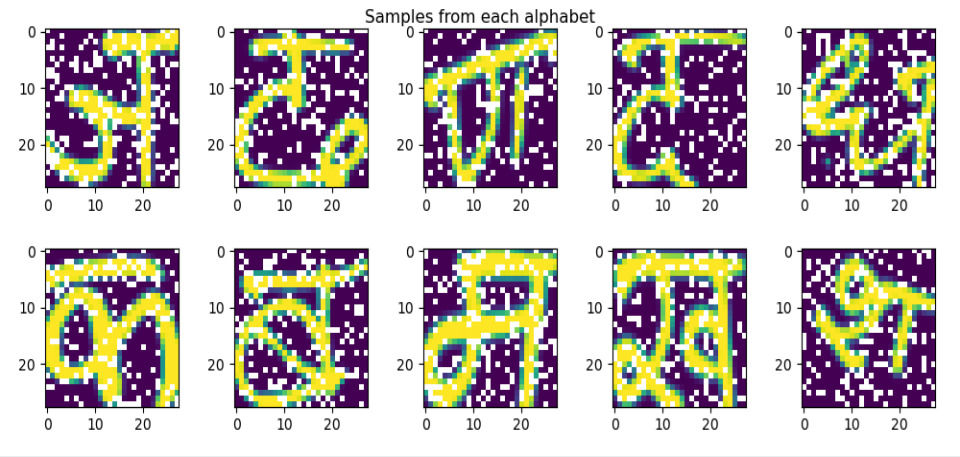
Image classification for Devanagari script

Sujay Bokil - ME17B120
Avinash Bagali - AE17B110

Problem Statement



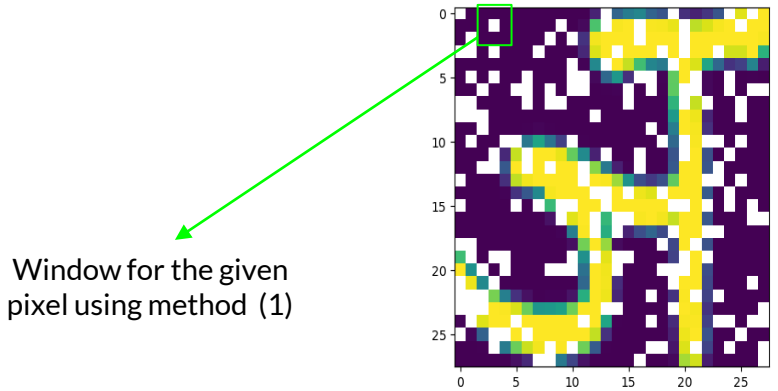
Our goal is to identify images containing alphabets of Devanagari script which is a classification task.

No. of alphabets: 10

Samples per alphabet: 1000

Dataset size is 10 X 1000 = 10000 samples

Missing value imputation



1. Mean imputation from surrounding pixels:

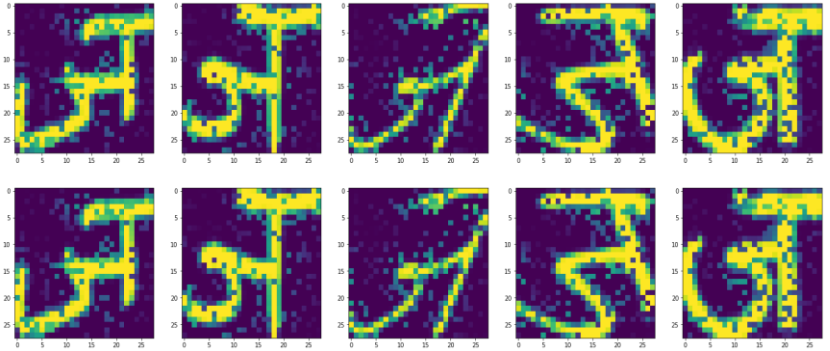
   A 3 X 3 pixel window is considered around every pixel containing NaN and the mean of neighbouring pixels is imputed

