BITS F464

MACHINE LEARNING

ASSIGNMENT - 2

Visweswar Sirish Parupudi
2020AAPS0330H

Kedarnath Senegavaram
2020A7PS0245H

Konkala Rithvik
2020A8PS0517H

Part A - Naive Bayes Classifier to predict income

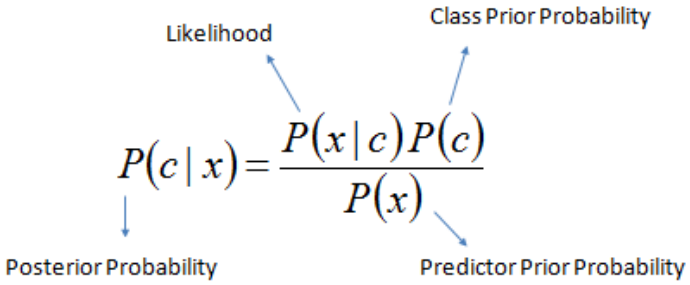
TASK 1: DATA PREPROCESSING

- The adult.data file provided in the UCI income dataset was loaded into the notebook as a pandas DataFrame and appropriate preprocessing was carried out.
- Missing values in the data were denoted by "?" in the dataset. These values were found and replaced with the most commonly occurring (mode) value in the respective feature column.
- Label encoding was further applied to the text features i.e each unique value if each categorical feature column was replaced with integers starting from 0, ending with [max(featureCol.unique)-1]
- The target feature was also changed to 0 and 1, 0 being less than 50k annual income and 1 above.
- The "final weight" and "number of years of education" features were both dropped. Final weight is essentially a unique value calculated from the census and holds no relevance to income, and number of years of education is a duplicate of the education feature.
- A list was initialized with 10 random integers, which were used as seeds for our 10 random testing and training dataset splits. 67 per cent of the total data was allocated to each training split and 33 per cent to each testing split.

TASK 2: Naive Bayes Classifier Implementation

- Bayes theorem provides a way of calculating the posterior probability, P(c|x), from P(c), P(x), and P(x|c). Naive Bayes classifier assumes that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.
- $P(c|x)$ is the posterior probability of *class* (*target*) given *predictor* (*attribute*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of the *predictor* given *class*.

- *P*(*x*) is the prior probability of the *predictor*.



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

- In the above expression for ***P(c|X)***, it is possible that one of the features $\boldsymbol{x_i}$ on the RHS of the equation may not belong to the training data. In such a scenario, the likelihood ***P(x_i | c)*** equates to 0 and this is a problem. In order to fix this issue of zero probability we use Laplace smoothing, in which each likelihood value is calculated as follows:

$$P(x_i \mid c) = \frac{(number\ of\ examples\ with\ x_i\ given\ c) + \alpha}{N + \alpha^* k}$$

- Here, N = number of examples for a given c, and k = total number of features
- A class called NaiveBayes was implemented in the code with a fit method to fit the model to the training data and a predict method to generate classifications for the given test data, and compare the accuracy of the model. This class contains the methods to calculate prior probability, posterior probability, likelihood and the added provision to provide smoothing parameter α if needed.

TASK 3: Evaluation and Improvement

- 10 models were fitted to the 10 training splits for both smoothed and unsmoothed implementation. Predictions were made for both on the testing input and they were compared to the testing classification by using the following metrics:
  1. Accuracy
  2. Precision
  3. Recall
  4. F1 score
- For further comparison, logistic regression and K nearest neighbor models were also implemented for each split, and these were directly imported using sci-kit learn.
- We observe that the general accuracy values lie in the 79%-80% range for naive bayes, and that slightly lesser for naive bayes with Laplace smoothing. This might be the case because Laplace smoothing simply adds 1 to the numerator in the likelihood calculation, and this leads to noise which affects our data.
- We observe that the metrics are slightly improved for Logistic Regression(LR), and there is a further improvement in the K-nearest neighbors models. The slight jump in the metrics for LR as compared to Naive Bayes is because there is no inherent assumption in LR that the features are independent, and they are not in the dataset either.
- The further jump in the metrics for KNN can be attributed to the fact that the training dataset is comparatively smaller in size, and this works in its advantage as it makes predictions based on the proximity to existing examples, rather than fitting a model to the data. It also may be due to the unbalanced nature of the dataset, with only 23% of the total examples belonging to the class earning more than 50k annually. k-NN assigns equal weight to each training example, whereas logistic regression can be biased towards the majority class.
- The performance metrics of each model for each dataset split and the average and variance of the metrics for the respective models are mentioned in the table below.

