the dataset, reading the pandas file, and applying pixel wise column for each 784 pixels, along with the label.
- Then, checked for missing values, handled it, and obtained the class distribution for all the images.

Output

Class distribution (train):

label

0.0 980

1.0 1135

2.0 1032

3.0 1010

4.0 982

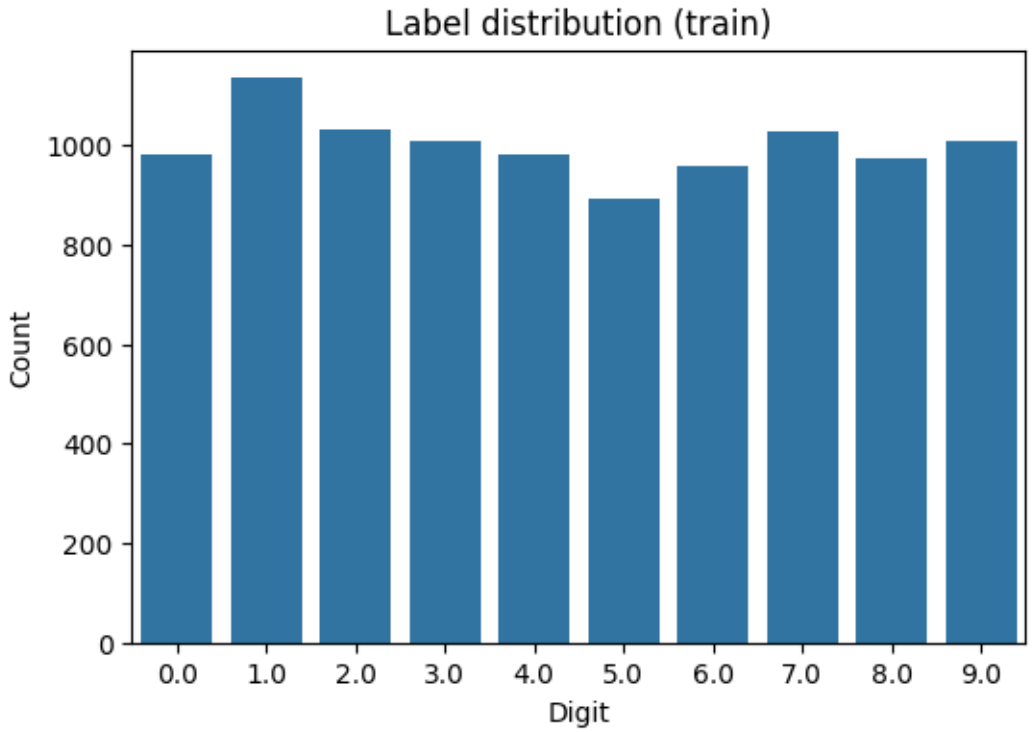
5.0 892

6.0 958

7.0 1028

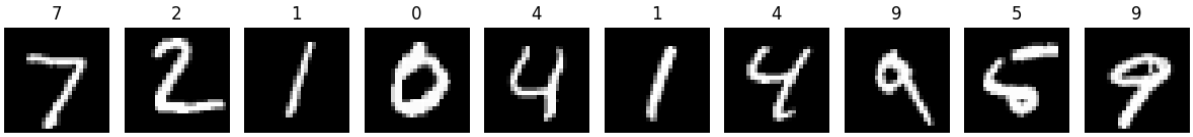
8.0 974

9.0 1009



Observation: This shows that we have the most images of class 1 and least images for class 5

- Then, normalizing of the images was done on the training dataset, and we have sample images for the training set:



- Also applied PCA, assuming 50 components and printing their explained variance ratio (shows proportion of the total variability captured of the dataset)
- Then KNN was applied with number of k components: 1, 3, 5, 7 and 9 to check how the algorithm performs with different components

Output

KNN (k=1) test accuracy: 0.9535

KNN (k=3) test accuracy: 0.9550

KNN (k=5) test accuracy: 0.9475

KNN (k=7) test accuracy: 0.9430

KNN (k=9) test accuracy: 0.9420

Best KNN -> k=3, test accuracy=0.9550

Observation: This shows that the algorithm works best with k=3 components with accuracy of 95.50%

- Next, tried to train the dataset with SVM algorithm trying various parameters. Tried both linear and rbf kernel, but the model works best with rbf kernel.

Output

SVM RBF (C=0.1, gamma=0.001) test accuracy: 0.8480

SVM RBF (C=0.1, gamma=0.01) test accuracy: 0.9225

SVM RBF (C=0.1, gamma=0.1) test accuracy: 0.7100

SVM RBF (C=1, gamma=0.001) test accuracy: 0.9115

SVM RBF (C=1, gamma=0.01) test accuracy: 0.9545

SVM RBF (C=1, gamma=0.1) test accuracy: 0.9610

SVM RBF (C=10, gamma=0.001) test accuracy: 0.9330

SVM RBF (C=10, gamma=0.01) test accuracy: 0.9685

SVM RBF (C=10, gamma=0.1) test accuracy: 0.9630

Best SVM RBF -> {'kernel': 'rbf', 'C': 10, 'gamma': 0.01}, test accuracy=0.9685

Observation: This shows that the algorithm works best with RBF kernel and gamma value 0.01 with an accuracy of 96.85%

- Next, the decision tree algorithm was chosen, with max depth set as 5, 10, 20, and one None value where we don't specify the max depth and let the algorithm decide.
- Also sample split values were chosen: 2, 5 and 10 for observing the behaviour for different splits.

