ASSIGNMENT REPORT

Part A - Naive Bayes Classifier to predict income

Performance metrics:

Accuracy	Precision	Recall	F1 Score
0.8014	0.6475	0.5083	0.5695

Naive Bayes is a probabilistic classifier that makes classifications using a Posterior decision rule inBayesian setting.

According to Bayes' Rule:
P(S|V) = P(V|S)*P(S) / (P(V|S)*P(S) + P(V |S')*P(S'))

Laplace Smoothing:

The smoothing technique which we have used is an additive smoothing also known as laplace smoothing which is incorporated in our code as "alpha". This small value alpha is added in numerator and denominator in calculation of some parameters which improves accuracy of our model.

Performance metrics:

Accuracy	Precision	Recall	F1 Score
0.8204	0.6799	0.5072	0.581

- ➢ Here we can observe that smoothing increases the accuracy of our naive bayes model and hence becomes a better model compared to before.

- ➢ Though precision and recall have changed , there is not much deviation in F1 score; it may increase or decrease w.r.t smoothing technique.

KNN:

This algorithm makes use of K nearest neighbour to the test case and choses them based on the euclidean distance between it and all the neighbours. Though this algorithm was not taught in the lectures, it is preferred to take the number of neighbours near the root of no of elements in the data set.

The given dataset has nearly 33000 elements and its root is between 179 and 180 and hence after experimenting with some values near 180, we chose 239 as it gives better accuracy.

Performance metrics:

Accuracy	Precision	Recall	F1 Score
0.825	0.7144	0.478	0.5728

Here we can observe that the accuracy is better than Naive Bayes model , we can say that KNN gives a better model for the dataset.

Logistic:

The logistic regression model is implemented using sklearn for the given dataset after appropriate preprocessing. This model is also a classification model which assigns any new data element to a class

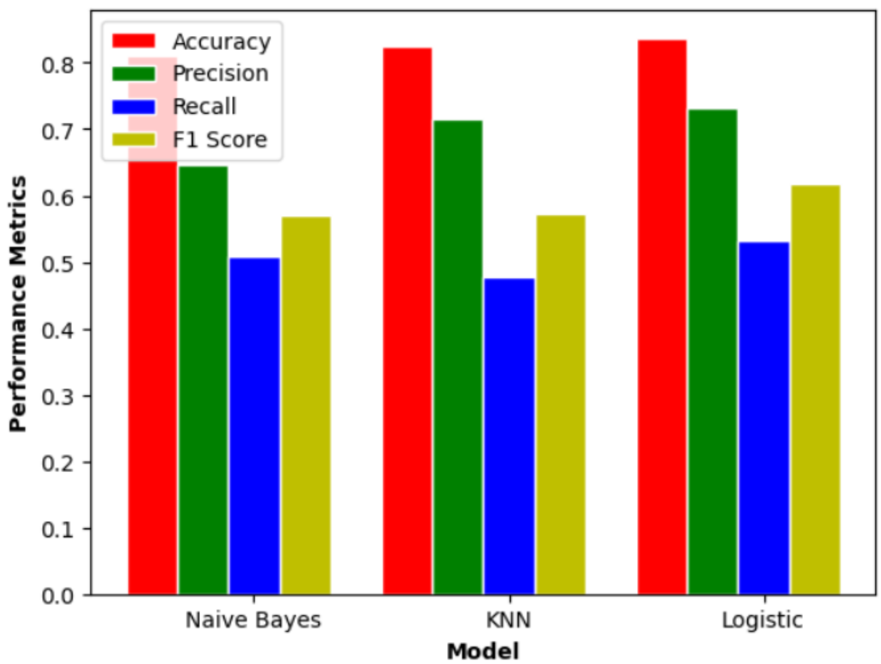
Performance metrics:

Accuracy	Precision	Recall	F1 Score
0.8374	0.7319	0.533	0.6168

Naive bayes vs KNN vs Logistic:

The below table shows the performance metrics comparison of three different models.

Performance Metrics	Model		
	Naive Bayes	**KNN**	**Logistic**
Accuracy	0.8114	0.825	0.8374
Precision	0.6475	0.7144	0.7319
Recall	0.5083	0.478	0.533
F1 score	0.5695	0.5728	0.6168



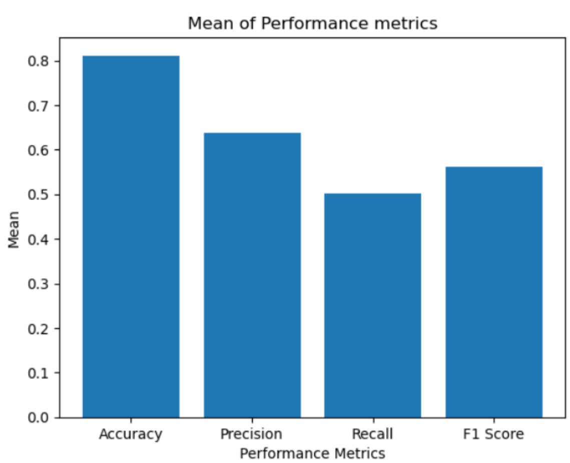
Analysis:

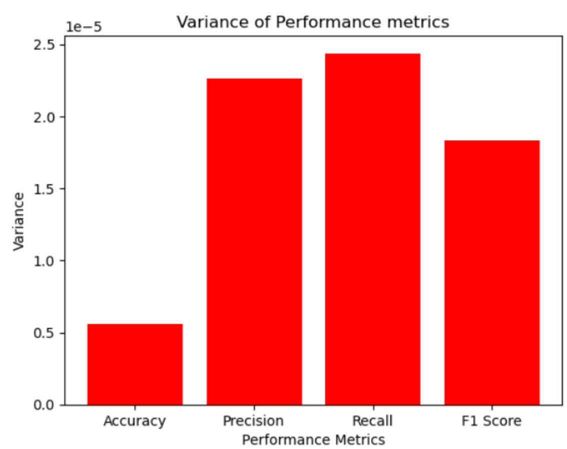
From the above graphs and tables we can observe from accuracy that the best model is logistic regression model followed by KNN and at last we have Naive Bayes model. So we can conclude that for adult data, the best Classification model is the **Logistic Model**.