METRIC	DATASET SPLIT	Naive Bayes No Smoothing	Naive Bayes Laplace Smoothing	Logistic Regression	K Nearest Neighbours
ACCURACY	SPLIT 1	79.7766403	79.52536063	80.01861331	81.75895765
	SPLIT 2	80.26989297	80.01861331	80.24197301	82.65239646
	SPLIT 3	79.72080037	79.51605398	80.06514658	82.33597022
	SPLIT 4	79.43229409	79.18101443	79.70218706	82.57794323
	SPLIT 5	79.41368078	79.20893439	79.81386691	82.53140996
	SPLIT 6	80.05583993	79.85109353	80.47463937	83.15495579
	SPLIT 7	79.91624011	79.58120056	80.27919963	82.97812936
	SPLIT 8	79.70218706	79.36714751	79.79525361	82.63378315
	SPLIT 9	79.83248022	79.53466729	79.7766403	82.14053048
	SPLIT 10	79.88832015	79.62773383	79.87901349	82.89436947
	AVERAGE	**79.8008376**	**79.54118195**	**80.00465333**	**82.56584458**
	VARIANCE	**0.067716604**	**0.060439985**	**0.059594634**	**0.152319919**
PRECISION	SPLIT 1	0.685169843	0.670178282	0.704407952	0.63253012
	SPLIT 2	0.705272256	0.689509306	0.708849558	0.644295302
	SPLIT 3	0.656146179	0.644607843	0.685868587	0.636024845
	SPLIT 4	0.663406683	0.64888535	0.686308492	0.648811921
	SPLIT 5	0.675313808	0.663105998	0.711151737	0.652384802
	SPLIT 6	0.670311186	0.657873042	0.712665406	0.652353427
	SPLIT 7	0.662369057	0.643921569	0.68792517	0.651993355
	SPLIT 8	0.677664975	0.657119476	0.692307692	0.648006444
	SPLIT 9	0.668664384	0.651085142	0.672291297	0.628502802
	SPLIT 10	0.653878232	0.638956805	0.658347676	0.642621599
	AVERAGE	**0.67181966**	**0.656524281**	**0.692012357**	**0.643752462**
	VARIANCE	**0.000231506**	**0.000224147**	**0.000315479**	**7.55426E-05**
RECALL	SPLIT 1	0.315648855	0.315648855	0.311068702	0.601145038
	SPLIT 2	0.314450867	0.314065511	0.30867052	0.628901734
	SPLIT 3	0.309197652	0.308806262	0.298238748	0.601174168
	SPLIT 4	0.311757947	0.312140942	0.303332057	0.617004979
	SPLIT 5	0.30672748	0.30672748	0.295705055	0.61345496
	SPLIT 6	0.312794349	0.313186813	0.295918367	0.620094192
	SPLIT 7	0.320968372	0.320577899	0.31589223	0.613041781
	SPLIT 8	0.307958478	0.308727413	0.297577855	0.618608228
	SPLIT 9	0.304959	0.304568528	0.295587661	0.613041781
	SPLIT 10	0.309881423	0.309881423	0.302371542	0.616205534
	AVERAGE	**0.311434442**	**0.311433113**	**0.302436274**	**0.614267239**
	VARIANCE	**2.25882E-05**	**2.19778E-05**	**5.23526E-05**	**6.95002E-05**
F1 SCORE	SPLIT 1	0.432192318	0.429164504	0.431559439	0.616438356
	SPLIT 2	0.434968017	0.431559439	0.430067114	0.63650546
	SPLIT 3	0.420324554	0.417570786	0.415711948	0.618108652
	SPLIT 4	0.42417926	0.421515387	0.420717131	0.632508834
	SPLIT 5	0.421850497	0.419438669	0.417718121	0.632321254
	SPLIT 6	0.426545357	0.424355225	0.418191902	0.635814889
	SPLIT 7	0.432403998	0.428050052	0.432967621	0.631917891
	SPLIT 8	0.423473434	0.420088935	0.416240925	0.632966168
	SPLIT 9	0.418879056	0.41500399	0.41063195	0.620676023
	SPLIT 10	0.420488067	0.417354272	0.414409534	0.6291364
	AVERAGE	**0.425530456**	**0.422410126**	**0.420821569**	**0.628639393**
	VARIANCE	**3.31212E-05**	**3.15304E-05**	**6.18777E-05**	**5.49761E-05**