Output

DT (max_depth=5, min_samples_split=2) test accuracy: 0.6735

DT (max_depth=5, min_samples_split=5) test accuracy: 0.6735

DT (max_depth=5, min_samples_split=10) test accuracy: 0.6735

DT (max_depth=10, min_samples_split=2) test accuracy: 0.8110

DT (max_depth=10, min_samples_split=5) test accuracy: 0.8105

DT (max_depth=10, min_samples_split=10) test accuracy: 0.8070

DT (max_depth=20, min_samples_split=2) test accuracy: 0.8035

DT (max_depth=20, min_samples_split=5) test accuracy: 0.8090

DT (max_depth=20, min_samples_split=10) test accuracy: 0.8050

DT (max_depth=None, min_samples_split=2) test accuracy: 0.8115

DT (max_depth=None, min_samples_split=5) test accuracy: 0.8115

DT (max_depth=None, min_samples_split=10) test accuracy: 0.8025

Best Decision Tree -> {'max_depth': None, 'min_samples_split': 2}, test accuracy=0.8115

Observation: This shows that model performs best with max_depth=None and sample split = 2.

➢ Then the models were checked on testing data to see how they perform individually

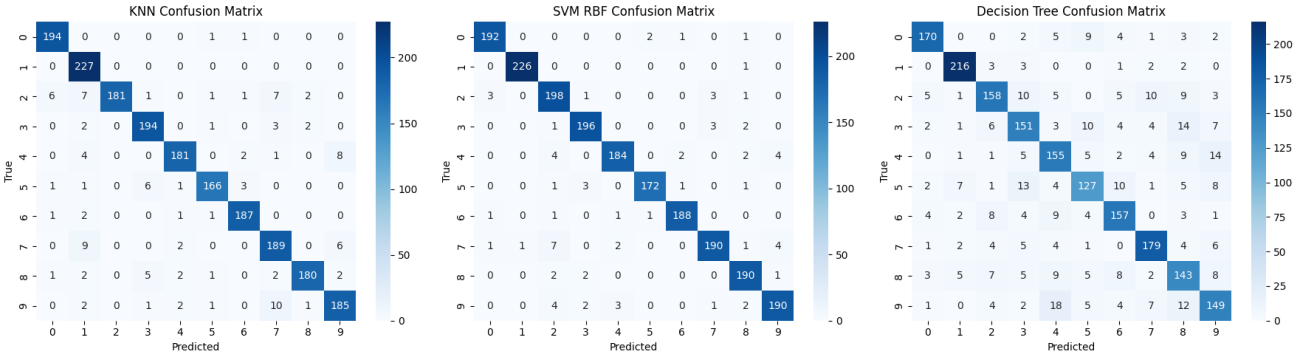
Output

Test Set Accuracies

KNN: 0.9420 (94.20%)

SVM RBF: 0.9630 (96.30%)

Decision Tree: 0.8025 (80.25%)

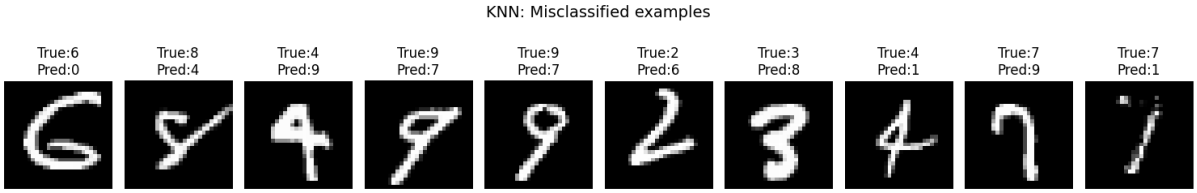


Observation: Tells that the best performing model was SVM with test accuracy 96.30%. It can be due to the fact that SVM is a more complex algorithm which can accurately identify and predict the complex patterns, especially due to powerful tools like RBF kernels. Even the confusion matrix depicts that most digits were predicted accurately, with certain exceptions. Decision tree was the worst performing model out of the 3 models.

➢ The first 10 misclassified images were printed along with their true and predicted values.

Output and Observations

KNN



Observation: In the first image, the curve structure is similar in 6 and 0, and the loop at the end is also not distinctly defined.