Performance Metrics for 10 random train-test splits:

Accuracy	Precision	Recall	F1 Score
0.8124	0.6344	0.4941	0.5556
0.8114	0.6475	0.5083	0.5695
0.8102	0.637	0.5056	0.5637
0.8125	0.6429	0.5002	0.5626
0.8112	0.6394	0.4996	0.5609
0.8118	0.6398	0.5067	0.5655
0.8089	0.6332	0.4956	0.556
0.8135	0.6445	0.4992	0.5626
0.8134	0.6346	0.5045	0.5621
0.8057	0.6364	0.4968	0.558
0.8111	0.638	0.501	0.5617
0.000005566666667	0.00002260233333	0.00002437822222	0.00001834055556

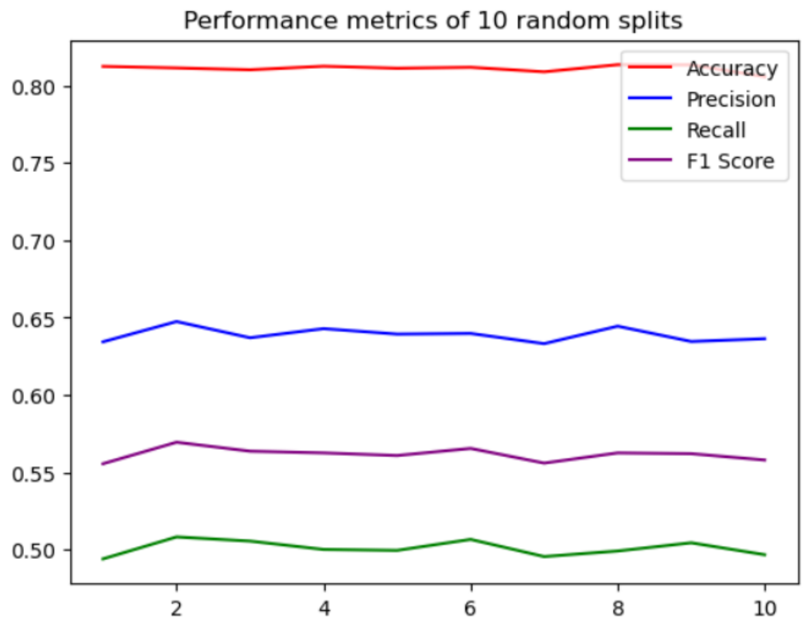
The last 2 rows indicate average and variance of the performance metrics for 10 random training and testing split of naive bayes model.





Observation:

- In the above graphs, variance for recall and F1 score are high stating that they have a larger deviation with mean than accuracy or precision.
- The average accuracy for our model without smoothing is nearly **81%**.



Analysis:

- ❖ From the above graph of performance metrics, we can say that the accuracy for the model is high
- ❖ precision for our model is >50%.
- ❖ This leads to very less classification of false positives.
- ❖ Whereas Recall value is around 50% leading to high classification of False Negatives.
- ❖ The values of F1 score is between recall and precision as it is the harmonic mean of the two

Part B: Building a Basic Neural Network for Image Classification

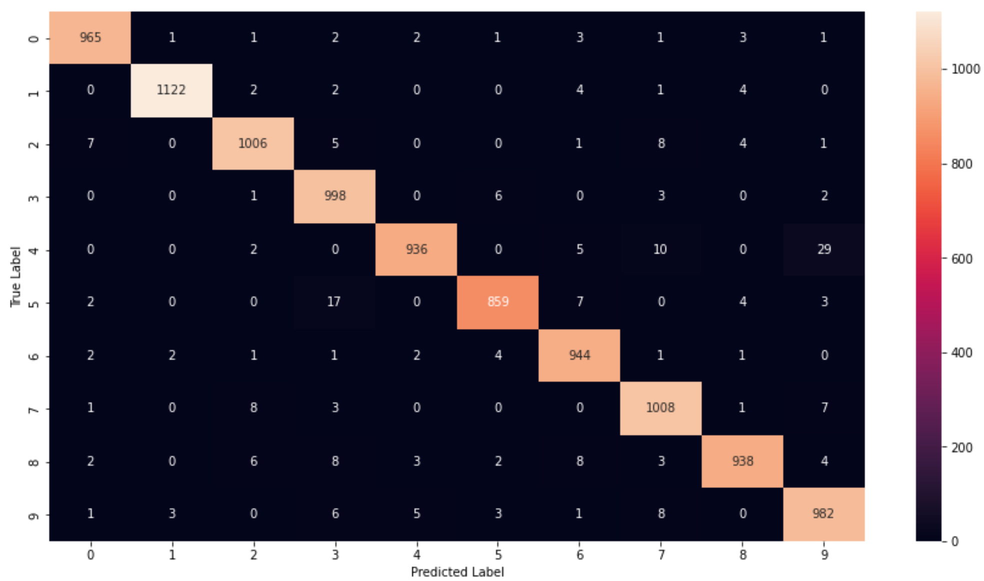
An Artificial Neural Network consists of multiple layers which process the data and predict the output. Each layer consists of multiple nodes, which takes inputs from the previous layers and transforms it according to a function. The goal of the Artificial Neural Network is to learn a suitable set of weights to minimise the loss function. This is done by updating the weights using gradient descent.
Different types of layers used here are