https://docs.google.com/spreadsheets/d/125HeGt-AjShWDGZ_yOS_xCL7q9pdqfxiXmTBspmrGHA/edit?usp=sharing

Part B - Building a Basic Neural Network for Image Classification

METRIC	DATASET/MODEL	MODEL 1	MODEL 2	MODEL 3	MODEL 4	MODEL 5	MODEL 6	MODEL 7	MODEL 8	MODEL 9	MODEL 10	MODEL 11	MODEL 12	MODEL 13	MODEL 14	MODEL 15
ACCURACY	SPLIT 1	0.96896104	0.95774892	0.96800866	0.96701299	0.95948052	0.96917749	0.96991342	0.95753247	0.9665368	0.95701299	0.94363636	0.96090909	0.97012987	0.9621645	0.96497835
	SPLIT 2	0.96774892	0.95632035	0.96909091	0.96852814	0.96056277	0.9678355	0.96627706	0.95935065	0.96822511	0.95805195	0.94701299	0.96038961	0.97090909	0.9625974	0.96874459
	SPLIT 3	0.96385281	0.95896104	0.97021645	0.96588745	0.9612987	0.97229437	0.96714286	0.9604329	0.97121212	0.95679654	0.94640693	0.96380952	0.97073593	0.96316017	0.97012987
	SPLIT 4	0.96809524	0.95718615	0.96935065	0.96891775	0.96121212	0.97181818	0.96857143	0.96034632	0.96809524	0.96082251	0.94082251	0.96251082	0.97151515	0.9612987	0.97103896
	SPLIT 5	0.96935065	0.95839827	0.97047619	0.96857143	0.96251082	0.96904762	0.97077922	0.95891775	0.9687013	0.95813853	0.94722944	0.96025974	0.97277056	0.96177489	0.97225108
	SPLIT 6	0.96991342	0.96	0.96761905	0.96852814	0.96012987	0.96670996	0.97004329	0.95757576	0.96705628	0.95974026	0.94458874	0.96367965	0.96987013	0.96290043	0.97069264
	SPLIT 7	0.96627706	0.95883117	0.96588745	0.96679654	0.96012987	0.96619048	0.9691342	0.95519481	0.96688312	0.95896104	0.94844156	0.96017316	0.96978355	0.9608658	0.96818182
	SPLIT 8	0.96800866	0.95731602	0.96584416	0.9687013	0.95965368	0.96969697	0.96787879	0.95787879	0.96813853	0.95974026	0.94506494	0.96281385	0.97112554	0.96082251	0.9674026
	SPLIT 9	0.96900433	0.95640693	0.9669697	0.96601732	0.95926407	0.96679654	0.96532468	0.96004329	0.96705628	0.95852814	0.94558442	0.96147186	0.96939394	0.96190476	0.9674026
	SPLIT 10	0.96571429	0.95627706	0.96818182	0.96874459	0.96090909	0.96805195	0.96679654	0.95831169	0.96757576	0.96138528	0.94645022	0.95714286	0.96965368	0.96212121	0.96861472
	AVERAGE	0.96769264	0.95774459	0.9681645	0.96777056	0.96051515	0.9687619	0.96818615	0.95855844	0.96794805	0.95891775	0.94552381	0.96131602	0.97058874	0.96196104	0.96894372
	VARIANCE	3.5486E-06	1.6268E-06	2.682E-06	1.4514E-06	9.9051E-07	4.3328E-06	3.2389E-06	2.6141E-06	1.8012E-06	2.3009E-06	4.6702E-06	4.0401E-06	1.0787E-06	6.4052E-07	4.5552E-06
PRECISION	SPLIT 1	0.96877432	0.95733377	0.9675518	0.96664387	0.95890497	0.96872986	0.96951603	0.95689039	0.9660898	0.95696928	0.94294228	0.96049169	0.96971031	0.96175414	0.965632
	SPLIT 2	0.96754546	0.9561481	0.96907243	0.96839484	0.96026006	0.96800242	0.96603073	0.95900546	0.96812973	0.95753093	0.94650549	0.96003403	0.97099451	0.96224609	0.96903892
	SPLIT 3	0.96413134	0.95878162	0.97006738	0.96571397	0.96115593	0.97214951	0.96701067	0.96013108	0.97118733	0.95692983	0.94605144	0.96363786	0.97044777	0.96293087	0.97014407
	SPLIT 4	0.9676962	0.95702387	0.96907409	0.96868007	0.96077213	0.97151097	0.96840209	0.95992073	0.96811237	0.96070107	0.94014891	0.9622573	0.97122905	0.96104037	0.97064229
	SPLIT 5	0.96922912	0.95818117	0.97040162	0.96861357	0.96225883	0.96906912	0.97068924	0.95868472	0.96899068	0.95781891	0.94710064	0.96057918	0.97269682	0.96183267	0.97222656
