Since the dataset is considered imbalanced, then any kind of alternative methods such as oversampling technique and the choice of different Machine Learning algorithms are implemented as well. The last step would be observing the accuracy and the F1 score of the model.

B. *Task II*

The first step that should be done in an image classification task is ingesting the training and validation data, and then preprocess them straight away such that they have a consistent size. Next, the pixel value in each of the images needs to be normalized as well.

After the preprocessing step is done, then an appropriate CNN model needs to be built. For this project, there are two CNN models that have been implemented. The first model is a custom model, meaning that the CNN model is built from the scratch. Meanwhile, the second model is built by utilizing the transfer learning method from InceptionV3 model, in which its weight has been trained on ImageNet dataset.

After each training, the learning curve needs to be observed in order to track the performance of the model. Finally, the prediction based on unseen images can be performed.

IV. EXPERIMENTS

A. *Task I*

As mentioned in Section III, the first step is to impute missing values with the median value. After the imputation, below is the updated statistical analysis of the numerical data.

	employee_id	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?	avg_training_score	is_promoted
count	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000	54808.000000
mean	39195.830627	1.253011	34.803915	3.329256	5.865512	0.023172	63.386750	0.085170
std	22586.581449	0.609264	7.660169	1.211661	4.265094	0.150450	13.371559	0.279137
min	1.000000	1.000000	20.000000	1.000000	1.000000	0.000000	39.000000	0.000000
25%	19669.750000	1.000000	29.000000	3.000000	3.000000	0.000000	51.000000	0.000000
50%	39225.500000	1.000000	33.000000	3.000000	5.000000	0.000000	60.000000	0.000000
75%	58730.500000	1.000000	39.000000	4.000000	7.000000	0.000000	76.000000	0.000000
max	78298.000000	10.000000	60.000000	5.000000	37.000000	1.000000	99.000000	1.000000

Now that there are no missing values anymore, the question that needs to be answered next is: what Machine Learning algorithms will suit the dataset best? To achieve this, we can do grid searching for every Machine Learning classifier out there, but it is not a straightforward method. The simpler method is by utilizing PyCaret library. The library will show the best algorithms for the given dataset in just a few minutes.

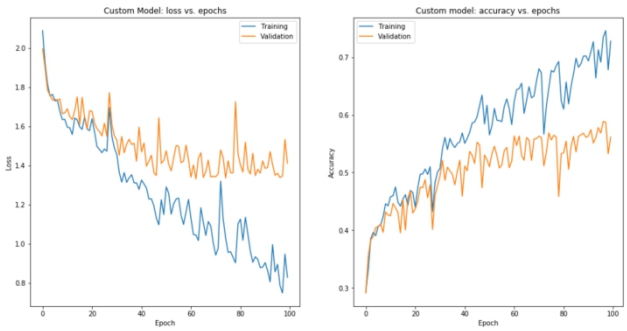
	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
catboost	CatBoost Classifier	0.9420	0.8236	0.6742	0.9415	0.9296	0.4854	0.5521	24.8370
lightgbm	Light Gradient Boosting Machine	0.9411	0.8183	0.6678	0.9410	0.9281	0.4725	0.5434	0.3860
xgboost	Extreme Gradient Boosting	0.9410	0.8109	0.6725	0.9395	0.9287	0.4790	0.5429	3.9450
gbc	Gradient Boosting Classifier	0.9394	0.8154	0.6550	0.9402	0.9250	0.4457	0.5256	4.6440
lda	Linear Discriminant Analysis	0.9380	0.7720	0.6624	0.9341	0.9249	0.4516	0.5123	0.4410
rf	Random Forest Classifier	0.9325	0.7826	0.6212	0.9300	0.9144	0.3605	0.4472	2.5150
ridge	Ridge Classifier	0.9264	0.0000	0.5724	0.9319	0.9003	0.2360	0.3654	0.0590
ada	Ada Boost Classifier	0.9263	0.7907	0.5849	0.9212	0.9031	0.2658	0.3640	1.1800
et	Extra Trees Classifier	0.9257	0.7692	0.6148	0.9130	0.9086	0.3270	0.3820	3.0390
nb	Naive Bayes	0.9154	0.7163	0.5086	0.9226	0.8765	0.0310	0.1214	0.0670
lr	Logistic Regression	0.9140	0.5739	0.5000	0.8353	0.8729	0.0000	0.0000	1.2460
knn	K Neighbors Classifier	0.9097	0.5308	0.5033	0.8499	0.8727	0.0113	0.0227	0.7950
dt	Decision Tree Classifier	0.8849	0.6655	0.6655	0.8920	0.8882	0.3114	0.3122	0.3390
svm	SVM - Linear Kernel	0.8252	0.0000	0.5054	0.8505	0.7871	0.0162	0.0255	1.2090
qda	Quadratic Discriminant Analysis	0.4705	0.4972	0.4972	0.8431	0.5458	-0.0006	-0.0013	0.1890

From the table, it can be seen that the CatBoost classifier gave the best performance in every metric given the dataset. With PyCaret, the CatBoost classifier can be built straight away and the task is done. However, the main focus now is to try whether it is possible to achieve those results with scikit-learn pipeline.

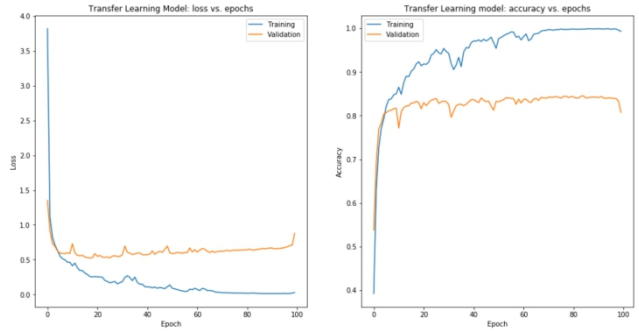
After implementing the pipeline to encode categorical features and to normalize numerical features, the Logistic Regression and the CatBoost classifier have been built and it can be concluded that the model gave roughly the same value of accuracy compared to the PyCaret model. However, the F1 score of the model needs to be improved, since it only has 52%. This phenomenon occurs because of the imbalanced dataset. Oversampling technique and a random forest classifier with adjusted weight have been implemented to try to overcome this problem, but they did not help to improve the F1 score.

B. *Task II*

Two different CNN models have been implemented for this task. One is the custom model and the other is the model based on transfer learning from InceptionV3 model. Both models have been trained in 100 epochs.



From the figure above, it can be seen that the performance of the custom model was not satisfying. Its training accuracy after 100 epochs is just around 73%, while the validation accuracy is just 54%. This is understandable since this model only consists of 4 Convolutional layers before the final dense layer. Hence, it is not sufficient to extract meaningful features from the images.



Meanwhile, the transfer learning model performed much better than the custom model. It achieved 99% accuracy in the training set and 83% accuracy in the validation set. This is understandable since this model is much deeper than the custom model. Also, it has the weight that has been trained on ImageNet, which is a large image dataset.