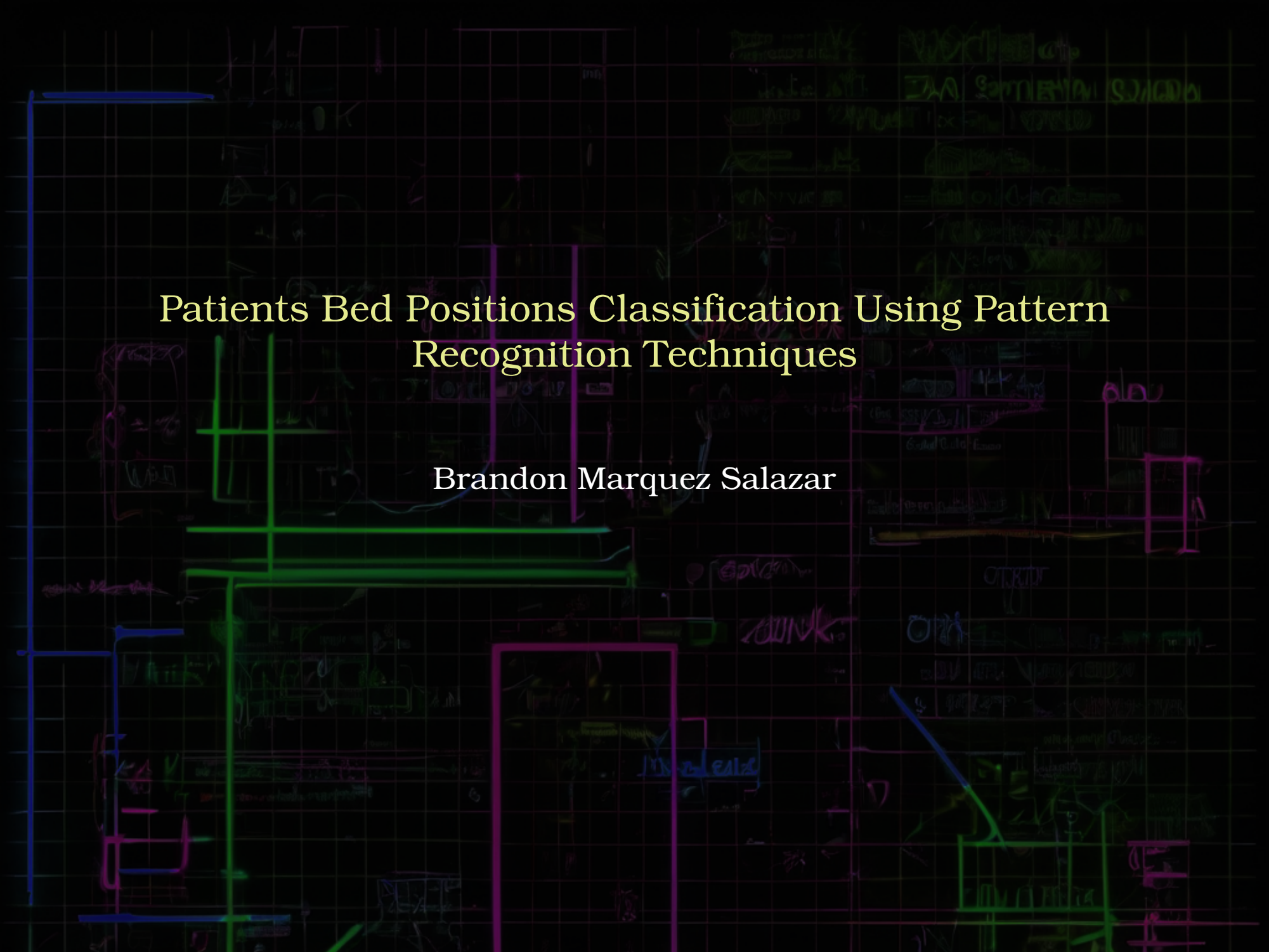


Patients Bed Positions Classification Using Pattern Recognition Techniques

Brandon Marquez Salazar

Objetive

The objective of this experiment is to test pattern recognition techniques—LBP in this experiment—and classifiers—such as Decision Trees, Bayes Naïve, KNN and SVM—. Also, testing whether a good data preprocessing—data augmentation, normalization, balancing, etc.—is required for a better classifier performance.

Methods

In this experiment, the methodology is described below

- First version
  - Data description
  - Data loading
  - Data balancing by undersampling
  - Feature extraction via LBP
  - Model selection via Grid Search Cross Validation
  - Get the best models metrics