	SPLIT 6	0.96962413	0.95980228	0.96747845	0.96843798	0.9598669	0.96636808	0.96976904	0.95748161	0.9670122	0.95936234	0.94394335	0.96384987	0.96962616	0.96279001	0.97076753
	SPLIT 7	0.96599961	0.95837919	0.96558955	0.96649575	0.9599232	0.96622076	0.96904537	0.95497291	0.96659307	0.95861224	0.94778926	0.95962498	0.96966975	0.9604892	0.96807681
	SPLIT 8	0.96773834	0.95706255	0.96561657	0.96826029	0.95921141	0.96956121	0.96751196	0.95753533	0.96793024	0.95934564	0.94443917	0.96247434	0.97094942	0.96054601	0.96773387
	SPLIT 9	0.96876393	0.95597918	0.96698302	0.96600443	0.95908146	0.96666065	0.96541404	0.95974478	0.96707001	0.95812847	0.94491183	0.96107328	0.96939312	0.9616768	0.96755366
	SPLIT 10	0.96553183	0.95604484	0.96819692	0.9685021	0.96087065	0.96770884	0.96659719	0.95778295	0.96712058	0.96118995	0.94596901	0.95744317	0.96943441	0.96178019	0.96884646
	AVERAGE	0.96750343	0.95747366	0.96800318	0.96757469	0.96023055	0.96859814	0.96799864	0.958215	0.9678236	0.95865886	0.94498014	0.96114657	0.97041513	0.96170864	0.96906622
	VARIANCE	3.1437E-06	1.653E-06	2.8203E-06	1.4453E-06	1.113E-06	4.1987E-06	3.0738E-06	2.5799E-06	2.1229E-06	2.1832E-06	5.0784E-06	3.8373E-06	1.1302E-06	6.9566E-07	3.7125E-06
RECALL	SPLIT 1	0.96858176	0.95713783	0.96778191	0.96658807	0.95899317	0.96897182	0.96950999	0.95717452	0.96641539	0.95654916	0.9427791	0.96058325	0.96989407	0.96172216	0.96460201
	SPLIT 2	0.96731546	0.95599871	0.96879967	0.96816893	0.96008963	0.96741222	0.96602121	0.95899599	0.96784691	0.95784	0.9464074	0.96025132	0.97039386	0.96224525	0.96834665
	SPLIT 3	0.96378045	0.95869801	0.97001997	0.96557372	0.96089151	0.97210411	0.96693197	0.96033308	0.97099811	0.95630917	0.94605509	0.9635278	0.97061734	0.96288191	0.96993029
	SPLIT 4	0.96791249	0.95669667	0.96927143	0.96867292	0.9608806	0.97177202	0.96821123	0.96025126	0.96791419	0.96030037	0.94037316	0.96221643	0.97146468	0.96102164	0.97095298
	SPLIT 5	0.96921383	0.95822053	0.9703872	0.96836009	0.9625165	0.96887244	0.97069003	0.95871997	0.96829071	0.95792896	0.94693055	0.95991198	0.97257115	0.96146869	0.97217546
	SPLIT 6	0.96976058	0.95967906	0.96736849	0.96827564	0.95984895	0.96664715	0.96991568	0.95724163	0.96679413	0.95947625	0.94445156	0.96332095	0.96966817	0.96251575	0.9704677
	SPLIT 7	0.96592298	0.95839491	0.96557248	0.96638016	0.95963289	0.96595666	0.96872821	0.95487	0.96646184	0.95847383	0.9477666	0.95988066	0.9694631	0.96046213	0.96767087
	SPLIT 8	0.96786062	0.95703242	0.96565062	0.96864905	0.95943779	0.96959812	0.96774377	0.95760043	0.96821755	0.95951989	0.94468625	0.96259722	0.97102021	0.96056595	0.96720148
	SPLIT 9	0.96882131	0.95609345	0.96669585	0.96572375	0.95885765	0.96660637	0.96490274	0.95980036	0.96682436	0.95821791	0.94523696	0.96142807	0.96922788	0.96155607	0.96711918
	SPLIT 10	0.96513602	0.95567388	0.96761218	0.96832207	0.96026521	0.96790698	0.96634751	0.9577638	0.96718953	0.96084162	0.945864	0.95640954	0.96925444	0.96157292	0.96814688
	AVERAGE	0.96743055	0.95736255	0.96791598	0.96747144	0.96014139	0.96858479	0.96790023	0.9582751	0.96769527	0.95854572	0.94505507	0.96101272	0.97035749	0.96160125	0.96866135
	VARIANCE	3.6844E-06	1.7641E-06	2.8527E-06	1.5671E-06	1.1749E-06	4.4682E-06	3.4564E-06	2.8639E-06	1.8436E-06	2.2383E-06	4.6652E-06	4.4703E-06	1.1796E-06	6.2573E-07	4.9785E-06