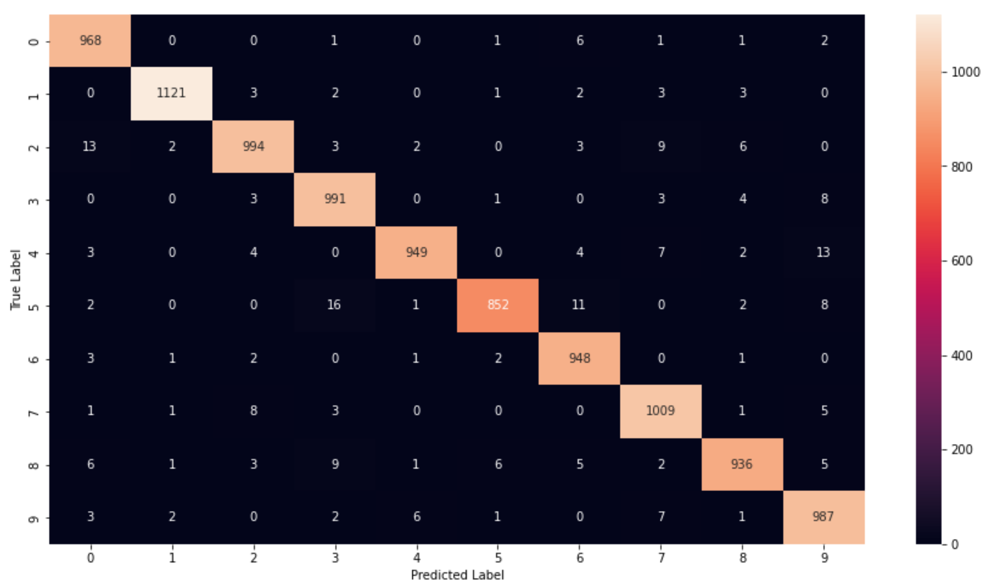
1. Tanh
2. Sigmoid
3. Rectified Linear Unit (ReLU)

Performance Metrics of ANN

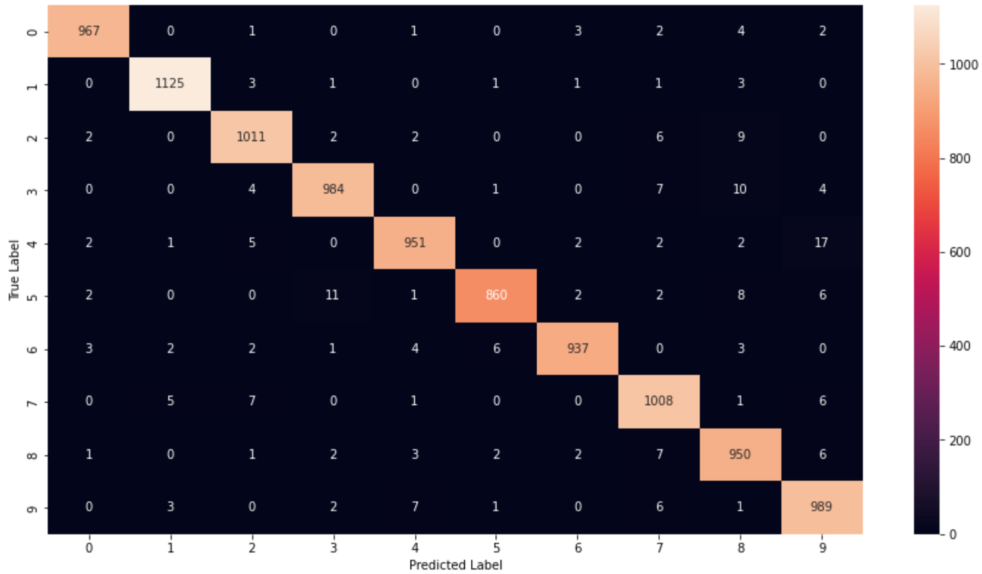
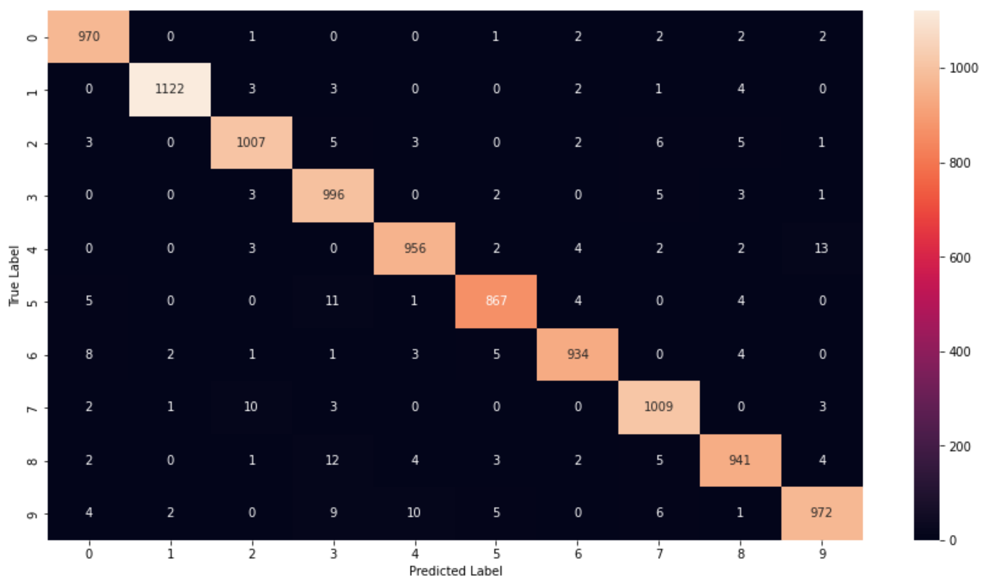
S. No.	No. of Hidden Layers	No. of Neurons in the Hidden Layers Respectively	Activation Functions Used Respectively	Accuracy
1	2	150, 100	relu, relu	0.9758
2	2	100, 100	relu, relu	0.9755
3	2	150, 150	relu, tanh	0.9782
4	2	150, 100	sigmoid, tanh	0.9774
5	2	100, 150	tanh, tanh	0.9712
6	2	150, 150	sigmoid, sigmoid	0.9745
7	2	150, 100	relu, sigmoid	0.9765
8	3	150, 100, 100	relu, sigmoid, tanh	0.9764
9	3	150, 100, 100	relu, relu, tanh	0.9791
10	3	150, 100, 100	sigmoid, tanh, tanh	0.9774
11	3	100, 150, 100	sigmoid, tanh, tanh	0.9763
12	3	150, 150, 150	sigmoid, relu, tanh	0.9746
13	3	100, 150, 100	relu, relu, relu	0.9804
14	3	150, 150, 100	tanh, sigmoid, relu	0.9748
15	3	150, 150, 100	tanh, relu, sigmoid	0.9748