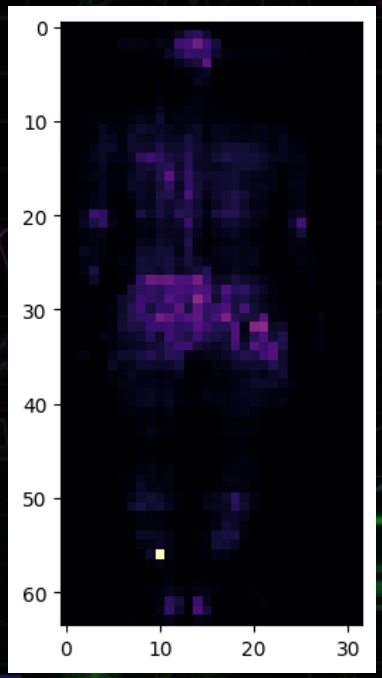
Methods

In this experiment, the methodology is described below

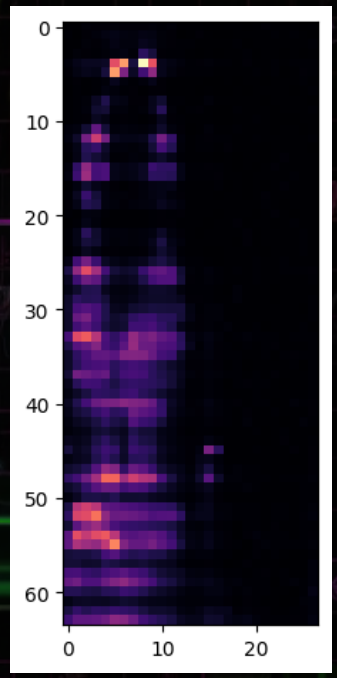
- Second version
  - Data description
  - Data loading
  - Data balancing by undersampling
  - Data augmentation using shifting
  - Feature extraction via LBP
  - Model selection via Grid Search Cross Validation
  - Get the best models metrics

The samples

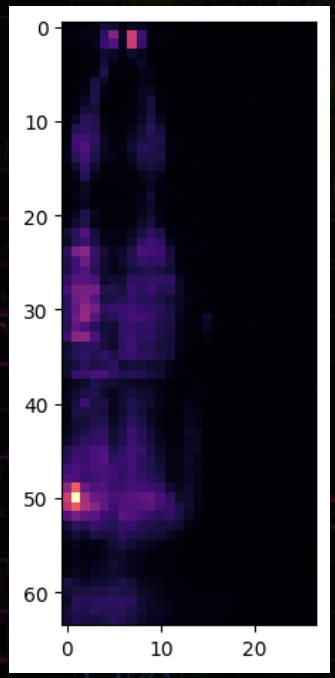
The samples are arrays of pressure levels wich can be interpreted as images in 1channel. The documentation gives the dimensions for the image.



(a) Exp. I, Sam. 1/F.45



(b) Exp. II, Sam. B1/AirMat



(c) Exp. II, Sam. B1/SpongeMat

Class	Number of Samples
Supine	10,613
Right	4,672
Left	4,739
Total	**20,024**

Table: Class distribution for Exp. I

Class	Number of Samples
Supine	270
Right	144
Left	48
Total	**462**

Table: Class distribution for Exp. II

Data Balancing and Augmentation

For balancing, the method used was undedrsampling which looks takes the cardinalities per class $|\mathcal{C}_i|$ and uses the smallest one $N_{\min}$, then, randomizes the samples and piks $N_{\min}$ samples from each class, getting the new dataset $\mathcal{S}$.

$$\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_n\}$$

$$N_{\min} = \min_{\mathcal{C}_i \in \mathcal{C}} |\mathcal{C}_i|$$

$$f : \mathcal{P}(\mathcal{C}) \times \mathbb{N} \to \mathcal{P}(\mathcal{C})$$

$$\mathcal{S} = \bigcup_{\mathcal{C}_i \in \mathcal{C}} f(\mathcal{C}_i, N_{\min})$$

Data Balancing and Augmentation

Then, for the second version, it was used a data augmentation technique which generates new samples by shifting the pixels of the original ones. Here, the shifting was made in 4 px—upside and downside, leftside and rightside—for x and y. With a non destructive shifting.

$$\text{shift} : \mathbb{R}^{m \times n} \times \mathbb{Z} \times \mathbb{Z} \to \mathbb{R}^{m \times n}$$

$$M_{i+N_{\min}} = \text{shift}(M_i, x, y)$$

$$M_{i+N_{\min}+1} = \text{shift}(M_i, -x, -y)$$

Feature extraction

Here, the feature extraction was done via Local Binary Patterns (LBP) which is easily defined by

$$\mathrm{LBP}_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} 2^p \cdot [g_p \geq g_c] \quad \text{where} \quad s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The LBP was computed for each sample without windowing (windowing each sample caused intensive RAM usage).

Model selection for classification I

For this, the approach given was Grid Search Cross Validation (GSCV) provided by scikit-learn library. Here, the main parameters used were:

Table: Block 1: Core classifiers and their hyperparameter search grids.

Classifier	**Hyperparameters**	**Search Values**
Decision Tree	max_depth min_samples_split criterion	[3, 5, 7, 10] [2, 5, 10] ['gini', 'entropy']
Random Forest	n_estimators max_depth min_samples_split	[50, 100, 200] [3, 5, 7, 10] [2, 5, 10]
KNN Coarse	n_neighbors weights	[10, 30, 50, 70, 90] ['uniform', 'distance']
AdaBoost	n_estimators learning_rate	[50, 100, 200] [0.01, 0.1, 1.0]

Model selection for classification II

Table: Block 2: K-Nearest Neighbors variants and their hyperparameter search grids.