https://docs.google.com/spreadsheets/d/1Cl1WXvyqfqyXEh9MxBrM0qZQBRDDGFuefZtCwF8ZjbU/edit?usp=sharing

MODEL 1: 2 LAYERS (50,50) TANH ACTIVATION

MODEL 2: 2 LAYERS (50,50) SIGMOID ACTIVATION

MODEL 3: 2 LAYERS (50,50) RELU ACTIVATION

MODEL 4: 2 LAYERS (75,25) TANH ACTIVATION

MODEL 5: 2 LAYERS (75,25) SIGMOID ACTIVATION

MODEL6: 2 LAYERS (75,25) RELU ACTIVATION

MODEL 7: 3 LAYERS (50,50,50) TANH ACTIVATION

MODEL 8: 3 LAYERS (50,50,50) SIGMOID ACTIVATION

MODEL 9: 3 LAYERS (50,50,50) RELU ACTIVATION

MODEL 10: 3 LAYERS (25,100,25) TANH ACTIVATION

MODEL 11: 3 LAYERS (25,100,,25) SIGMOID ACTIVATION

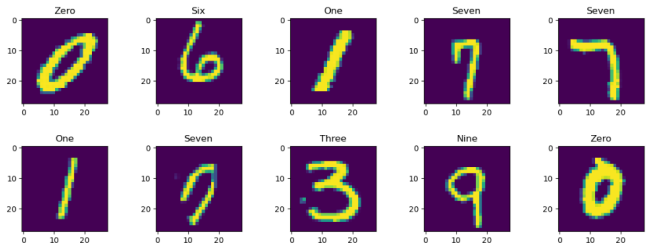
MODEL 12: 3 LAYERS (25,100,,25) RELU ACTIVATION

