# BITS F464 : Machine Learning Assignment – 2 Report

## Team Members

1. Kushagra Verma (2020A7PS0225H)
2. Sai Panda (2020A7PS0080H)
3. Khooshrin Aspi Pithawalla - 2020A7PS2067H

# Part A: Naive Bayes Classifier to predict income

## Task 1

### The Dataset

The dataset provided had 32561 examples, with each example containing 14 features and a label indicating whether the annual income was 50K. Out of the 14 features, 6 features had continuous values (namely, "age", "fnlwgt", "education-num", "capital-gain", "capital-loss" and "hours-per-week") and the remaining 8 features had discrete values (namely, "workclass", "education", "marital-status", "occupation", "relationship", "race", "sex" and "native-country"). There were 24720 examples with income 50K and 7841 examples with income 50K.

#### Fill Missing Values

There were 2399 examples which had one or more missing values. We filled the missing values by going over each column and replacing the missing values with the most commonly occurring non-missing value in case of discrete features and with the mean of all the non-missing values in case of continuous features.

#### Create Training and Testing Sets

We split the dataset and used 67% of the dataset for training and 33% for testing. Our testing split contained 21816 examples and training split contained 10745 examples.

## Task 2

### Calculating Prior Probability

We calculated prior probability of each of two classes using maximum likelihood estimator.

### Calculating Conditional Probability

We calculated conditional probabilities of every feature assuming that features are independent of each other given the income class. We binned the continuous features into 10 bins of equal size for calculating likelihood.

### Predicting Class

We made predictions for the testing split using prior and likelihood that we had calculated earlier. We ignored the denominator since it is common to both income classes and directly compared the numerators to make our prediction.

## Task 3

### Evaluation

We evaluated our predictions by calculating the accuracy, precision, recall and F1-score. We calculated them using the following formulae:

### Smoothing

We applied Laplacian smoothing with values as 1, 10, 50 and 100. We used the following formula for smoothing:

where is the number of discrete values that education can take.

### Results

Here are the results averaged over 10 different test-train splits:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| α | TP | FP | TN | FN | Accuracy | Precision | Recall | F1-Score |
| N/A | 6821.0 | 587.8 | 1992.0 | 1344.2 | 0.689511 | 0.920660 | 0.835379 | 0.875944 |
| 1.0 | 6821.0 | 591.2 | 1988.6 | 1344.2 | 0.689828 | 0.920238 | 0.835378 | 0.875753 |
| 10.0 | 6853.1 | 621.8 | 1958.0 | 1312.1 | 0.695663 | 0.916818 | 0.839310 | 0.876347 |
| 50.0 | 7036.6 | 772.9 | 1806.9 | 1128.6 | 0.726803 | 0.901034 | 0.861786 | 0.880967 |
| 100.0 | 7243.1 | 948.4 | 1631.4 | 922.1 | 0.762355 | 0.884225 | 0.887078 | 0.885641 |

# Part B: Building a Basic Neural Network for Image Classification

We used Tensorflow to implement 27 different Artificial Neural Networks by varying the number of neurons in each layer and its activation function. We tested all the ANNs on the same test-train split of the MNIST dataset. Here are the results:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No. of neurons(HL1)** | Activation(HL1) | No. of neurons(HL2) | Activation(HL2) | Accuracy |
| 100 | relu | 50 | relu | 0.9777 |
| 100 | relu | 50 | tanh | 0.9777 |
| 50 | relu | 100 | tanh | 0.9772 |
| 100 | relu | 50 | sigmoid | 0.9770 |
| 100 | tanh | 50 | tanh | 0.9758 |
| 100 | tanh | 50 | relu | 0.9755 |
| 100 | sigmoid | 50 | tanh | 0.9753 |
| 100 | tanh | 50 | sigmoid | 0.9749 |
| 100 | sigmoid | 50 | relu | 0.9738 |
| 50 | relu | 100 | relu | 0.9738 |
| 50 | relu | 50 | tanh | 0.9731 |
| 50 | tanh | 100 | relu | 0.9729 |
| 50 | tanh | 50 | relu | 0.9729 |
| 100 | sigmoid | 50 | sigmoid | 0.9727 |
| 50 | tanh | 100 | sigmoid | 0.9722 |
| 50 | relu | 50 | relu | 0.9721 |
| 50 | tanh | 50 | sigmoid | 0.9719 |
| 50 | relu | 100 | sigmoid | 0.9716 |
| 50 | tanh | 50 | tanh | 0.9713 |
| 50 | sigmoid | 100 | relu | 0.9709 |
| 50 | tanh | 100 | tanh | 0.9704 |
| 50 | sigmoid | 50 | tanh | 0.9704 |
| 50 | relu | 50 | sigmoid | 0.9698 |
| 50 | sigmoid | 100 | tanh | 0.9695 |
| 50 | sigmoid | 50 | relu | 0.9688 |
| 50 | sigmoid | 100 | sigmoid | 0.9669 |
| 50 | sigmoid | 50 | sigmoid | 0.9648 |

The model with highest accuracy had 100 neurons in the first hidden layer and 50 neurons in the second hidden layer with ReLU as activation function for both layers. All the 27 ANNs had similar performance, the difference in their accuracies were not statistically significant.