Confusion Matrices for each case respectively

<table>
<tr><th>Model Number</th><th>Confusion Matrix</th></tr>
<tr><td>1</td><td>


</td></tr>
<tr><td>2</td><td>


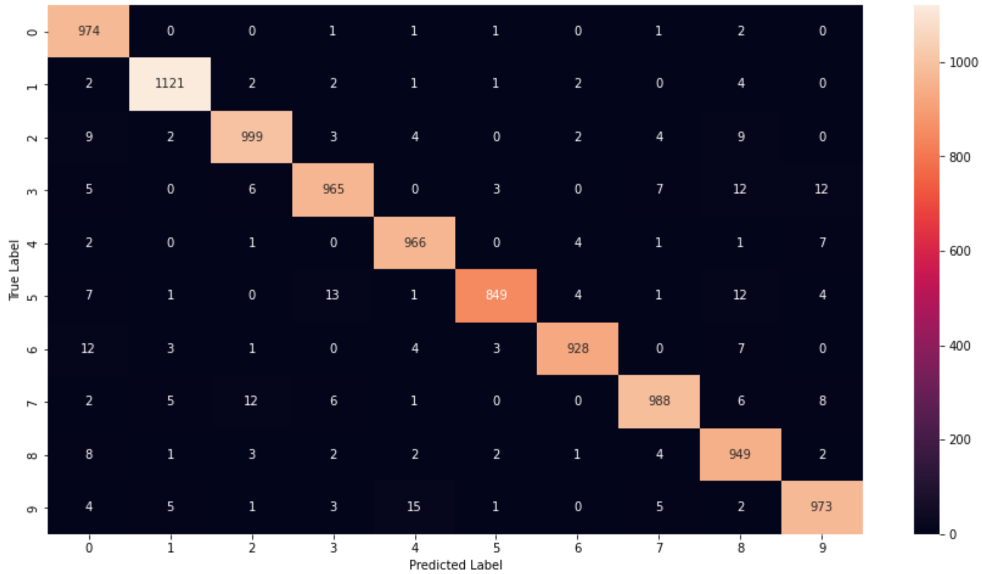
</td></tr>
</table>

3	
4	

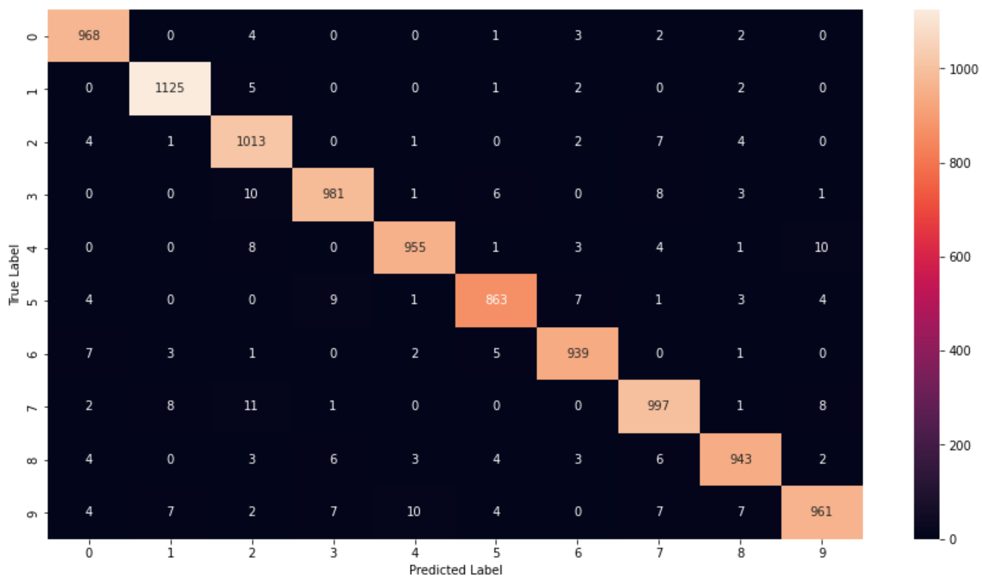
3
True Label
Predicted Label
967 0 1 0 1 0 3 2 4 2
0 1125 3 1 0 1 1 1 3 0
2 0 1011 2 2 0 0 6 9 0
0 0 4 984 0 1 0 7 10 4
2 1 5 0 951 0 2 2 2 17
2 0 0 11 1 860 2 2 8 6
3 2 2 1 4 6 937 0 3 0
0 5 7 0 1 0 0 1008 1 6
1 0 1 2 3 2 2 7 950 6
0 3 0 2 7 1 0 6 1 989
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

4
True Label
Predicted Label
970 0 1 0 0 1 2 2 2 2
0 1122 3 3 0 0 2 1 4 0
3 0 1007 5 3 0 2 6 5 1
0 0 3 996 0 2 0 5 3 1
0 0 3 0 956 2 4 2 2 13
5 0 0 11 1 867 4 0 4 0
8 2 1 1 3 5 934 0 4 0
2 1 10 3 0 0 0 1009 0 3
2 0 1 12 4 3 2 5 941 4
4 2 0 9 10 5 0 6 1 972
0 1 2 3 4 5 6 7 8 9
1000
800
600
400
200
0

