LIKHITH BHARGAV (2021A7PS0444H)

SRIVARAN KUKATLA (2021A4PS1988H)

OMKAR BASANI (2021A7PS0433H)

**BITS F464 -MACHINE LEARNING**

ASSIGNMENT-2

PART A:

Naive Bayes classifier:

* The Naive Bayes classifier is simple, fast, and requires relatively few training examples compared to other classification algorithms. It is often used in natural language processing and text classification tasks, where it has been shown to perform well. However, its assumption of independence between features may not hold true in some cases, leading to inaccurate predictions.
* The performance metrics we got are as follows,

Accuracy: 75.37

Precision: 75.37

Recall: 1

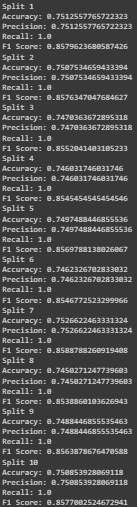
F1 score: 0.859

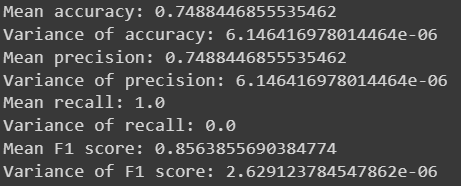
Confusion Matrix:

[0 2514]

[0 7440]

* After 10 splits in Naïve Bayes:





Smoothing:

* Laplace smoothing:

This is also known as add-one smoothing. In this technique, we add one to the count of each feature for each class. This ensures that the conditional probability of each feature for each class is non-zero.