In the next image, 4 is printed because the two loops of 8 are not clearly visible and the top loop is also open, which makes the model think that it is 3 lines (in case of 4) rather than two loops (in case of 8)

in the next image, the curve is slightly pointed, making it look like 3 sharp lines, rather that a loop and one vertical line.

in the next image, the model mistook the slightly slanted line which is in made in 7 and considered the top loop of 9 as a horizontal line which is curved at one end

Same happened in the next case

in the next image the top part of 2 is not fully curved making the model believe that it is a slightly tilted line which is usually in 6

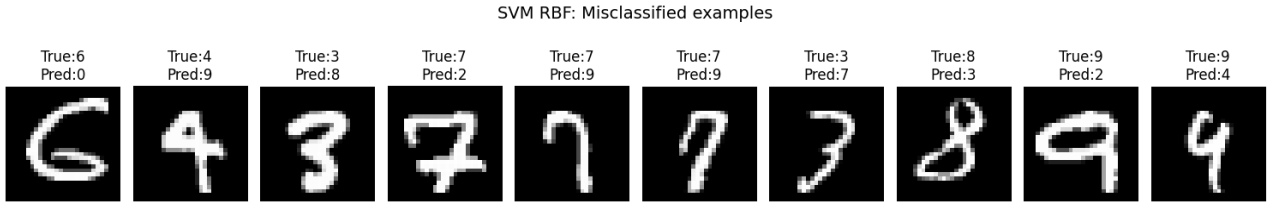
The number 3 and 8 are very similar so the model got confused with the two shapes

the slanted and horizontal line was not properly defined, making the model confused and treating those lines as noise

in the next image, the sharpness of the horizontal line and slanted line in 7 is not distinct so the model treats it like a loop and straight line, thus predicting 9

in the last image, the horizontal line of 7 is not clearly visible, so the model thinks it is a 1.

SVM



Observation: In the first image, the curve structure is similar in 6 and 0, and the loop at the end is also not distinctly defined.

In the next image, the sharpness of the slanted and horizontal line of 4 is not very distinct to the model, hence classifying it as 9 instead of 4

The number 3 and 8 are very similar so the model got confused with the two shapes

the model treats the top horizontal line of 7 as a curve and the below line as the line which is in the bottom of the number 2

in the next image, the sharpness of the horizontal line and slanted line in 7 is not distinct so the model treats it like a loop and straight line, thus predicting 9

same in the next image

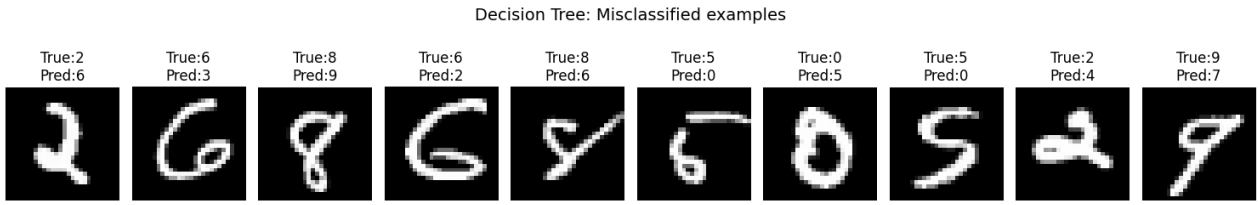
in the next image, the two curves of 3 are not properly written, making the model think it is a slanted line, hence it predicts 7

in the next image, again 3 and 8 have similar shapes so the model confused the two shapes

the model treats the loop of 9 as a curved line of 2 as the loop is not a proper circle

the top part of the loop is not closed, making the model think that it the 3 lines which are there in 4 rather than a loop and a straight line

Decision Tree



Observation: in these cases we can see that the decision tree has not learnt the curve or the patterns in the image properly. it is predicting based on the most likely values which will be present in the 28x28 matrix and guessing based on the assumptions provided. hence, decision tree was not a good algorithm for image data classification in the mnist digit dataset.

- Next, ensemble methods were applied, on the model with the best hyperparameters evaluated at the end of task 3, their individual accuracies along with accuracy of hard and soft voting ensemble techniques were evaluated.

Output

Individual accuracies:

KNN: 0.9420

SVM: 0.9630

DT: 0.8025

Ensemble -> Hard: 0.9600, Soft: 0.9610

Observation: Even though ensemble techniques were effective and give a good accuracy of 96%, yet SVM model has a slightly higher accuracy making it the best model so far.

- Next the models scores were evaluated after using PCA for SVM and KNN with the number of components assumed to be 50, reducing the data dimensionality from 784D to only 50D.

Output

PCA Effect (50 components)

KNN (784D): 0.942 → PCA (50D): 0.9635

SVM (784D): 0.963 → PCA (50D): 0.9685

PCA explained variance: 83.1%

Observation: Both KNN and SVM had an improve in their accuracy by reducing the dimensionality. The PCA also has a explained variance of 83.1% which is good and means that it captures 83.1% variability in the dataset even after reducing the dimensions

- A final comparison of all the models observed along with their PCA dimensionality was printed

Output

	Model	784D	PCA-50D
0	KNN	0.942	0.9635
1	SVM	0.963	0.9685
2	Ensemble Soft	0.961	NaN

Observation: It shows that, the best performing model was SVM with 50D dimensionality.