MODEL 13: 3 LAYERS (75,50,,25) TANH ACTIVATION

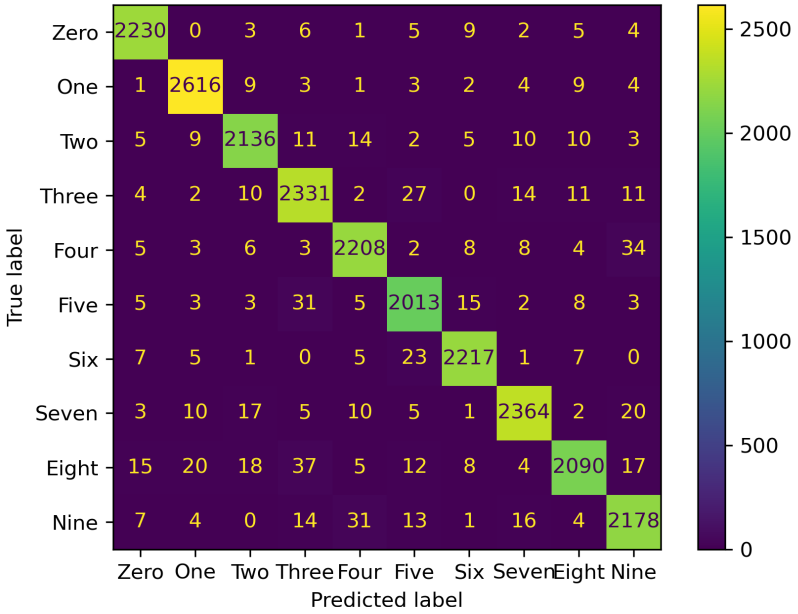
MODEL 14: 3 LAYERS (75,50,,25) SIGMOID ACTIVATION

MODEL 15: 3 LAYERS (75,50,,25) RELU ACTIVATION

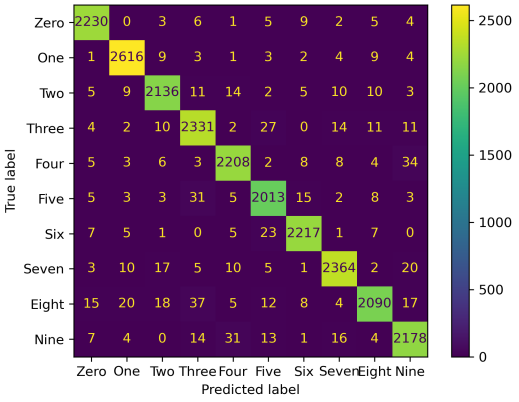
- 10 train test splits were implemented with a 67-33 split on the **MNIST dataset**.



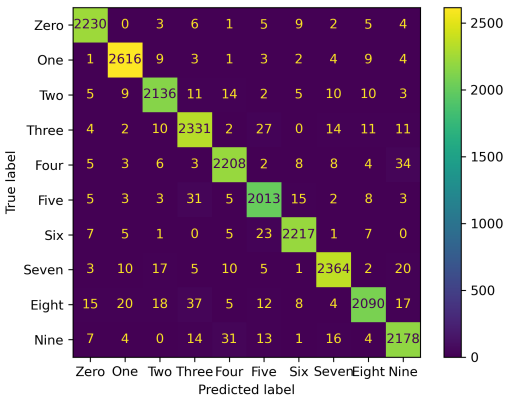
- 15 models were implemented for each of these splits and their metrics were calculated and tabulated as shown above.
- Keras was used to build the architectures and was fitted on the training set for 10 epochs, Adam optimiser and sparse categorical cross-entropy loss.
- Better accuracies were achieved for more hidden layers and activation functions like tanh and ReLu than sigmoid, which gave considerably lesser accuracies.
- The highest accuracy was achieved by MODEL 13 with 3 hidden layers and tanh activation.
- Model 6 also performed well with only 2 layers and ReLu activation.
- 150 confusion matrices were generated, and the ones of the best models are shown below.
- The average and variance of the performance metrics for each model are also recorded.