Accuracy: 74.74

Precision: 74.74

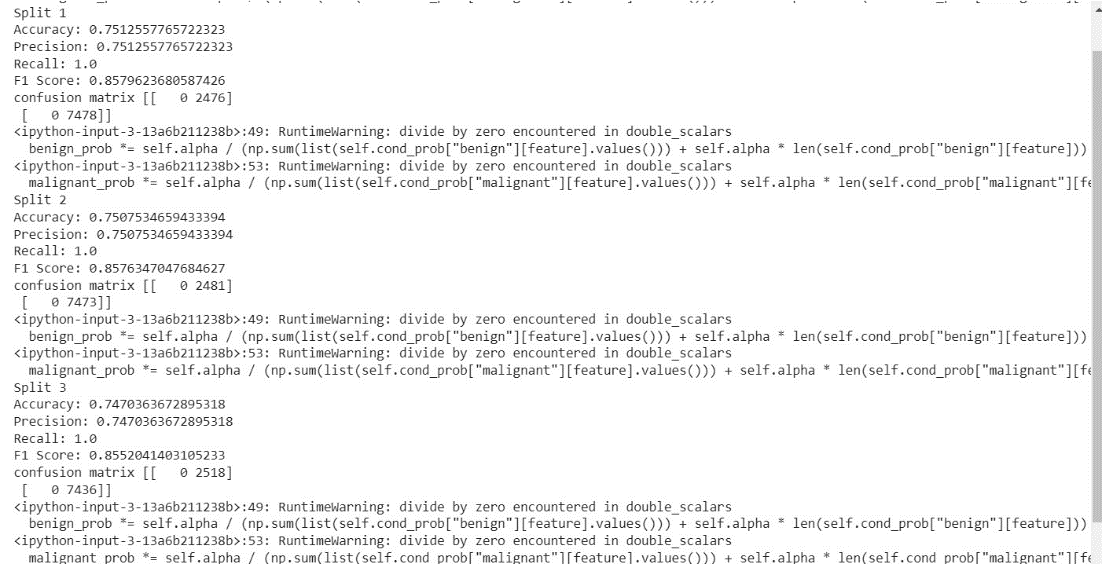
Recall: 1.0

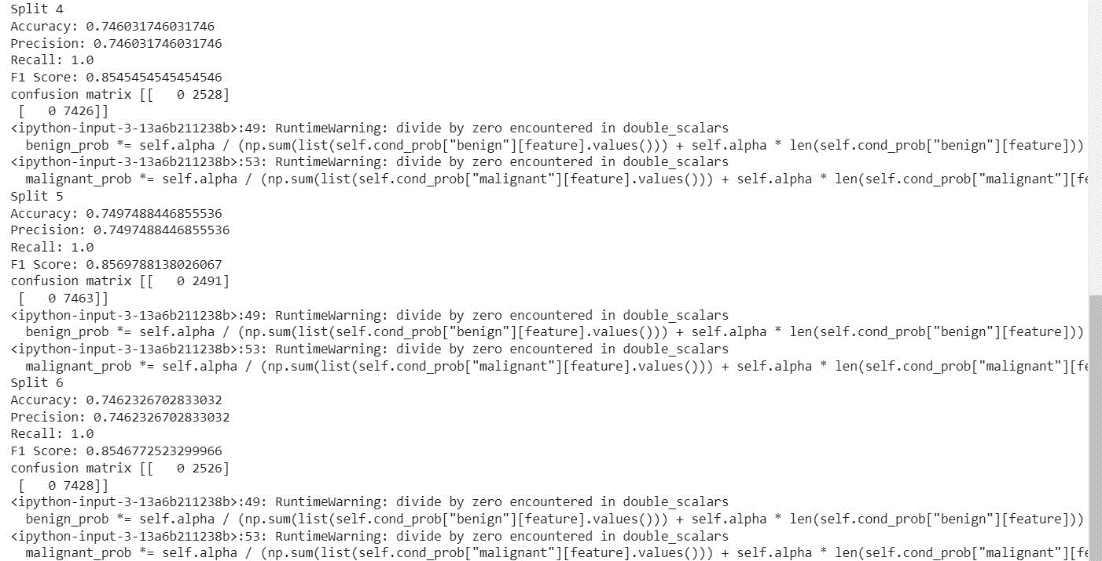
Confusion Matrix:

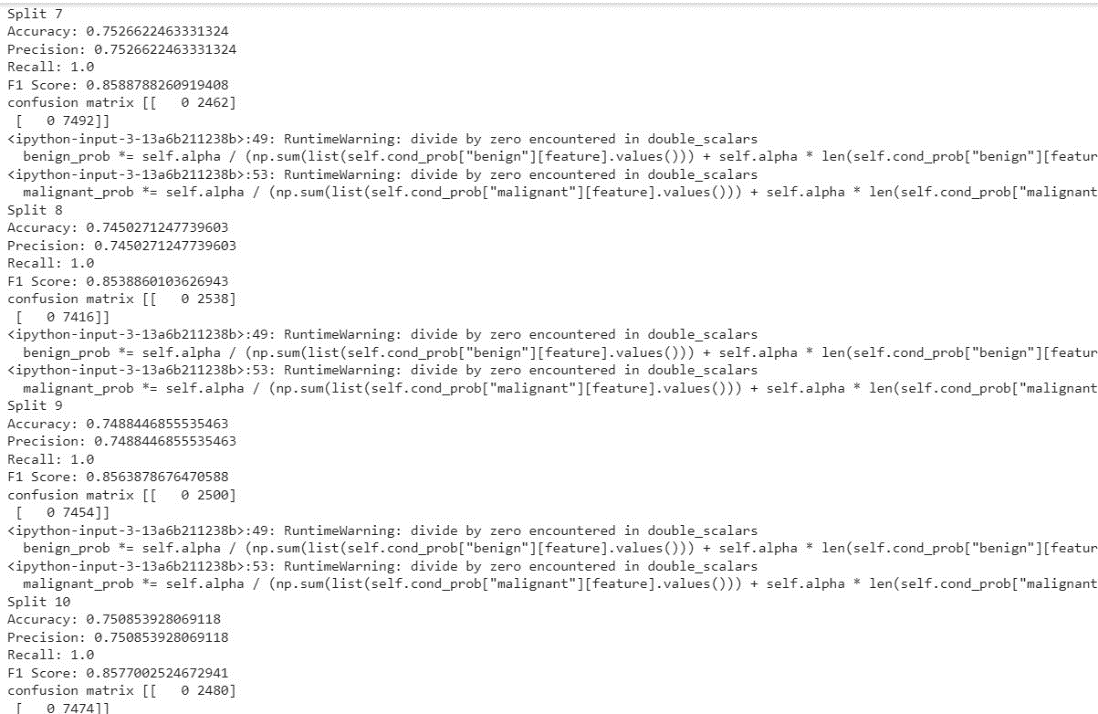
[0 2514]

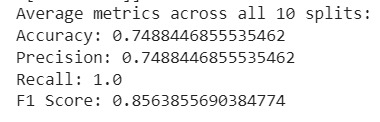
[0 7440]

After 10 splits:





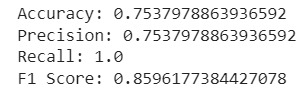




* Different smoothing techniques also include:

Good Turing smoothing:

Good-Turing smoothing can improve the accuracy of language models, machine translation systems, and other natural language processing applications that rely on estimating probabilities from limited data. However, it may not always be the best choice for all applications, as it assumes that the observed frequencies follow a specific distribution, which may not be true in all cases.



