COP528 Applied Machine Learning Coursework

I. Introduction

This report describes an experiment with two tasks, each involving the application of machine learning algorithms to a different problem. The first half of the report details Task 1, the application of two supervised machine learning methods to predict whether an employee is promoted in a large multinational corporation. The overall purpose of this task was to build a model which could speed up the corporation's first-round promotion process by recommending certain employees for promotion, based on several factors.

A dataset was provided to build and test the promotion prediction model on. This dataset contained 54,808 rows of 13 attributes, where each row represented one employee at the corporation. This dataset needed to be cleaned and modified before building the predictive model, and this cleaning process will be described in detail in later sections of the report. Categorical attributes were converted into numeric attributes, which were then standardized. The dataset was also split into a training set (75% of samples) and a testing set (25%), to allow for testing models on unseen data. Principal component analysis (PCA) was performed before building the models to reduce dimensionality, reducing the risk of overfitting.

Two machine learning models were trained using the training set of employee promotion data and tested on the testing set. The first method attempted was a random forest classifier, which correctly classified 79.6% of cases in the test set. The second method used was a support vector machine (SVM), which correctly classified 92.4% of cases in the test dataset. Significant problems were identified in both models. The SVM method was less accurate but did at least detect most cases where an employee should be promoted - though this came at the expense of incorrectly recommending many employees for promotion who should not be promoted.

The SVM model was more accurate, but closer inspection of classification predictions by the model revealed that the model was highly biased towards not recommending an employee for promotion. With the purpose of the model in mind, the random forest model is arguably the better model - it detects most cases where an employee should be promoted, and any erroneous recommendations for promotion can be rectified in the second stage of the promotion process.

All analysis for both tasks was performed through various Python packages, in Jupyter notebooks

Task 1 is described in more detail in Section II. The code for Task 1 is available at: https://colab.research.google.com/drive/12GjUC9eAKNEHw1jVx7WkBWfdiikN3lwa?usp=sharing

Section III describes the second task undertaken in this project. Here, the objective was to design or modify a convolutional neural network (CNN) architecture for the purpose of classifying a set of images. For this task, a set of images was provided in a zip file. The file was already split into a training set and validation set, and each image was labelled as belonging to one of ten classes.

CNNs are deep learning algorithms which learn the importance of certain aspects within an image and can then use this information to classify images into different classes. CNNs are different from more traditional machine learning methods in that they can learn both feature extraction and classification within the network. This makes CNNs a very useful tool for the purpose of image classification.

In this task, three types of CNN architecture were applied to the provided image set. First, a simple architecture, with three convolutional layers, two pooling layers, and two fully connected layers was built. After ten epochs, this achieved a classification accuracy on the validation set of 0.192, while achieving an accuracy of 0.700 on the training set - evidence of a high level of overfitting. The second CNN architecture used was LeNet [1], another simple architecture which takes 32x32 images as input. The LeNet model achieved 0.457 accuracy on the test set and 0.825 on the training set - a significant improvement, but still with evidence of overfitting. The third architecture used was AlexNet [2], a more complex neural network that used dropout layers to reduce overfitting. The AlexNet model showed the most promise initially, as it achieved an accuracy of 0.494 on the validation set and 0.598 on the training set after ten epochs, at which point it still had not converged.

After this, several modifications were made to the LeNet model to try to improve the overfitting problem and increase validation accuracy. These attempts were mostly unsuccessful. The AlexNet model was also ran for 25 epochs, to see if the model converged at all. Here, it was found that after 25 epochs the AlexNet model had a validation accuracy of 0.542. However, higher accuracy scores had been achieved in previous epochs.

The code for Task 2 is available at: https://colab.research.google.com/drive/1ctaIz_chF8jrVzgaH4CVh0pm9ivDXKJ4?usp=sharing

II. Task 1: Candidate promotion prediction

A. Data and Preliminary Analysis

The promotion dataset contained 54,808 rows of 13 attributes, where each row represented one employee at the corporation. The 13 columns represented various demographic and performance-related attributes describing the employees, such as their age, department and level of education. The binary target attribute, *is_promoted*, denoted whether an employee had received a promotion or not, where 1 represented an employee who had been promoted, and 0 represented an employee who had not been promoted.

The first stage of building a model to predict whether an employee will be promoted was an exploratory data analysis, where attributes were visualised and potential relationships between attributes were explored. At this stage, the *employee_id* attribute was also removed, as it represented a unique identifier for each employee, and so was not relevant for the building of the model.

Numerical attributes were visualised first, with simple histograms. Here, it was evident that only a small proportion of employees were promoted (see Fig. 1).

Relationships between attributes and the target variable were visualised in more detail by using a pair plot (see Appendix A), and then a correlation heatmap (see Fig. 2). This showed little relationship between *is_promoted* and either *age*, *no_of_trainings* or *length_of_service*. However, there was a small correlation between *is_promoted* and *avg_training_score*, *awards_won* and *previous_year_rating*, likely important variables for the model.