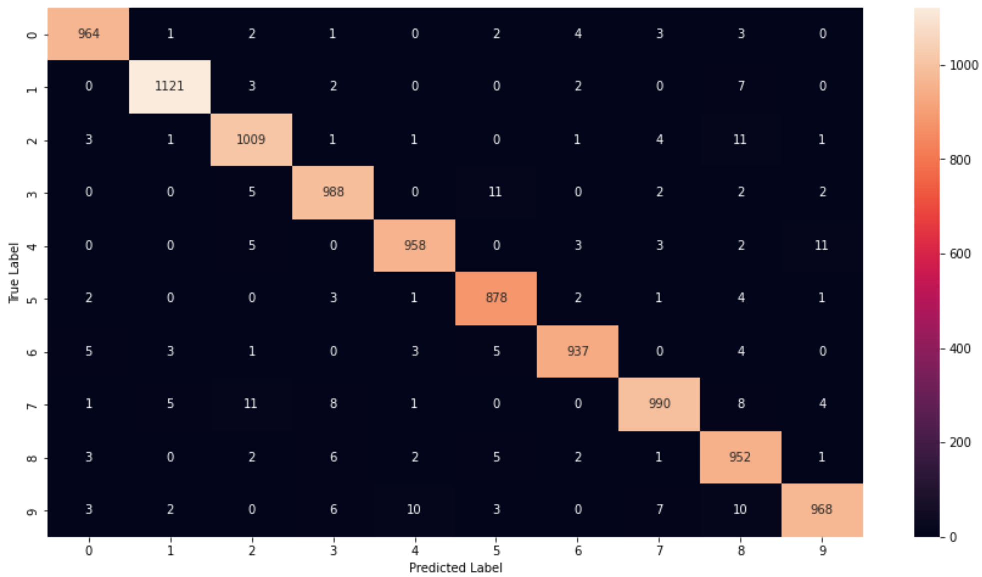
5


True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
0: 974 0 0 1 1 1 0 1 2 0
1: 2 1121 2 2 1 1 2 0 4 0
2: 9 2 999 3 4 0 2 4 9 0
3: 5 0 6 965 0 3 0 7 12 12
4: 2 0 1 0 966 0 4 1 1 7
5: 7 1 0 13 1 849 4 1 12 4
6: 12 3 1 0 4 3 928 0 7 0
7: 2 5 12 6 1 0 0 988 6 8
8: 8 1 3 2 2 2 1 4 949 2
9: 4 5 1 3 15 1 0 5 2 973
1000
800
600
400
200
0

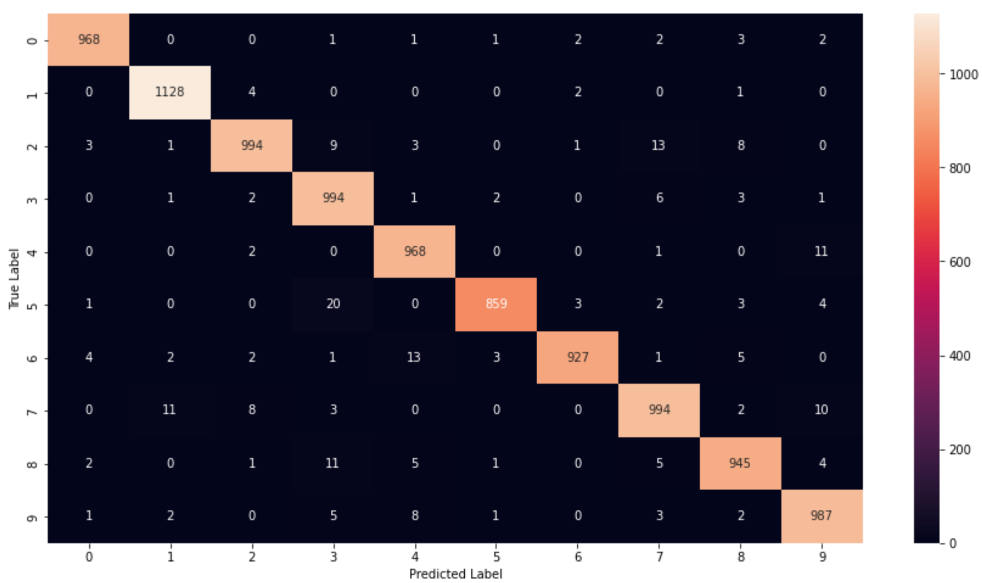
6


True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
0: 968 0 4 0 0 1 3 2 2 0
1: 0 1125 5 0 0 1 2 0 2 0
2: 4 1 1013 0 1 0 2 7 4 0
3: 0 0 10 981 1 6 0 8 3 1
4: 0 0 8 0 955 1 3 4 1 10
5: 4 0 0 9 1 863 7 1 3 4
6: 7 3 1 0 2 5 939 0 1 0
7: 2 8 11 1 0 0 0 997 1 8
8: 4 0 3 6 3 4 3 6 943 2
9: 4 7 2 7 10 4 0 7 7 961
1000
800
600
400
200
0