* Dirichlet smoothing :

Dirichlet smoothing adds a small constant to the count of each feature value, effectively creating a non-zero count for all possible feature values. This ensures that no probability is zero and that even unseen feature values have a non-zero probability.

Accuracy: 0.7474382157926461

Precision: 0.7474382157926461

Recall: 1.0

F1 Score: 0.8554674025526042

confusion matrix [[ 0 2514]

[ 0 7440]]

* Logistic regression:

Accuracy: 77.62

Precision: 79.60

Recall: 0.94

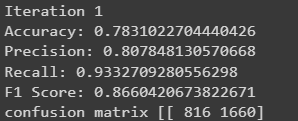
F1 score: 0.8628

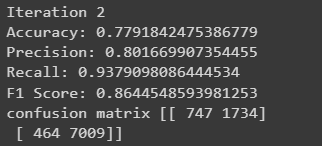
Confusion Matrix:

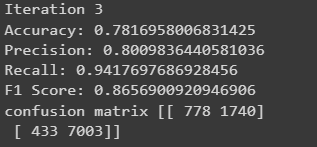
[719 1795]

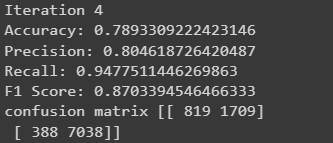
[432 7008]

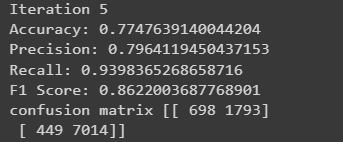
After 10 splits:

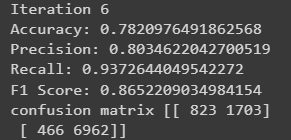


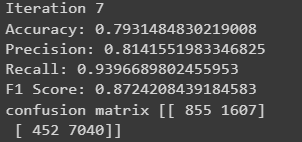


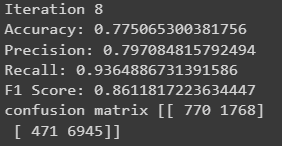


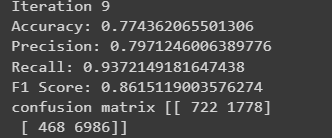


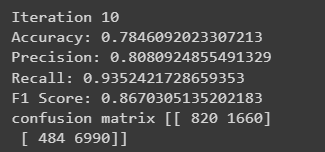


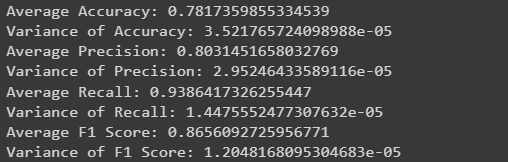












Comparison of Laplace and Logistic regression:

* The accuracies in the both the cases of logistic regression and naïve bayes are almost the same, we got a slightly higher accuracy in logistic regression
* K nearest neighbours:

Accuracy: 82.06

Precision: 86.01

Recall: 0.90

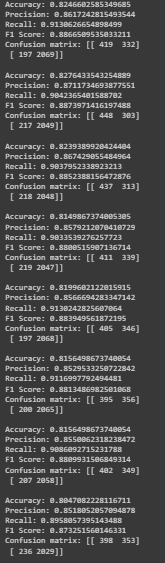
F1 score: 0.8832

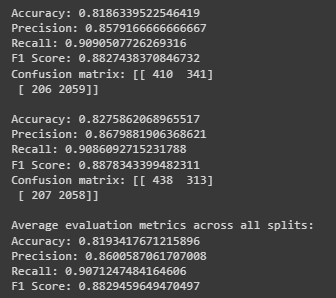
Confusion Matrix:

[1416 1098]

[687 6753]

After 10 splits:





Comparison of Laplace and K nearest neighbours:

* K-nearest neighbours (KNN) can be better than Naive Bayes for nonlinear data, imbalanced datasets, noisy data, and high-dimensional data. KNN does not assume any distribution for the input features, making it more suitable for handling nonlinear data. It can capture the distribution of each class separately, which makes it more useful for imbalanced

datasets

* Here in our case, we got a better accuracy in case of k-nearest neighbours than naïve bayes.