Classifier	**Hyperparameters**	**Search Values**
KNN Fine	n_neighbors weights	[1, 2, 3] ['uniform', 'distance']
KNN Minkowski	n_neighbors p (Minkowski param.) weights	[5, 10, 15] [1, 2, 3] ['uniform', 'distance']
KNN Weighted	n_neighbors weights	[5, 10, 15] ['distance']
KNN Medium	n_neighbors weights	[10, 15, 20, 25] ['uniform', 'distance']

Table: Block 3: Probabilistic classifiers and their hyperparameter search grids.

Classifier	**Hyperparameters**	**Search Values**
Naive Bayes (Gaussian)	var_smoothing	$[1 \times 10^{-9}, 1 \times 10^{-8}, 1 \times 10^{-7}, 1 \times 10^{-6}]$
LDA	solver shrinkage	['svd', 'lsqr', 'eigen'] ['auto', 0.1, 0.5, 0.9]
KNN Cosine	n_neighbors weights	[5, 10, 15] ['uniform', 'distance']

Model selection for classification III

Table: Block 4: Support Vector Machine classifiers and their hyperparameter search grids.

Classifier	**Hyperparameters**	**Search Values**
Linear SVM	C (Regularization) kernel	[0.1, 1, 10, 100] ['linear']
Quadratic SVM	C (Regularization) degree gamma	[0.1, 1, 10] [2] ['scale', 'auto']
Cubic SVM	C (Regularization) degree gamma	[0.1, 1, 10] [3] ['scale', 'auto']
Fifth SVM	C (Regularization) degree gamma	[0.1, 1, 10] [5] ['scale', 'auto']

Experimental Results Summary

Performance metrics across all blocks and experiments

Experiment	Best Model	Acc.	Prec.	Rec.	F1	AUC Score	CV Score
Experiment I							
Block 1: Tree-Based	KNN Coarse	0.8734	0.8733	0.8734	0.8733	0.9676	0.8639
Block 2: KNN Variants	KNN Medium	0.8734	0.8733	0.8734	0.8733	0.9676	0.8639
Block 3: Probabilistic	KNN Cosine	0.8146	0.8156	0.8146	0.8144	0.9439	0.8160
Block 4: SVM Variants	Quadratic SVM	0.6555	0.6612	0.6555	0.6460	N/A	0.6540
Experiment II							
Block 1: Tree-Based	Decision Tree	0.5172	0.5157	0.5172	0.4937	0.6558	0.5391
Block 2: KNN Variants	KNN Fine	0.5172	0.5640	0.5172	0.4875	0.6849	0.5304
Block 3: Probabilistic	KNN Cosine	0.5517	0.5552	0.5517	0.5526	0.7247	0.4609
Block 4: SVM Variants	Linear SVM	0.5517	0.5699	0.5517	0.5507	N/A	0.4348

Note: Acc. = Accuracy, Prec. = Precision, Rec. = Recall, CV = Cross-Validation.
N/A indicates the metric was not available for that model.

Impact of Data Augmentation on Model Performance

Comparative results before and after augmentation

Experiment	Best Model	Acc.	Prec.	Rec.	F1	AUC Score	CV Score
Without Data Augmentation (Experiment I)							
Tree-Based	KNN Coarse	0.8730	0.8730	0.8730	0.8730	0.9680	0.8640
KNN Variants	KNN Medium	0.8730	0.8730	0.8730	0.8730	0.9680	0.8640
Probabilistic	KNN Cosine	0.8140	0.8150	0.8140	0.8140	0.9440	0.8160
SVM Variants	Quadratic SVM	0.6550	0.6610	0.6550	0.6460	N/A	0.6540
With Data Augmentation (Experiment II)							
Tree-Based	Decision Tree	0.5170	0.5160	0.5170	0.4940	0.6560	0.5390
KNN Variants	KNN Fine	0.5170	0.5640	0.5170	0.4870	0.6850	0.5300
Probabilistic	KNN Cosine	0.5520	0.5550	0.5520	0.5530	0.7250	0.4610
SVM Variants	Linear SVM	0.5520	0.5700	0.5520	0.5510	N/A	0.4350

Note: Acc. = Accuracy, Prec. = Precision, Rec. = Recall, CV = Cross-Validation Score. Results show significant performance difference between augmented and non-augmented datasets.

Block 1: Tree-Based Classifiers (Experiment I)