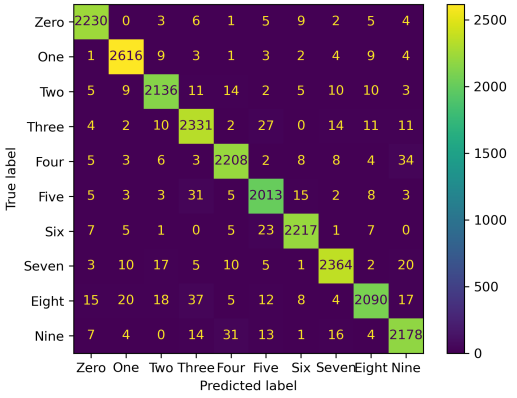
MODEL 13: 3 LAYERS (75,50,,25) TANH ACTIVATION (97.27%)



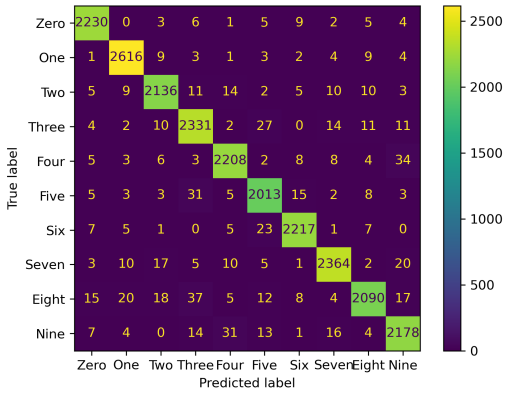
MODEL6: 2 LAYERS (75,25) RELU ACTIVATION(97.22%)



MODEL 7: 3 LAYERS (50,50,50) TANH ACTIVATION(97.07%)



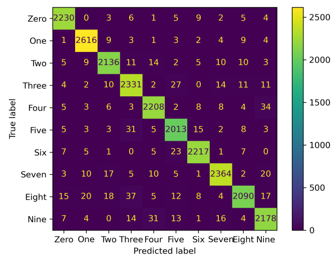
MODEL 15: 3 LAYERS (75,50,,25) RELU ACTIVATION(97.22%)

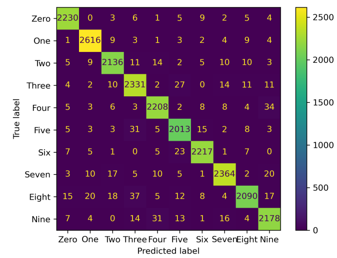


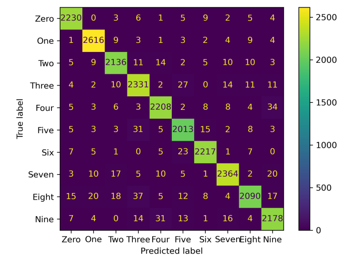
MODEL 9: 3 LAYERS (50,50,50) RELU ACTIVATION(97.12%)

- The above models gave some of the better results and accuracies, as expected since ReLu and TanH are better activation functions, while more neurons/more hidden layers are able to extract the feature of the images better into 10 class outputs.
- The output layer is using softmax activation for classification nad sparse categorical cross-entropy loss is sued when we have a list of multiple classes. Categorical cross entropy can be used if our outputs are one-hot encoded.
- Given below are 15 confusion matrices for one split of training test data for each model.

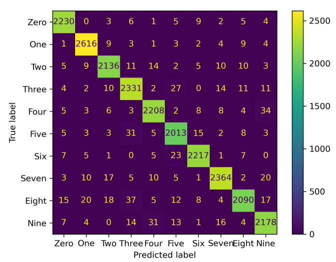
MODEL 1: 2 LAYERS (50,50) TANH ACTIVATION	MODEL 2: 2 LAYERS (50,50) SIGMOID ACTIVATION	MODEL 3: 2 LAYERS (50,50) RELU ACTIVATION



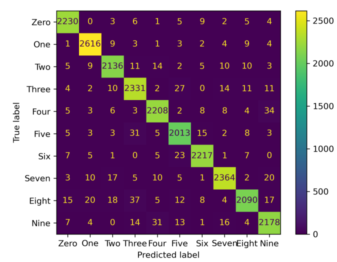




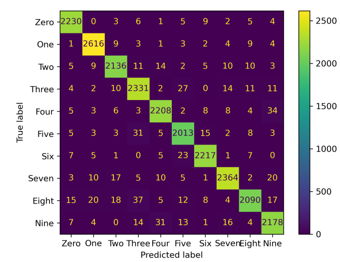
MODEL 4: 2 LAYERS (75,25) TANH ACTIVATION