PART B:

Artificial neural networks:

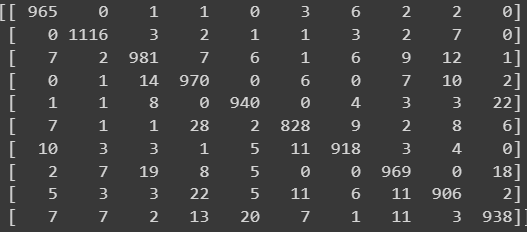
The performance metrics we got are as follows,

Model-1:

Accuracy: 95.31

Precision: 95.3

Confusion Matrix:

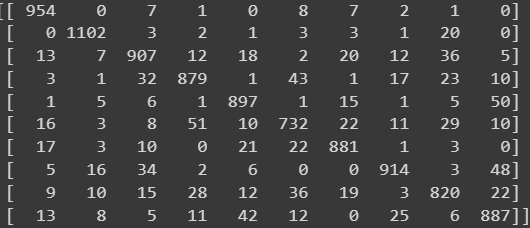


Model-2:

Accuracy: 89.7

Precision: 89.58

Confusion Matrix:

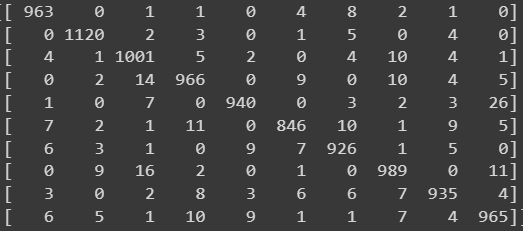


Model-3:

Accuracy: 96.51

Precision: 96.5

Confusion Matrix:

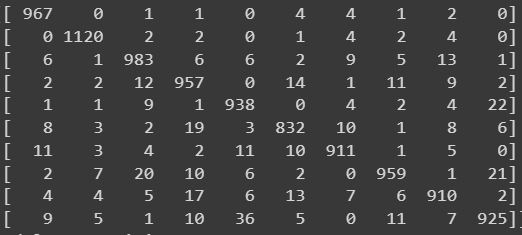


Model-4:

Accuracy: 95.02

Precision: 94.98

Confusion Matrix:

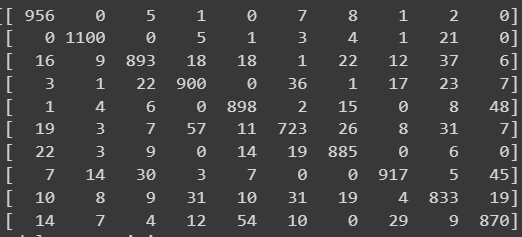


Model-5:

Accuracy: 89.75

Precision: 89.61

Confusion Matrix:

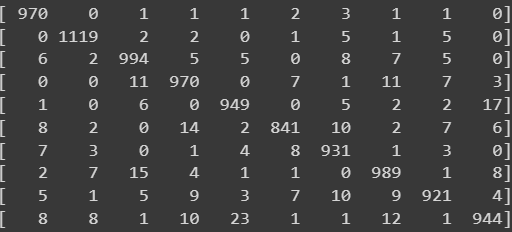


Model-6:

Accuracy: 96.28

Precision: 96.27

Confusion Matrix:

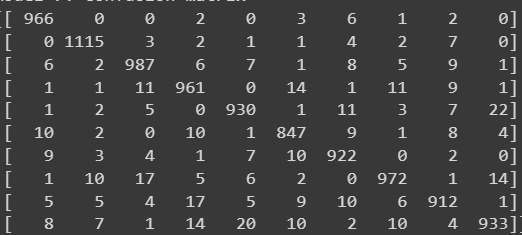


Model-7:

Accuracy: 95.47

Precision: 95.41

Confusion Matrix:

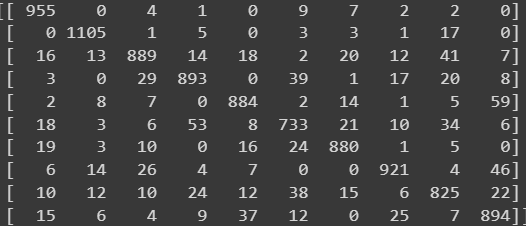


Model-8:

Accuracy: 89.79

Precision: 89.64

Confusion Matrix:

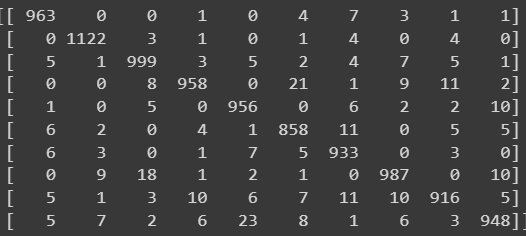


Model-9:

