End sem

November 15, 2025

1 Implementation of logistic regression for a multi class problem

- First, I scaled the values by dividing by 255
- And then implemented the Multinomial Logistic regression code
- On average, the weighted f1 score i got for logistic regression is 0.90 with 1000 epochs and learning rate 0.1, and it took approximately 59.19 seconds for execution

1.1 Hyperparameter tuning

Epochs	LR	Accuracy (test)	Accuracy (train)	Weighted F1 (test)	Weighted F1 (train)	Time
100	0.1	0.8595	0.8562	0.8584	0.8553	5.76 s
200	0.1	0.8752	0.8746	0.8744	0.8740	16.91 s
500	0.1	0.8892	0.8940	0.8886	0.8936	34.13 s
1000	0.1	0.9004	0.9074	0.8999	0.9071	58.83 s
2000	0.1	0.9088	0.9184	0.9084	0.9181	119.03 s

Table 1: Model performance across different epoch settings

Epochs	LR	Accuracy (test)	Accuracy (train)	Weighted F1 (test)	Weighted F1 (train)	Time
1000	0.01	0.8619	0.8574	0.8608	0.8565	60.42 s
1000	0.02	0.8739	0.8746	0.8732	0.8740	58.21 s
1000	0.1	0.8996	0.9077	0.8992	0.9074	65.12 s
1000	0.2	0.9076	0.9185	0.9072	0.9183	58.32 s

Table 2: Performance comparison across different learning rates (1000 epochs)

1.2 Conclusions

- Approximately the weighted f1 score for 1000 and 2000 epcohs is same
- It is just that 2000 epochs took more time
- Similarly, the weighted f1 score for learning rates 0.1 and 0.2 is approximately the same
- But the time for the 0.2 learning rate is somewhat less
- As we can see, the time taken for each and every learning rate is approximately the same
- Without scaling/normalizing the values the f1 score is around 0.86 and hence I did normalization .(not including that code snippet in my submissions)
- **By changing the number of epochs to 2000 and the learning rate to 0.2, the best f1 score I got is 0.91288 and the best accuracy is 0.9132 in time -117.66 seconds .**
- And for this best model, the f1 score for training set is 0.9288.
- Epochs-1000 and learning rate -0.1 gave the weighted f1 score in the training dataset as 0.899 and in the test dataset 0.9074

1.3 Bias Variance Tradeoff Analysis

In this section, I analyzed how the model performance changes with respect to (i) the number of training epochs and (ii) the learning rate.

1.3.1 Effect of Number of Epochs

As the number of epochs increases from 100 to 2000 (learning rate fixed at 0.1), the training and test metrics consistently improve.

Bias

The models trained with fewer epochs (100, 200) show lower train and test F1 scores, indicating **higher bias**. The model has not fully learned the underlying patterns. As the epochs increase (500, 1000, 2000), the training loss continues to decrease, and performance improves reflecting a **reduction in bias**.

Variance

The gap between training and test performance remains very small across all epoch settings:

$$\text{Gap}_{100} \approx 0.0033, \quad \text{Gap}_{1000} \approx 0.0075, \quad \text{Gap}_{2000} \approx 0.0096.$$

Although the gap increases slightly at higher epochs, it remains small. This indicates that the increase in epochs improves representation learning without causing significant overfitting.

1.3.2 Effect of Learning Rate

Bias

Lower learning rates (0.01, 0.02) converge more slowly and result in lower train and test F1 scores, indicating **higher bias**. As we increase the learning rate to 0.1 and 0.2, both training and test F1 scores improve substantially, showing a **reduction in bias**.

Variance

The train test gaps remain small for all learning rates:

$$\text{Gap}_{0.01} \approx 0.001, \quad \text{Gap}_{0.1} \approx 0.008, \quad \text{Gap}_{0.2} \approx 0.011.$$

While the variance increases slightly at higher learning rates, the improvement in test F1 score suggests that the model still generalizes well and does not overfit severely.

1.3.3 Direct Comparison: 1000 vs 2000 Epochs

Comparison of the two strongest models:

$$\text{Model A: 1000 epochs, learning rate} = 0.1$$

$$\text{Model B: 2000 epochs, learning rate} = 0.2$$

- **Model A**

  $$\text{Train F1} = 0.9074, \quad \text{Test F1} = 0.8996$$

  Bias is moderate, and variance is low:

  $$\text{Gap} \approx 0.0078$$

- **Model B**

  $$\text{Train F1} = 0.9183, \quad \text{Test F1} = 0.9084$$

  Bias is significantly reduced, and variance remains small:

  $$\text{Gap} \approx 0.0099$$

Although Model B has a slightly larger train test gap, it achieves a clear improvement in both training and test metrics. This indicates a substantial reduction in bias with only a mild increase in variance.