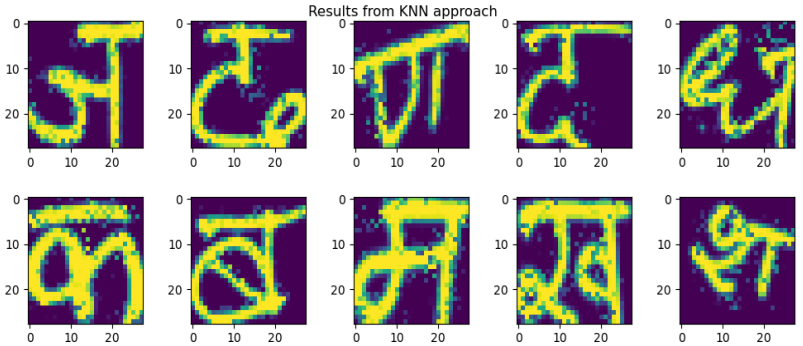
1. Imputation using other images
2. K-Nearest Neighbours imputation

Missing value imputation

Method (2) is scrapped because of the noise observed visually in imputed images

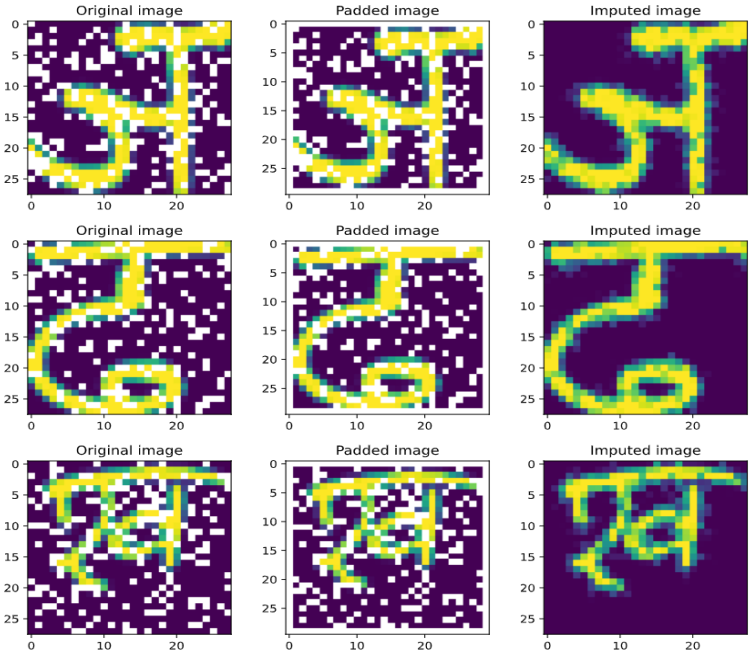
KNN imputation also provides decent results





Best solution (Preprocessing)

After performing basic modelling and validation on both the KNN imputed dataset and the dataset imputed through method (1), we concluded that **method (1) showed better results** both in terms of the validation scores as well as visual observation of images themselves



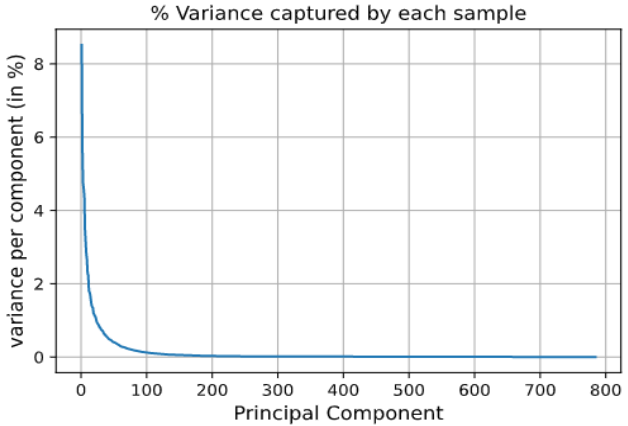
We pad the images before performing NaN imputation to avoid handling corner cases separately

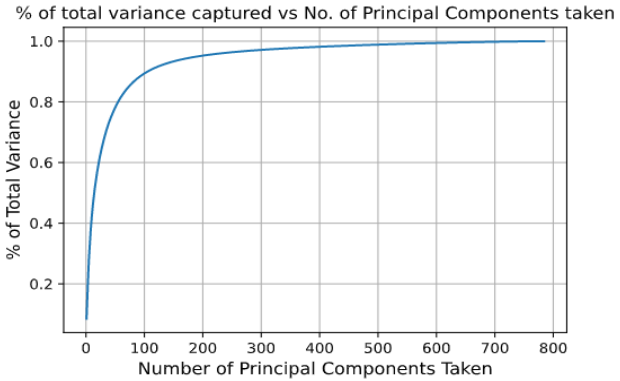
PCA and Model selection

We use PCA with 100 components to strike a good trade off between reduction in feature size and capturing most information. It captures **89%** of the variance

Then we tried the following models on the transformed dataset .