Accuracy: 96.4

Precision: 96.36

Confusion Matrix:

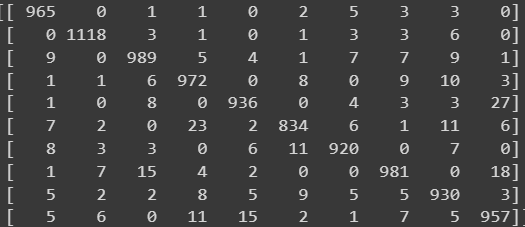


Model-10:

Accuracy: 96.02

Precision: 96.00

Confusion Matrix:

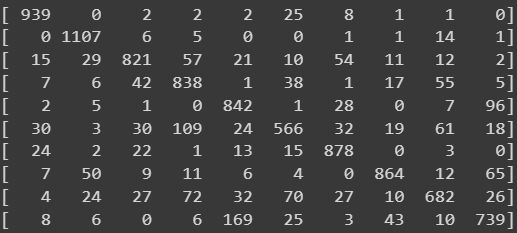


Model-11:

Accuracy: 82.76

Precision: 82.48

Confusion Matrix:

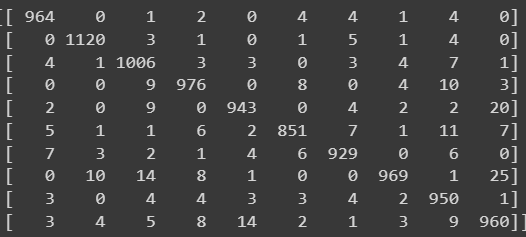


Model-12:

Accuracy: 96.68

Precision: 96.68

Confusion Matrix:

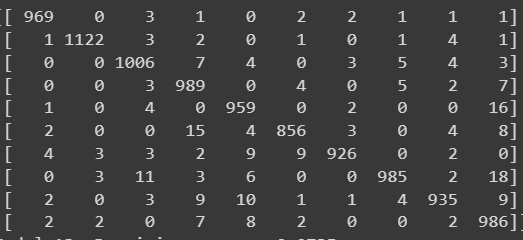


Model-13:

Accuracy: 97.33

Precision: 97.35

Confusion Matrix:

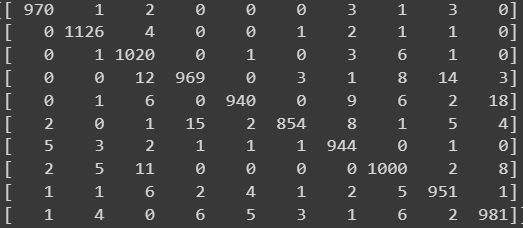


Model-14:

Accuracy: 97.55

Precision: 97.57

Confusion Matrix:

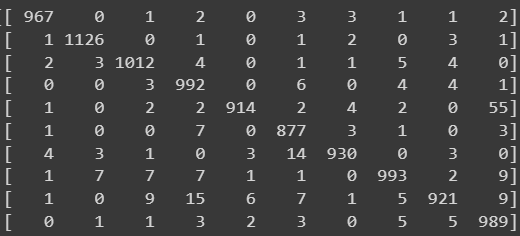


Model-15:

Accuracy: 97.21

Precision: 97.23

Confusion Matrix:



* In the provided code, model 14 has 2 hidden layers, each with 150 neurons, and uses the hyperbolic tangent (tanh) activation function, and the Adam optimizer.

The Adam optimizer is a popular optimization algorithm that uses a combination of the gradient descent with momentum and RMSProp algorithms, which can help it converge faster and avoid getting stuck in local minima.

The tanh activation function is similar to the sigmoid function, but with a range of [-1, 1], which can help it avoid the saturation problem that occurs when the sigmoid function is close to 0 or 1.

Additionally, having more neurons in the hidden layers can allow the model to learn more complex representations of the input data.

Therefore, the combination of the Adam optimizer, tanh activation function, and more neurons in the hidden layers may have allowed model 14 to learn more complex representations of the MNIST dataset and achieve higher accuracy on the test set.

* some possible reasons for the lower accuracy of Model 11 could be:

1)The number of neurons in the hidden layers might not be sufficient for the model to learn the complex features of the MNIST dataset.

2)The activation function used in the hidden layers might not be suitable for the given architecture, and as a result, the model might not be able to capture the underlying patterns in the data.

3)The optimizer used for Model 11 might not be efficient in minimizing the loss function of the model and might have resulted in a suboptimal solution.

4)Model 11 might have overfit on the training data, which is causing the lower accuracy on the test set.