1.3.4 Conclusion

Increasing the number of epochs and tuning the learning rate improves the model's ability to fit the data, thereby reducing bias. The variance increases slightly but remains well controlled, as indicated by the small train test gaps across configurations.

Overall, the model with **2000 epochs and learning rate = 0.2** offers the best bias variance tradeoff:

- Highest train and test F1 scores,
- Significant bias reduction,
- Acceptable variance and good generalization.

Thus, this configuration provides the most effective balance between underfitting and overfitting.

2 Implementation of logistic regression for a multi class problem with PCA

- First,I scaled the values by dividing by 255.
- And then did PCA By reducing components.
- On average ,the weighted f1 score I got for PCA+logistic regression is 0.9043 with 1000 epochs,learning rate 0.1,components 100 and it took approximately 20.01 seconds.

2.1 Hyperparameter tuning

Components	**Train Acc**	**Val Acc**	**Train F1**	**Val F1**	**Time (s)**
600	0.9050	0.9048	0.9046	0.9043	46.21
500	0.9048	0.9043	0.9044	0.9038	38.95
400	0.9048	0.9040	0.9044	0.9035	32.31
300	0.9045	0.9044	0.9041	0.9039	27.07
200	0.9045	0.9044	0.9041	0.9039	20.62
100	0.9016	0.9048	0.9012	0.9043	14.04
50	0.8936	0.8991	0.8932	0.8986	11.78

Table 3: Model Performance across different components with learning rate=0.1,number of epochs=1000

Epochs	**Train Acc**	**Val Acc**	**Train F1**	**Val F1**	**Time (s)**
100	0.8525	0.8613	0.8512	0.8598	2.32
200	0.8723	0.8800	0.8715	0.8790	3.86
500	0.8899	0.8958	0.8894	0.8952	6.34
1000	0.9018	0.9047	0.9014	0.9042	14.08
2000	0.9091	0.9104	0.9087	0.9100	46.41

Table 4: Softmax Regression Performance for PCA (100 Components) Across different Epochs with learning rate =0.1

Learning Rate	**Train Acc**	**Val Acc**	**Train F1**	**Val F1**	**Time (s)**
0.01	0.8533	0.8628	0.8519	0.8613	15.19
0.02	0.8716	0.8792	0.8708	0.8782	15.64
0.10	0.9016	0.9039	0.9012	0.9034	14.45
0.20	0.9094	0.9102	0.9090	0.9098	14.36

Table 5: Softmax Regression Performance for PCA (100 Components) Across Learning Rates with epochs=1000

2.2 Conclusions

- Approximately the f1 score for 600,500,400,300,200,100 components is similar .
- But the time taken for 100 components is less
- When used 100 components and 2000 epochs the f1 score increased to 0.91 compared to only Logistic regression.
- The F1 score for 0.1 and 0.2 learning rate is almost similar
- **By changing the number of epochs to 2000 ,learning rate to 0.2,number of components to 100 ,the best f1 score I got 0.9133**

2.3 Bias–Variance Tradeoff Analysis

In this section, we analyze how the model performance changes with respect to:

1. Number of PCA components,
2. Number of training epochs, and
3. Learning rate.

2.3.1 Effect of PCA Components

We varied the number of PCA components from 50 to 600, keeping epochs fixed at 1000 and learning rate at 0.1.

Bias When PCA components are high (600–300), both train and validation F1 scores are around 0.904–0.905, indicating low bias. Reducing components to 100 and 50 decreases train and validation scores (train F1 drops to 0.9012 and 0.8932, val F1 drops to 0.9043 and 0.8986), indicating **increased bias** due to loss of information in lower-dimensional representations.

Variance The train validation gaps remain small across all PCA component settings (gap $\approx$ 0.0006–0.005), indicating low variance and stable generalization.

2.3.2 Effect of Number of Epochs

We trained models with PCA=100 and learning rate=0.1 for 100, 200, 500, 1000, and 2000 epochs.

Bias Models with fewer epochs (100, 200) show lower F1 scores (train F1 $\approx$ 0.8512–0.8715, val F1 $\approx$ 0.8598–0.8790), indicating **higher bias**. Increasing epochs improves both train and validation metrics, reducing bias (train F1 reaches 0.9087, val F1 0.9100 at 2000 epochs).

Variance The train validation gap remains small:

$$\text{Gap}_{100} \approx 0.0084, \quad \text{Gap}_{2000} \approx 0.0013$$

indicating only a minor increase in variance with longer training.