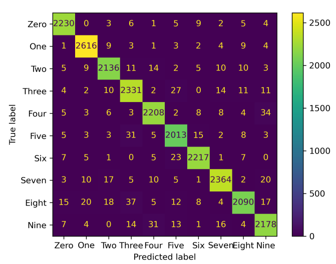
MODEL 5: 2 LAYERS (75,25) SIGMOID ACTIVATION



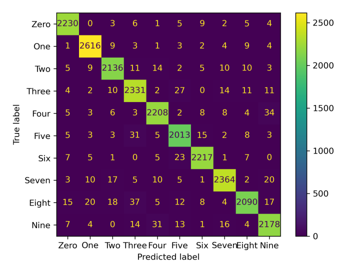
MODEL6: 2 LAYERS (75,25) RELU ACTIVATION



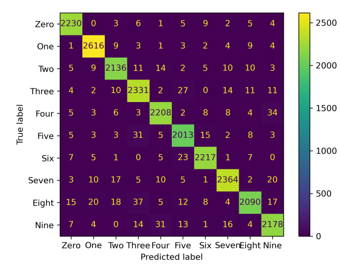
MODEL 7: 3 LAYERS (50,50,50) TANH ACTIVATION



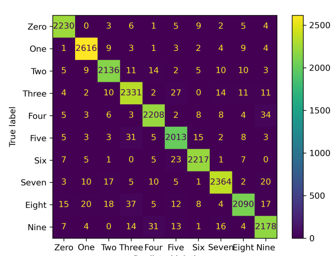
MODEL 8: 3 LAYERS (50,50,50) SIGMOID ACTIVATION



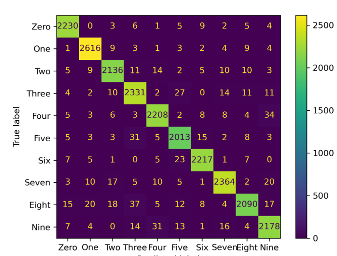
MODEL 9: 3 LAYERS (50,50,50) RELU ACTIVATION



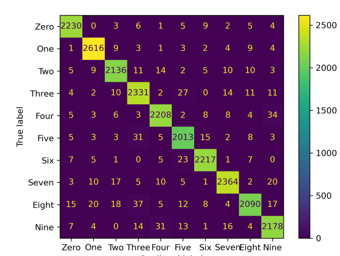
MODEL 10: 3 LAYERS (25,100,25) TANH ACTIVATIO



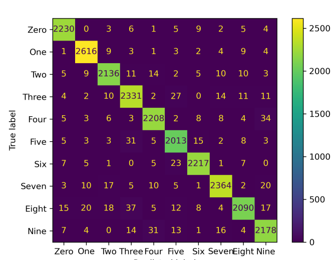
MODEL 11: 3 LAYERS (25,100,,25) SIGMOID ACTIVATION



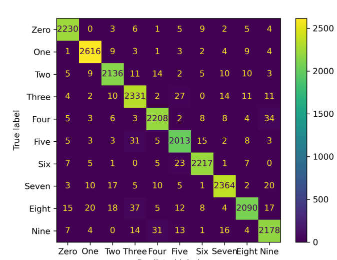
MODEL 12: 3 LAYERS (25,100,,25) RELU ACTIVATION



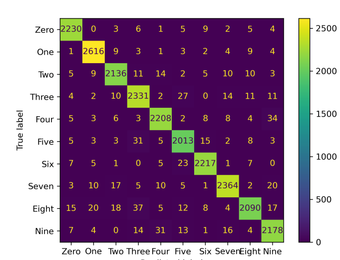
MODEL 13: 3 LAYERS (75,50,,25) TANH ACTIVATION



MODEL 14: 3 LAYERS (75,50,,25) SIGMOID ACTIVATION



MODEL 15: 3 LAYERS (75,50,,25) RELU ACTIVATION



- Thus we have achieved the highest accuracy at **MODEL 13: 3 LAYERS (75,50,,25) TANH ACTIVATION with an accuracy of 97.27%.**
- Models with ReLu activation also performed with up to 97% accuracy.
- Architectures with SIGMOID activation are not statistically significant from the best classifiers.
- All 150 images for the confusion matrices are available in the zip folder provided.