Here are the confusion matrices for all the classifiers:

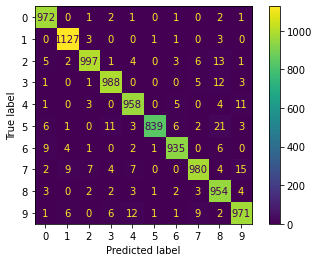


Figure : 50\_relu\_50\_relu

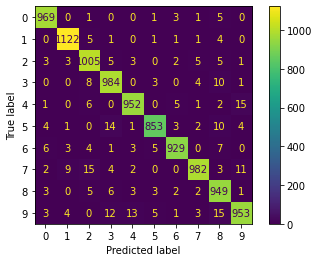


Figure : 50\_relu\_50\_sigmoid

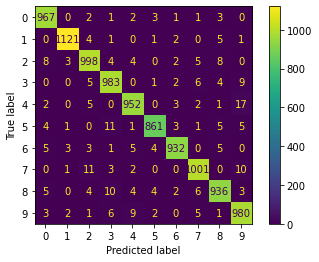


Figure : 50\_relu\_50\_tanh

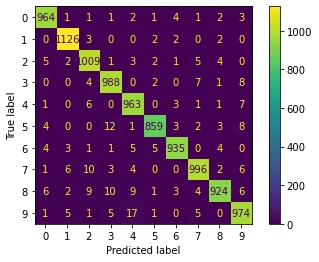


Figure : 50\_relu\_100\_relu

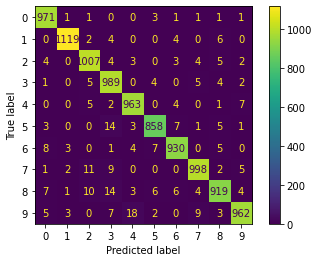


Figure : 50\_relu\_100\_sigmoid

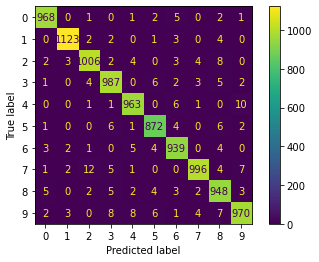


Figure : 50\_relu\_100\_tanh

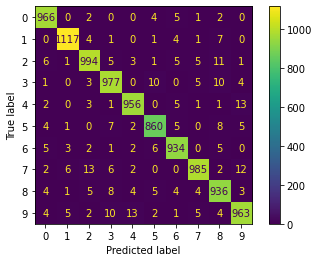


Figure : 50\_sigmoid\_50\_relu

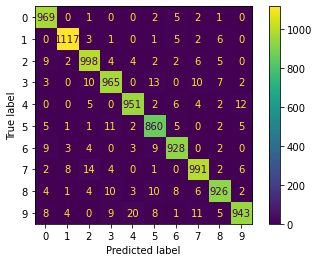


Figure : 50\_sigmoid\_50\_sigmoid

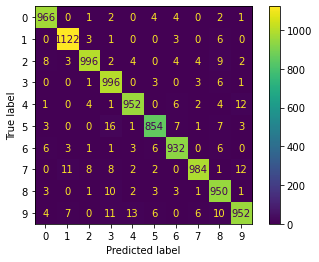


Figure : 50\_sigmoid\_50\_tanh

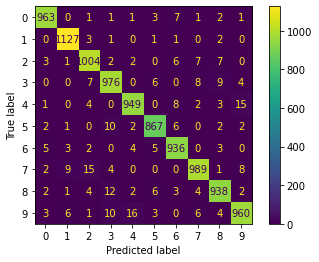


Figure : 50\_sigmoid\_100\_relu

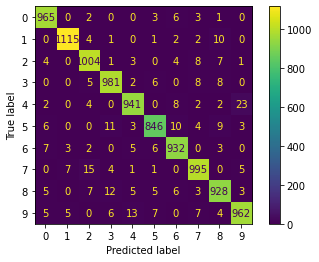


Figure : 50\_sigmoid\_100\_sigmoid

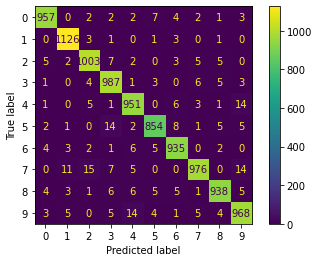


Figure : 50\_sigmoid\_100\_tanh

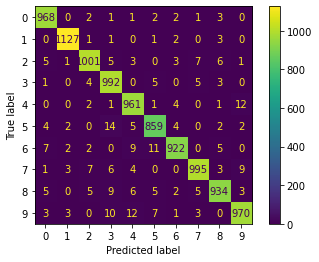


Figure : 50\_tanh\_50\_relu