Best Model: KNN Coarse

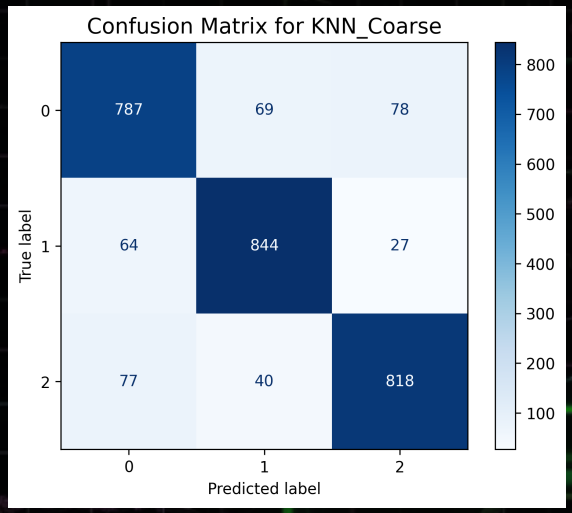


Figure: Confusion Matrix

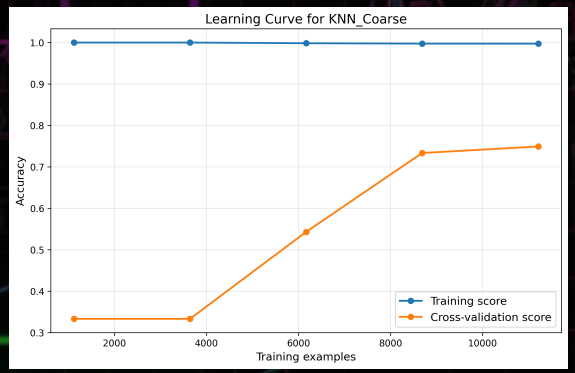


Figure: Learning Curve

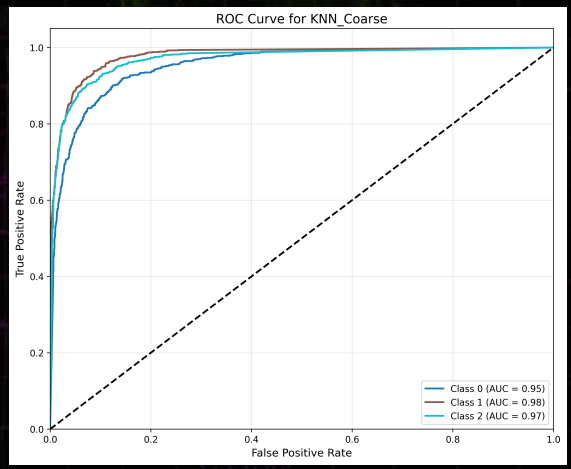


Figure: ROC Curve

Block 2: KNN Variants (Experiment I)

Best Model: KNN Medium

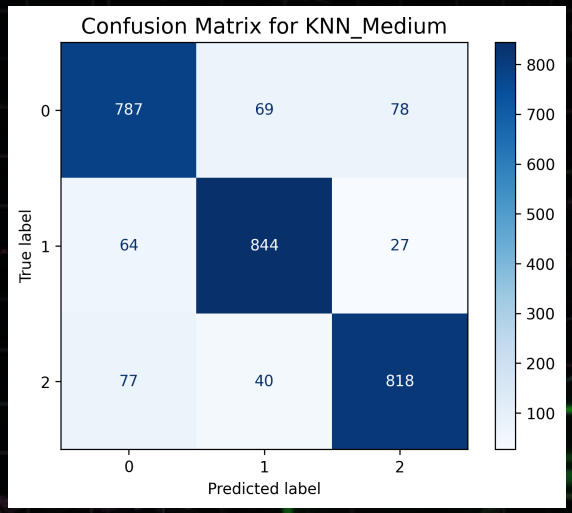


Figure: Confusion Matrix

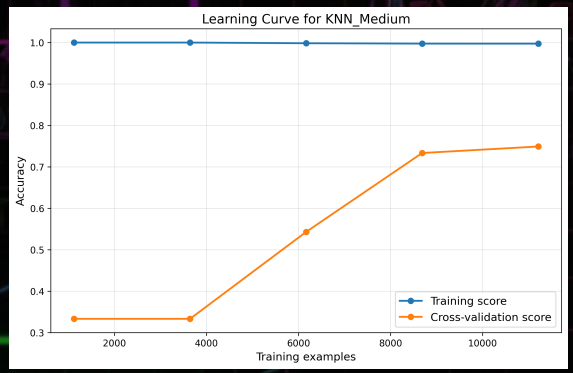


Figure: Learning Curve

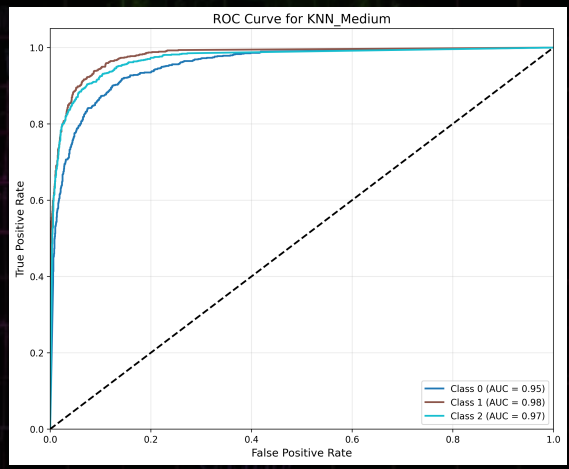


Figure: ROC Curve

Block 3: Probabilistic Classifiers (Experiment I)