2.3.3 Effect of Learning Rate

We trained models with PCA=100, epochs=1000, and varied learning rates (0.01, 0.02, 0.1, 0.2).

Bias Low learning rates (0.01, 0.02) lead to lower train and validation F1 scores (train F1 0.8519–0.8708, val F1 0.8613–0.8782), reflecting **high bias**. Increasing the learning rate to 0.1 and 0.2 improves metrics substantially (train F1 0.9012–0.9090, val F1 0.9034–0.9098), reducing bias.

Variance Train–validation gaps are small but slightly increase with higher learning rate:

$$\text{Gap}_{0.01} \approx 0.009, \quad \text{Gap}_{0.2} \approx 0.0092$$

This shows that variance increases only marginally while improving performance.

2.3.4 Direct Comparison of Strongest Models

Two strongest configurations:

- **Model A: PCA=100, epochs=1000, lr=0.1**

$$\text{Train F1} = 0.9014, \quad \text{Val F1} = 0.9043, \quad \text{Gap} \approx 0.0029$$

- **Model B: PCA=100, epochs=2000, lr=0.2**

$$\text{Train F1} = 0.9163, \quad \text{Val F1} = 0.9133, \quad \text{Gap} \approx 0.0030$$

Model B reduces bias substantially while the increase in variance remains minimal.

2.3.5 Conclusion

- Reducing PCA components below 100 increases bias, while higher components retain information without increasing variance.
- Increasing epochs and learning rate consistently reduces bias.
- Train validation gaps remain small, indicating controlled variance.
- The best tradeoff is achieved with **PCA=100, epochs=2000, learning rate=0.2**, which maximizes F1 scores while keeping variance low.

This configuration provides the optimal balance between underfitting and overfitting.

3 Implementation of KNN with PCA

- On reducing number of components to 100 keeping k=5 training accuracy i got is 0.9675 and validation accuracy -0.9520.
- Training F1 score -0.9674 and Validation F1 score is 0.9519

3.1 Hyperparameter tuning

k	Train Acc	Val Acc	Train F1	Val F1	Time (s)
3	0.9755	0.9508	0.9755	0.9505	50.18
5	0.9675	0.9520	0.9674	0.9519	48.31
10	0.9558	0.9476	0.9557	0.9473	50.68
20	0.9410	0.9380	0.9409	0.9377	49.03
30	0.9329	0.9296	0.9329	0.9294	48.51

Table 6: KNN + PCA results for different k values,Components=100

3.2 Conclusions

- As we can see that for k=5 we got best f1 score both for training and validation datasets
- As we increase k the f1 score is decreasing along with accuracy .
- On decreasing number of components to 100 we got best F1 score
- Also F1 score for 200 and 100 components is same despite of their different but close accuracy values
- On decreasing number of components to 50 F1 score is reducing to approximately 92

PCA Components	Train Acc	Val Acc	Train F1	Val F1	Time (s)
784	0.9624	0.9504	0.9623	0.9502	654.94
600	0.9624	0.9504	0.9623	0.9502	508.52
500	0.9624	0.9500	0.9623	0.9498	440.21
400	0.9624	0.9516	0.9623	0.9514	337.17
300	0.9634	0.9512	0.9633	0.9510	164.80
200	0.9652	0.9520	0.9651	0.9519	108.92
100	0.9675	0.9520	0.9674	0.9519	51.12

Table 7: KNN + PCA results for $k = 5$ with varying number of PCA components.

3.3 Bias–Variance Tradeoff Analysis

In this section, we analyze how the KNN + PCA model performance changes with respect to:

1. Number of PCA components,
2. Number of neighbors k in KNN.

3.3.1 Effect of Number of Neighbors k (with PCA=100)

We fixed PCA components at 100 and varied k from 3 to 30.

Bias Lower k values (3, 5) show higher train F1 scores (0.9755, 0.9674) and slightly lower validation F1 (0.9505, 0.9519), indicating **low bias**. Increasing k to 10, 20, 30 reduces both train and validation F1 (train F1 0.9558–0.9329, val F1 0.9476–0.9296), showing **slightly increased bias** as the model becomes smoother and less flexible.

Variance The gap between train and validation F1 decreases with higher k:

$$\text{Gap}_{k=3} \approx 0.0250, \quad \text{Gap}_{k=30} \approx 0.0035$$

This indicates that variance decreases as k increases, because the model relies on more neighbors and generalizes better.

3.3.2 Effect of PCA Components (with $k = 5$)

We fixed $k = 5$ and varied PCA components from 100 to 784.