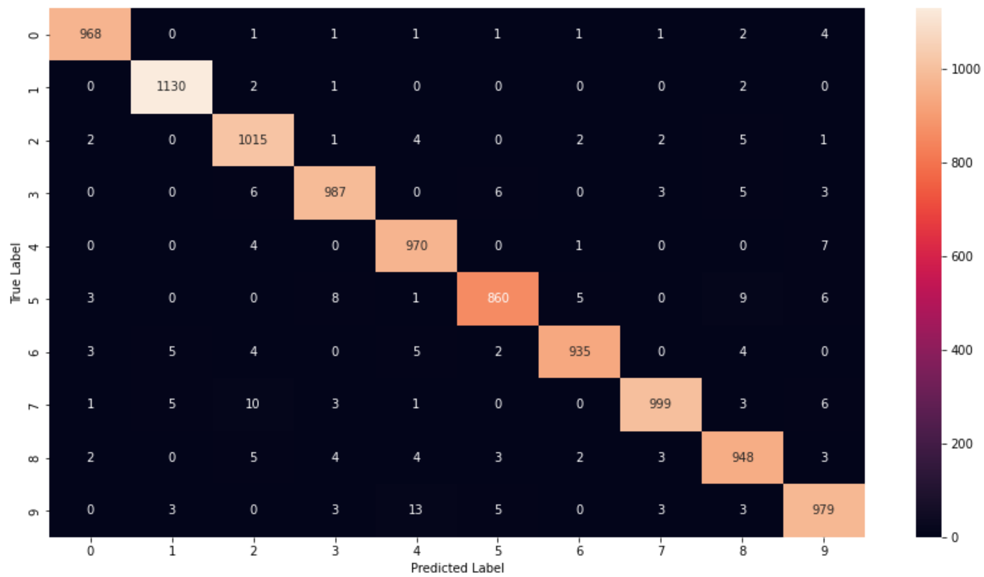
7


True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
964 1 2 1 0 2 4 3 3 0
0 1121 3 2 0 0 2 0 7 0
3 1 1009 1 1 0 1 4 11 1
0 0 5 988 0 11 0 2 2 2
0 0 5 0 958 0 3 3 2 11
2 0 0 3 1 878 2 1 4 1
5 3 1 0 3 5 937 0 4 0
1 5 11 8 1 0 0 990 8 4
3 0 2 6 2 5 2 1 952 1
3 2 0 6 10 3 0 7 10 968
0
200
400
600
800
1000

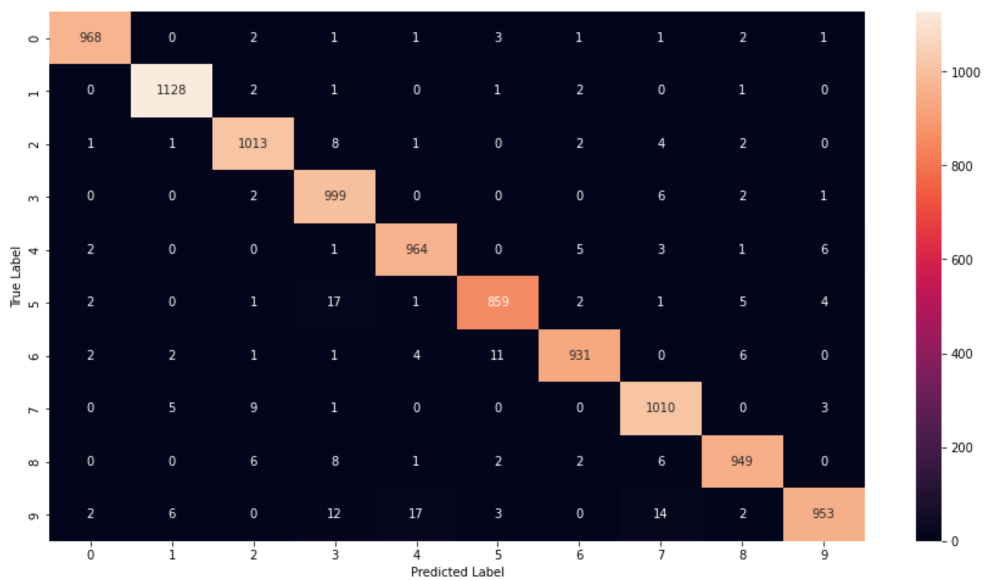
8


True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
968 0 0 1 1 1 2 2 3 2
0 1128 4 0 0 0 2 0 1 0
3 1 994 9 3 0 1 13 8 0
0 1 2 994 1 2 0 6 3 1
0 0 2 0 968 0 0 1 0 11
1 0 0 20 0 859 3 2 3 4
4 2 2 1 13 3 927 1 5 0
0 11 8 3 0 0 0 994 2 10
2 0 1 11 5 1 0 5 945 4
1 2 0 5 8 1 0 3 2 987
0
200
400
600
800
1000

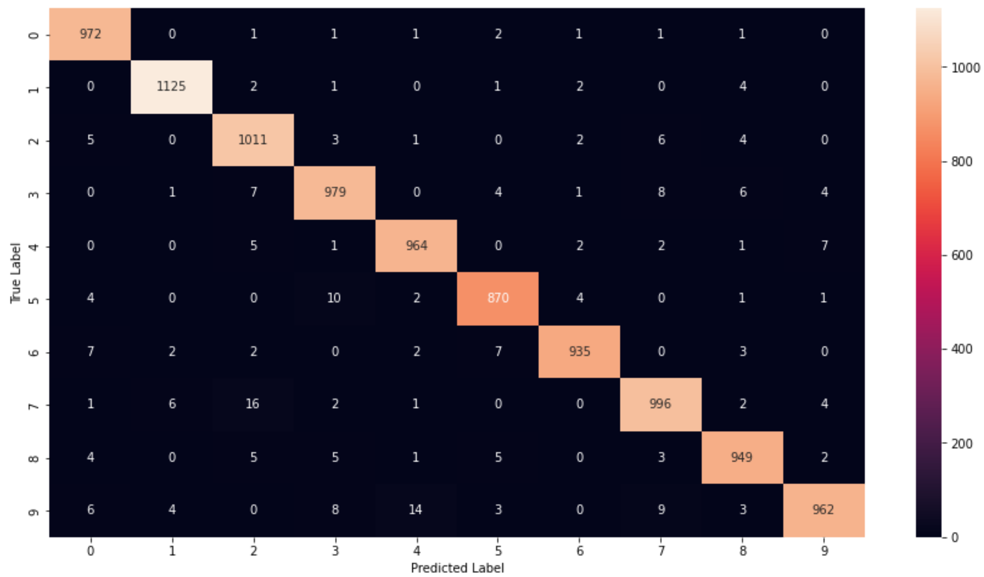
9


True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
0: 968 0 1 1 1 1 1 1 2 4
1: 0 1130 2 1 0 0 0 0 2 0
2: 2 0 1015 1 4 0 2 2 5 1
3: 0 0 6 987 0 6 0 3 5 3
4: 0 0 4 0 970 0 1 0 0 7
5: 3 0 0 8 1 860 5 0 9 6
6: 3 5 4 0 5 2 935 0 4 0
7: 1 5 10 3 1 0 0 999 3 6
8: 2 0 5 4 4 3 2 3 948 3
9: 0 3 0 3 13 5 0 3 3 979
1000
800
600
400
200
0