Best Model: KNN Cosine

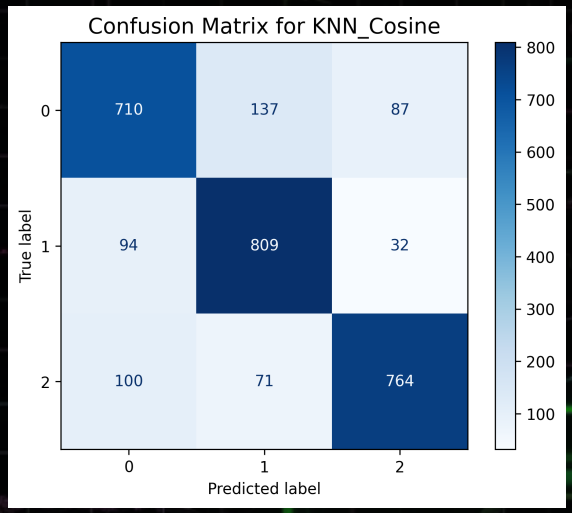


Figure: Confusion Matrix

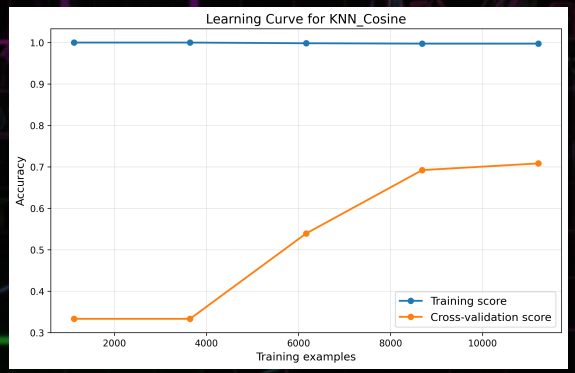


Figure: Learning Curve

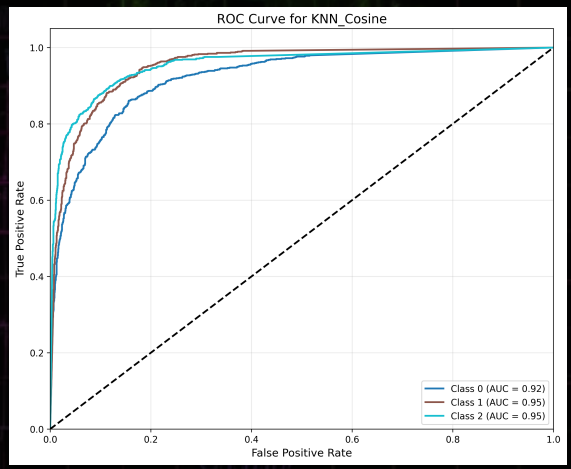


Figure: ROC Curve

Block 4: SVM Variants (Experiment I)

Best Model: Quadratic SVM

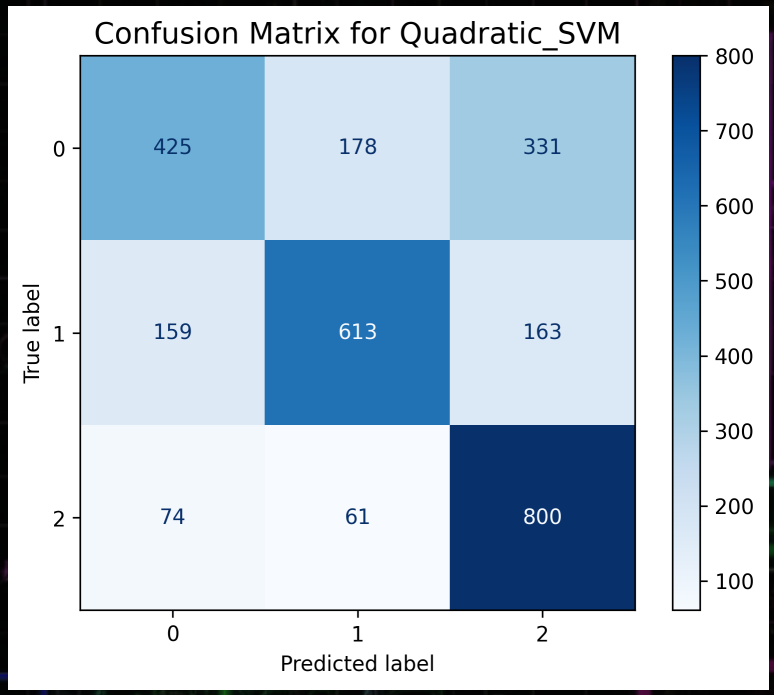


Figure: Confusion Matrix

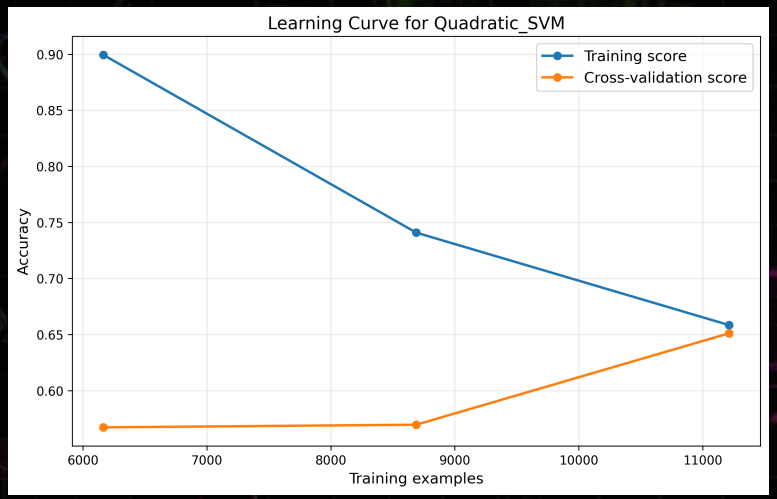


Figure: Learning Curve

Block 1: Tree-Based Classifiers (Experiment II)