Bias High-dimensional representations (784, 600, 500) show train F1 $\approx$ 0.9623 and val F1 $\approx$ 0.9502–0.9500, indicating **low bias**. Reducing PCA components to 100 slightly improves train F1 (0.9674) and val F1 (0.9519), showing that moderate dimensionality reduction preserves most information.

Variance Train-validation gaps remain small across all PCA components (gap $\approx$ 0.0121–0.0155), indicating controlled variance.

3.3.3 Best Configuration

The best configuration is:

- **PCA=100, k=5**

$$\text{Train F1} = 0.9674, \quad \text{Val F1} = 0.9519, \quad \text{Gap} \approx 0.0155$$

This configuration balances bias and variance well: low bias from sufficient neighbors and informative PCA components, and moderate variance that does not hurt validation performance.

3.3.4 Conclusion

- Lower k values give high variance, higher k values increase bias but reduce variance.
- PCA components above 100 retain most information, reducing bias without significantly increasing variance.
- The optimal tradeoff is achieved with **PCA=100, k=5**, maximizing F1 scores while keeping variance controlled.

4 Implementation of XGBoost

- For n-estimators=50,learning rate=0.1,maxdepth=3 the f1 score i got is 0.9129
- but for XGBOOST it is taking soo much time for training model .
- It is taking more than 30 minutes for hyperparameter tuning hence i am not including those values here .
- To keep training time ¡5 mins i used PCA with 100 components and i got f1 score to be around 0.88 and validation set accuracy to be around 0.8832.
- On hypertuning parameters after PCA we can get a much better accuracy .

PCA Comp	Train Acc	Val Acc	Train F1	Val F1	Time (s)	LR	Max D	Estimators
50	0.9149	0.8820	0.9147	0.8817	90.43	0.1	3	50
100	0.9169	0.8832	0.9167	0.8830	90.41	0.1	3	50
200	0.9156	0.8856	0.9155	0.8856	90.39	0.1	3	50
300	0.9159	0.8812	0.9157	0.8809	90.35	0.1	3	50

Table 8: XGBoost (Multi-class) performance for different PCA components.

Estimators	PCA Comp	LR	Max D	Train Acc	Val Acc	Train F1	Val F1	Time (s)
100	100	0.1	3	0.9630	0.9120	0.9630	0.9117	181.56
150	100	0.1	3	0.9881	0.9260	0.9881	0.9259	272.28
200	100	0.1	3	0.9984	0.9292	0.9984	0.9290	365.97
500	100	0.1	3	1.0000	0.9448	1.0000	0.9447	922.18

Table 9: XGBoost (Multi-class) + PCA results for varying number of estimators.

4.1 Bias–Variance Tradeoff Analysis

Effect of PCA Components From Table 1, increasing PCA components from 50 to 300 slightly improves validation accuracy (0.8817 $\to$ 0.8856) while keeping train accuracy stable (0.9147 $\to$ 0.9159). This indicates **low bias** across all components. The small train-validation gap shows **low variance**, meaning the model generalizes well regardless of moderate dimensionality reduction.

Effect of Number of Estimators From Table 2, increasing estimators (100 $\to$ 500) increases train accuracy from 0.9630 $\to$ 1.0, while validation accuracy improves moderately (0.9120 $\to$ 0.9448). This shows that **bias decreases** as the model becomes more complex. However, the growing train-validation gap (0.9630-0.9120 $\to$ 1.0-0.9448) indicates **slightly increased variance**, but still controlled.

Conclusion

- Moderate PCA reduction (100–300 components) retains most information, keeping bias low and variance controlled.
- Increasing the number of estimators reduces bias, slightly increasing variance, but improves overall performance.
- The best tradeoff is achieved with PCA=100 and 500 estimators, giving high accuracy and F1 while maintaining acceptable variance.

5 Conclusions

- Thought with 100 estimators in xgboost i got a better f1 score but we will have to compromise training time.
- Hence i think KNN is the best one in terms of accuracy ,f1 score and training time .
- But on compromising training time i think xgboost with PCA is also good .
- To increase the f1 score and decrease training time I normalized the values and used PCA .
- For 100 components we are getting a good f1 score with less training time .
- For increasing the f1 score even more i tried implementing stacked models but session is crashed due to the usage of all available ram .But i uploaded that code file in github repository .
- Most of the models have similar train and validation f1 score.

6 Thoughts and conclusions

- F1 score is not the only thing to consider we have to check the training time to decide which is the best model.
- If there are more features we can try using PCA to reduce the number of features without loosing a lot of information .
- In most of the cases PCA should be able to give a better f1 score and reduce the training time .
- In most of the cases ensemble of models works much better than single model but in this case session is getting crashed due to some reasons .