10

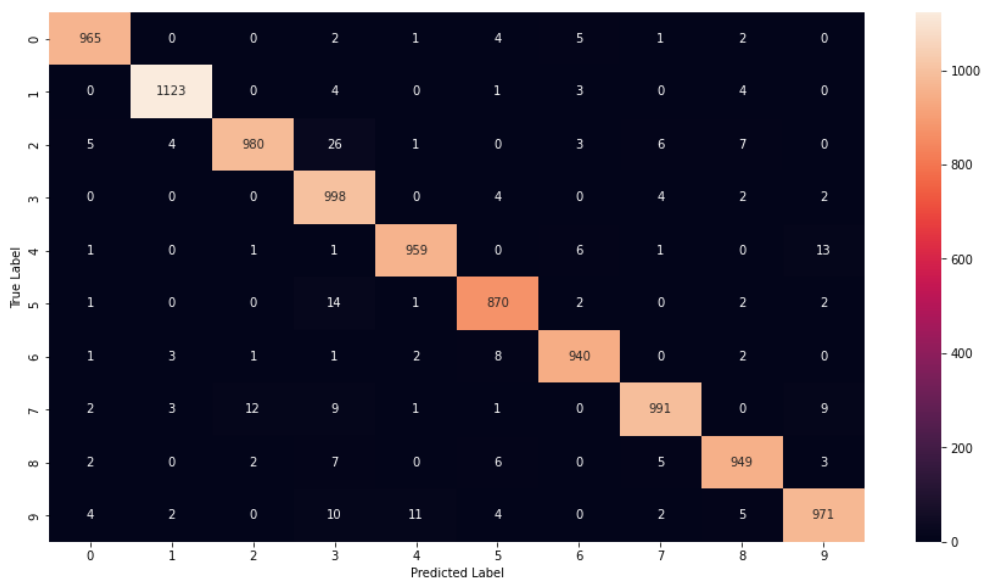

True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
0: 968 0 2 1 1 3 1 1 2 1
1: 0 1128 2 1 0 1 2 0 1 0
2: 1 1 1013 8 1 0 2 4 2 0
3: 0 0 2 999 0 0 0 6 2 1
4: 2 0 0 1 964 0 5 3 1 6
5: 2 0 1 17 1 859 2 1 5 4
6: 2 2 1 1 4 11 931 0 6 0
7: 0 5 9 1 0 0 0 1010 0 3
8: 0 0 6 8 1 2 2 6 949 0
9: 2 6 0 12 17 3 0 14 2 953
1000
800
600
400
200
0

11



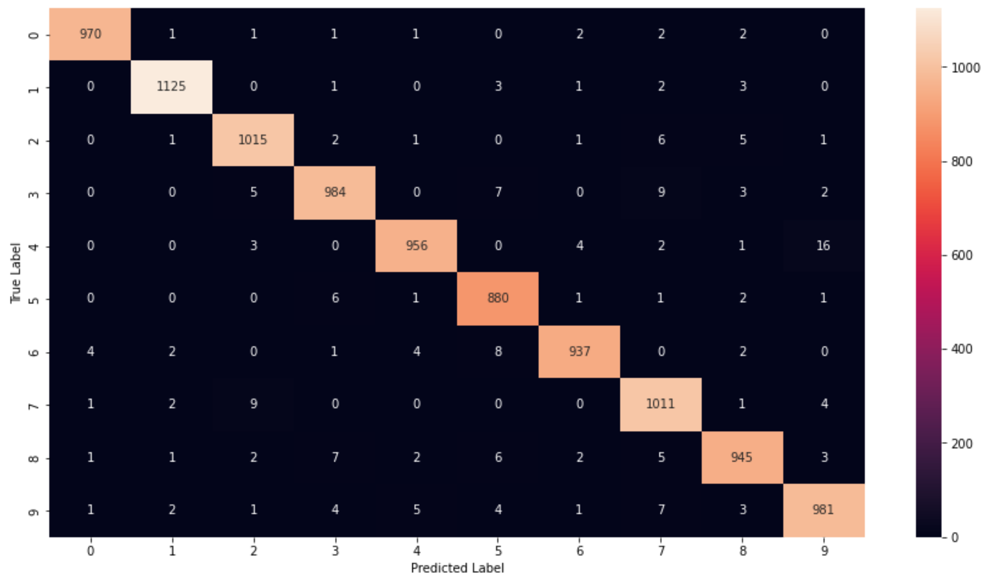
True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
972 0 1 1 1 2 1 1 1 0
0 1125 2 1 0 1 2 0 4 0
5 0 1011 3 1 0 2 6 4 0
0 1 7 979 0 4 1 8 6 4
0 0 5 1 964 0 2 2 1 7
4 0 0 10 2 870 4 0 1 1
7 2 2 0 2 7 935 0 3 0
1 6 16 2 1 0 0 996 2 4
4 0 5 5 1 5 0 3 949 2
6 4 0 8 14 3 0 9 3 962
1000
800
600
400
200
0