Best Model: Random Forest

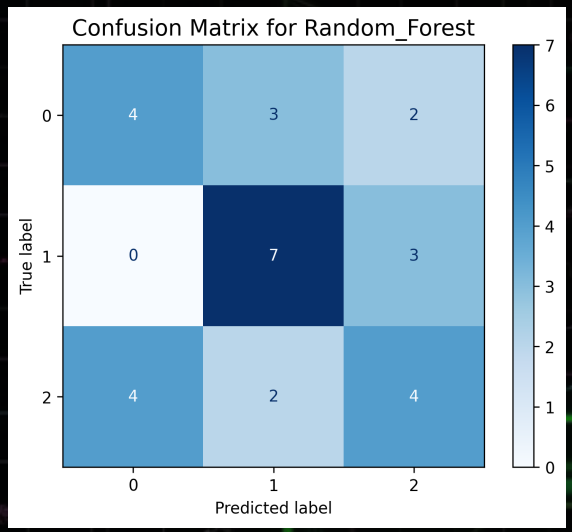


Figure: Confusion Matrix

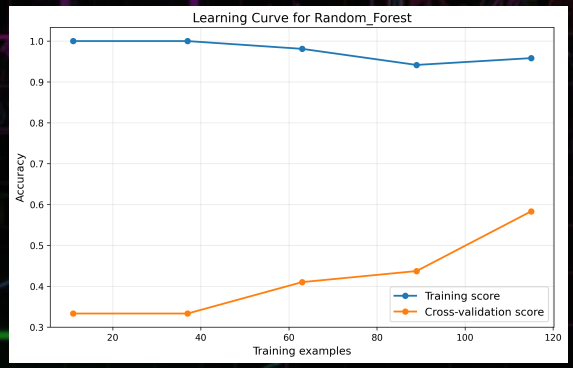


Figure: Learning Curve

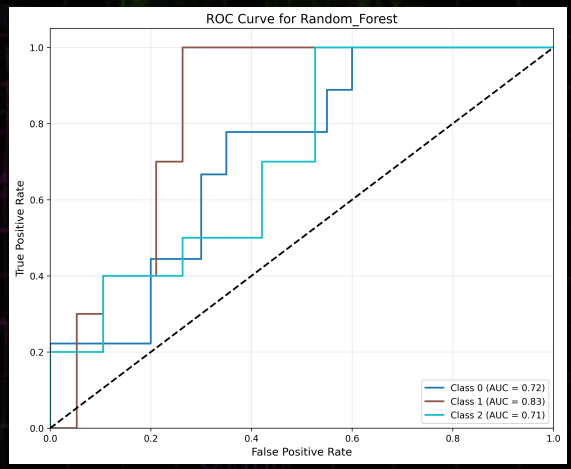


Figure: ROC Curve

Block 2: KNN Variants (Experiment II)

Best Model: KNN Fine

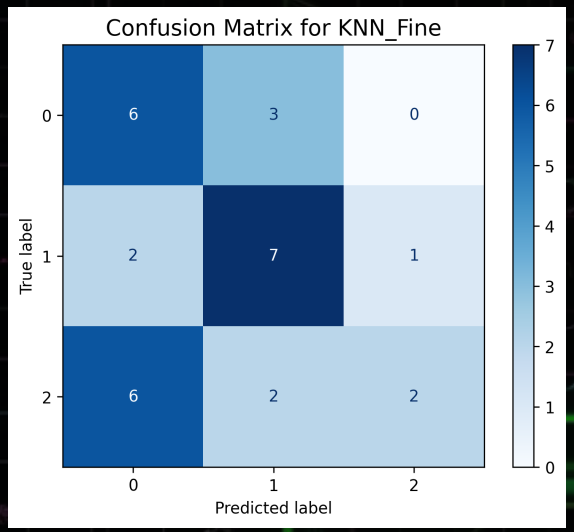


Figure: Confusion Matrix

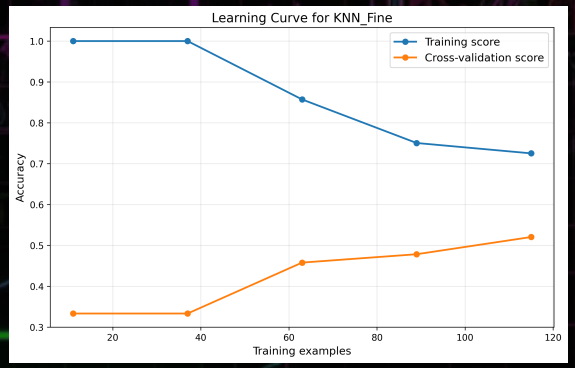


Figure: Learning Curve

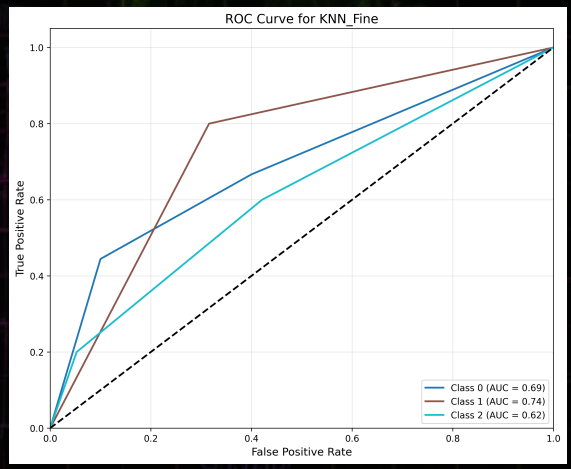


Figure: ROC Curve

Block 3: Probabilistic Classifiers (Experiment II)