1)SVM
2)Logistic Regression
4)KNN
3)Gaussian Naive Bayes
5)Random Forest





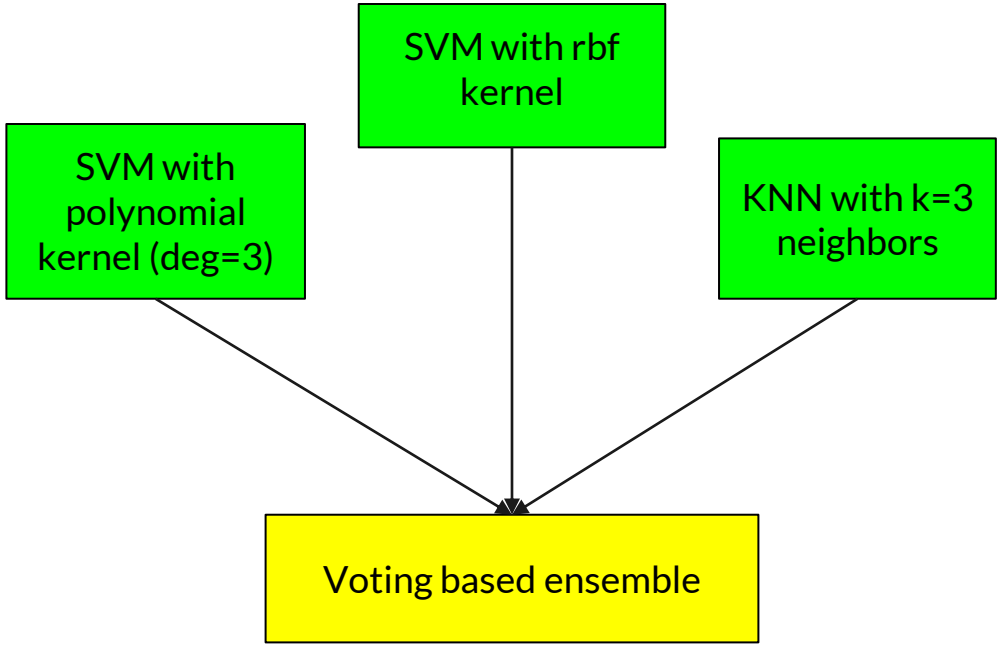
Models

Model	Parameters	Validation scores
Naive Bayes	---	Accuracy: 0.7915
Random Forest	Estimators = 100 , max_depth =11, ccp_aplha=0.004	Accuracy: 0.884
Log Regression	Max_iter = 10000 Penalty = 'l2'	Accuracy: 0.8275
SVM (poly)	kernel = 'poly' ;degree =3 ; gamma = 2.1e-07 ; C = 1	Accuracy: 0.96
SVM (rbf)	kernel = 'rbf' ; gamma = 3.2e-07 ; C = 2	Accuracy: 0.9665
KNN	n_neighbors = 3 weights = 'distance'	Accuracy: 0.956

Best Solution (modelling)

After imputation we reduce the dataset using PCA with n=100 components

We use an ensemble of the 3 separately tuned models with weights (1,1,1) to each model which performs the best



Results

Model	Parameters	Validation scores
SVM (poly)	kernel = ‘poly’ gamma = 2.1e-07 degree =3 C = 1	Accuracy: 0.96, F1 score: 0.9600
SVM (rbf)	kernel = ‘rbf’ gamma = 3.2e-07 C = 2	Accuracy: 0.9665, F1 score: 0.9664
KNN	n_neighbors = 3 weights = ‘distance’	Accuracy: 0.956, F1 score: 0.9559
Ensemble	(SVM_poly: 1, SVM_rbf: 1, KNN: 1)	Accuracy: 0.977, F1 score: 0.9769

Conclusion

- Method (2) of imputation fails because it isn't robust against different orientations and stretching of the characters. Also, it is data dependent instead of sample dependent.
- Images contain a lot of features usually compared to tabulated data so compressing them for feature extraction is a must to save training time.
- Models like logistic regression and Decision Trees are not suitable due to a high number of features making them susceptible to high bias and high variance respectively.
- The voting ensemble makes the models cover each others mistakes on their respective weak portions of the dataset eventually leading to better scores.

References

1. SVM
2. KNN
3. Parameter tuning
4. KNNImputer
5. Validation