12

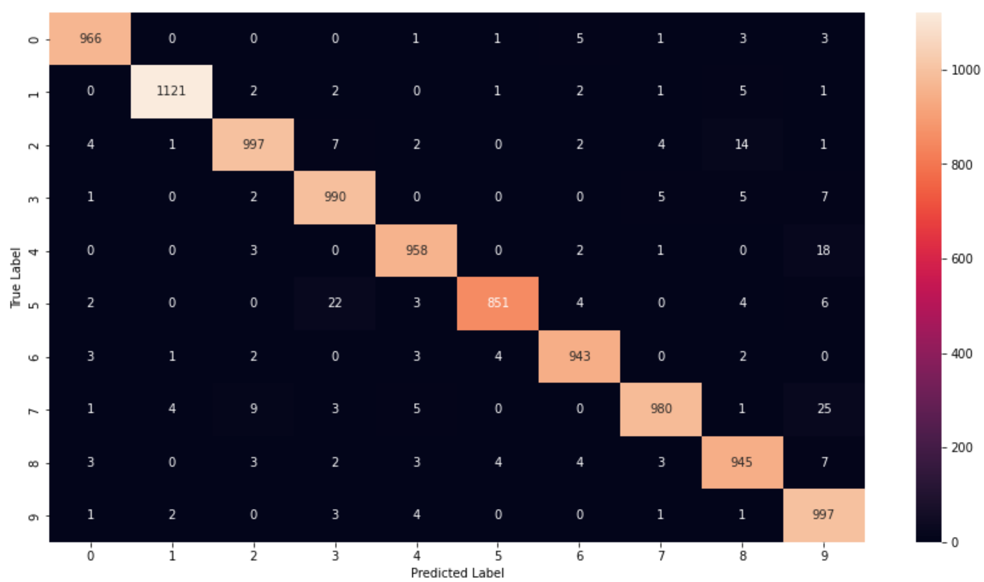


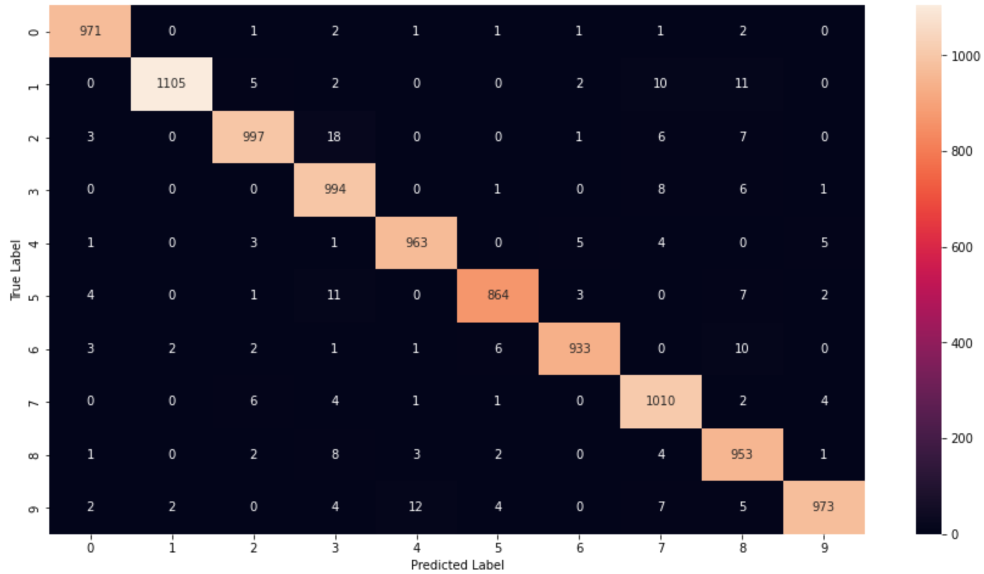
True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
965 0 0 2 1 4 5 1 2 0
0 1123 0 4 0 1 3 0 4 0
5 4 980 26 1 0 3 6 7 0
0 0 0 998 0 4 0 4 2 2
1 0 1 1 959 0 6 1 0 13
1 0 0 14 1 870 2 0 2 2
1 3 1 1 2 8 940 0 2 0
2 3 12 9 1 1 0 991 0 9
2 0 2 7 0 6 0 5 949 3
4 2 0 10 11 4 0 2 5 971
1000
800
600
400
200
0

13


True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
970 1 1 1 1 0 2 2 2 0
0 1125 0 1 0 3 1 2 3 0
0 1 1015 2 1 0 1 6 5 1
0 0 5 984 0 7 0 9 3 2
0 0 3 0 956 0 4 2 1 16
0 0 0 6 1 880 1 1 2 1
4 2 0 1 4 8 937 0 2 0
1 2 9 0 0 0 0 1011 1 4
1 1 2 7 2 6 2 5 945 3
1 2 1 4 5 4 1 7 3 981
0
200
400
600
800
1000

14


True Label
Predicted Label
0 1 2 3 4 5 6 7 8 9
966 0 0 0 1 1 5 1 3 3
0 1121 2 2 0 1 2 1 5 1
4 1 997 7 2 0 2 4 14 1
1 0 2 990 0 0 0 5 5 7
0 0 3 0 958 0 2 1 0 18
2 0 0 22 3 851 4 0 4 6
3 1 2 0 3 4 943 0 2 0
1 4 9 3 5 0 0 980 1 25
3 0 3 2 3 4 4 3 945 7
1 2 0 3 4 0 0 1 1 997
0
200
400
600
800
1000

<table>
<tr><td>15</td><td>
</td></tr>
</table>

T-test for accuracy for above:

(Default) $\alpha = 0.05$
Number of degrees of freedom for the above= n(no of models)-1 = 15-1 =14
$\mu_0 = 0.975$

H0: $\mu = \mu_0$
H1: $\mu \neq \mu_0$

Calculated mean for the above accuracy(μ) = 0.9762
Standard deviation for the above accuracy(s) = 0.0022

$$t = (\mu - \mu_0)/(s/\sqrt{n})$$
$= 0.9762 - 0.975 /(0.0022/\sqrt{15})$
$= 2.112$

t value for alpha (0.05) = 1.761
t value for alpha (0.025) = 2.145
=> $\alpha_t < \alpha(0.05)$
=> H0 (null hypothesis) is true

Conclusion:

As we got the null hypothesis(H0) true, which means that the accuracy of our models does not vary a lot, we can say we did not find any model to be statistically significant.