Best Model: KNN Cosine

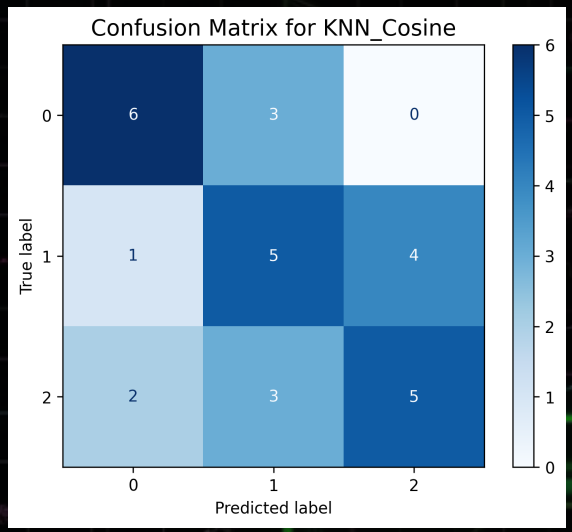


Figure: Confusion Matrix

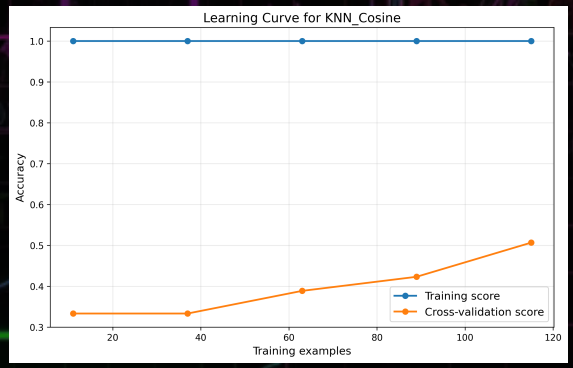


Figure: Learning Curve

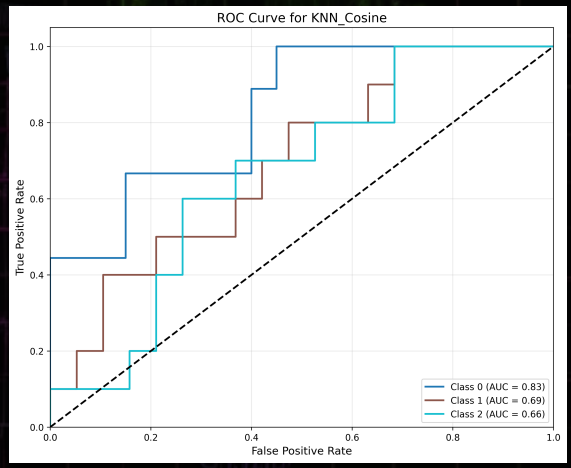


Figure: ROC Curve

Block 4: SVM Variants (Experiment II)