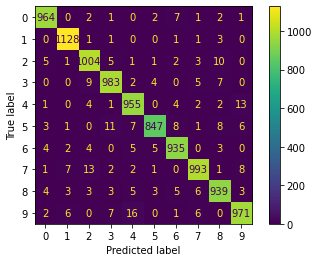


Figure : 50\_tanh\_50\_sigmoid

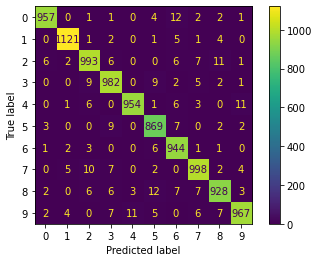


Figure : 50\_tanh\_50\_tanh

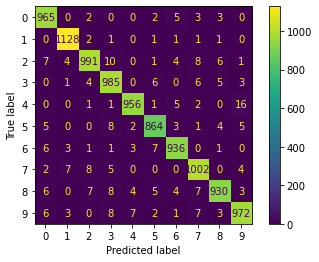


Figure : 50\_tanh\_100\_relu

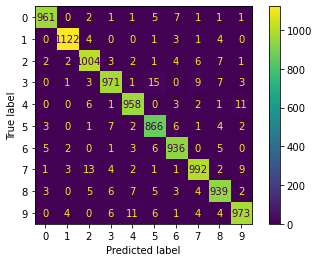


Figure : 50\_tanh\_100\_sigmoid

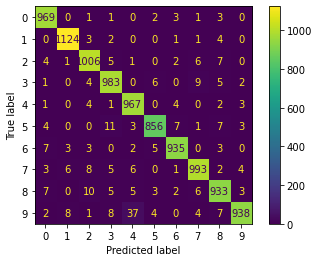


Figure : 50\_tanh\_100\_tanh

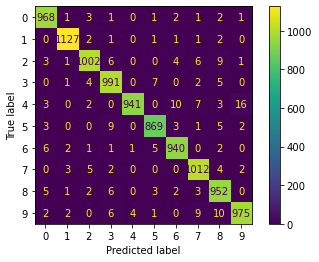


Figure : 100\_relu\_50\_relu

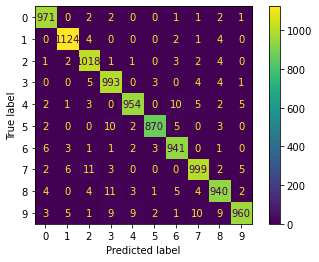


Figure : 100\_relu\_50\_sigmoid

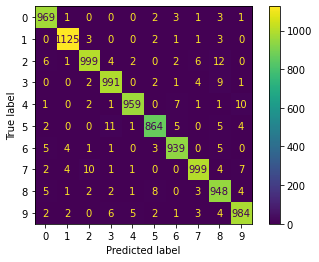


Figure : 100\_relu\_50\_tanh

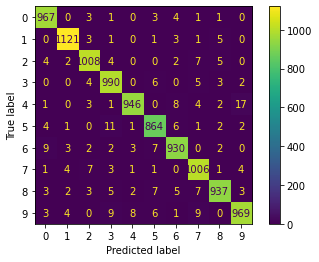


Figure : 100\_sigmoid\_50\_relu

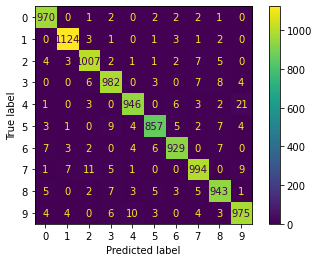


Figure : 100\_sigmoid\_50\_sigmoid

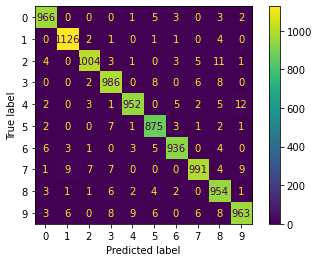


Figure : 100\_sigmoid\_50\_tanh

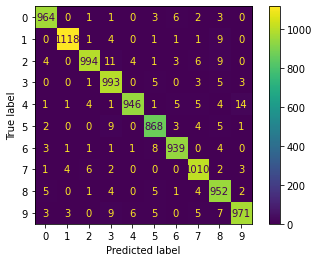


Figure : 100\_tanh\_50\_relu

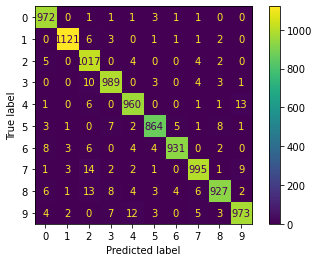


Figure : 100\_tanh\_50\_sigmoid

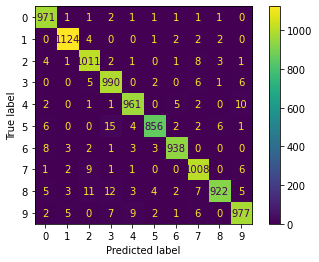


Figure : 100\_tanh\_50\_tanh