Best Model: Linear SVM

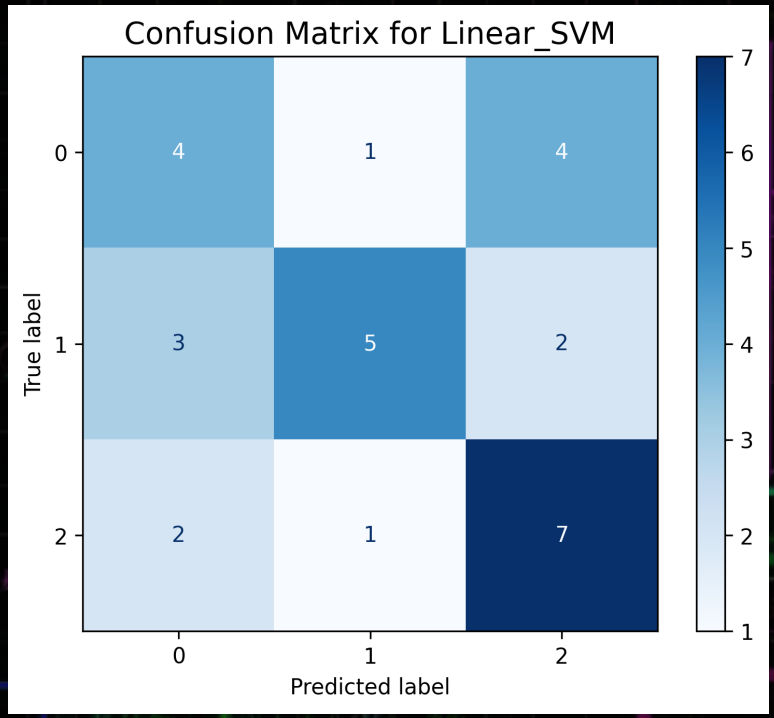


Figure: Confusion Matrix

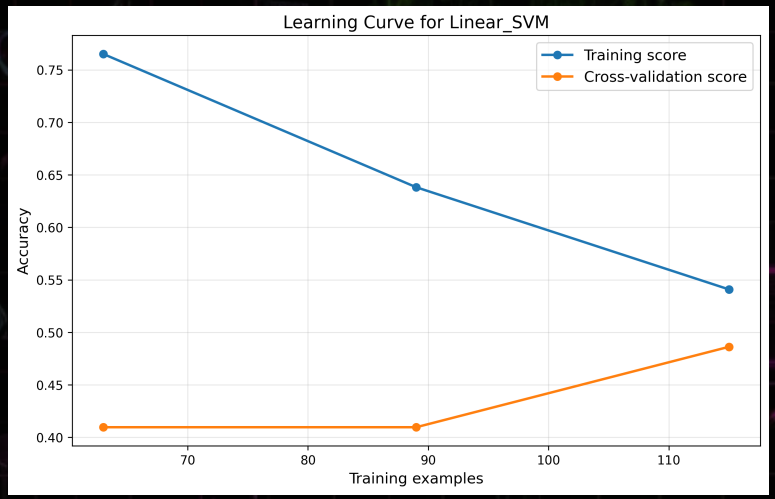
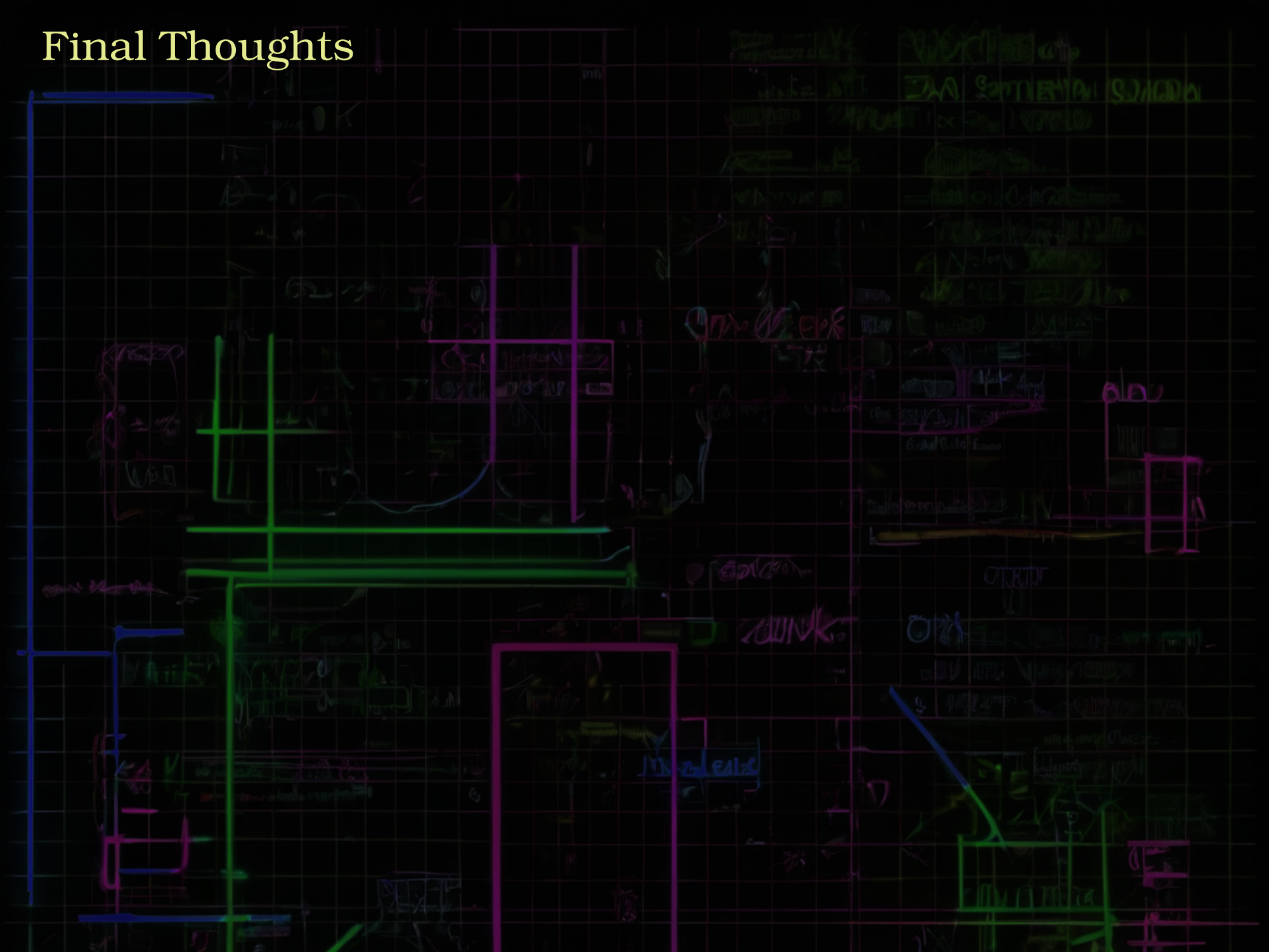


Figure: Learning Curve

Final Thoughts



Thanks