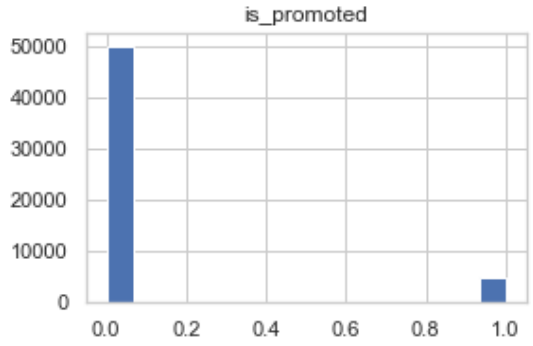


Fig. 1. Histogram of *is_promoted* attribute.

Categorical attributes were visualised using bar charts (see Appendix B). Here, it was possible to see that male employees outnumbered female employees roughly 2:1. Also, the vast majority of employees were educated to at least degree level, and very few employees had been referred to the job.

To explore the categorical attributes in more detail, the proportion of employees who were promoted at each level of a categorical attribute was calculated. Some particularly useful insights were found here. First, for the *education* attribute, employees with a bachelor's degree or below secondary education were similarly likely to get promoted, but employees with a master's degree were around 20% more likely to be promoted. Second, for the *gender* attribute, female employees were slightly more likely to be promoted. This gender discrimination was later removed from the final model. Finally, looking at the *recruitment_channel* attribute, employees who had obtained their job through a referral were around 40% more likely to be promoted.

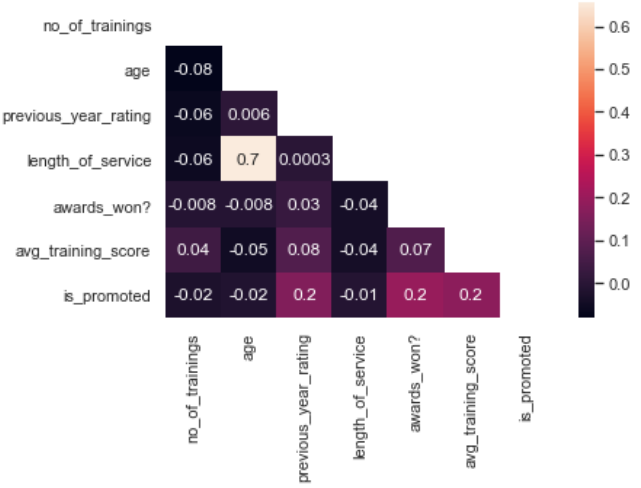


Fig. 2. Correlation heatmap showing correlations between attributes, including the target attribute.

B. *Methods and preprocessing*

To build a model to predict whether an employee is promoted or not, the dataset had to first be cleaned and pre-processed, so its attributes were a suitable format. The first stage of this cleaning process was to deal with null values in the dataset. *Education* and *previous_year_rating* both contained null values. Upon closer inspection, it was clear that all employees with null values for *previous_year_rating* had only been at the job for a year, which meant they could not have a rating for the previous year. These null values were converted to 0 - if a person had only been at the job for a year, it is unlikely they would be promoted, and so it would be better for the promotion prediction model to treat employees with less than a year of service as having a score of zero for *previous_year_rating.*

The null values in *education* were more confusing. It was not clear what, if anything, the null values signified, and so these null values were replaced with 'other'.

At this stage, the *gender* attribute was also removed from the dataset, as it is immoral to consider gender when giving somebody a promotion. *Age*, *no_of_trainings*, and *length_of_service* were also removed, as they correlated with the target variable very weakly.

Some machine learning methods cannot take categorical data as an input so, in the next stage of data pre-processing, categorical attributes were converted into numeric attributes. Of the categorical attributes, only *education* was ordinal. Th three original levels of *education* - 'Master's & above', 'Bachelor's', and 'Below Secondary' had a clear hierarchical relationship, with 'Master's & above' being the highest level for education. However, it was unclear where the 'other' fitted into this relationship. To inspect this further, the proportion of employees at the *education* level 'other' who were promoted was calculated. This proportion was significantly lower than the other levels, so it was concluded that 'other' should be the lowest level of *education*. Thus, *education* was coded numerically with 'other' = 0 and 'Master's & above' = 3.

recruitment_channel was a categorical variable with 3 levels. Employees in the first 2 levels, 'other' and 'sourcing' were almost equally likely to be promoted, but employees in the third level, 'referred', were much more likely to be promoted. Therefore, this attribute was changed to the numeric attribute *is_referred*, where 1 signifies an employee who was referred and 0 signifies anything else.

The final two categorical variables, *department* and *region*, were converted to numeric dummy variables, as they could not be condensed in the same way as *recruitment_channel* was.

The dataset was then split into a training and testing set, with 75% of the dataset allocated to the training set and 25% to the testing set. All processing and model building from this point onwards was performed on the training set, with the test set reserved only for testing models.

The training set was then standardised, as many machine learning methods are sensitive to data range. Each attribute in the dataset was scaled to between 0 and 1. The test set was then standardised, using the same scale as was used to standardise the training set, so that values were consistent across both datasets.

After this stage, the dataset contained 48 attributes. This was rather a lot, so correlations were calculated between all attributes in the training set, to see if any attributes could be removed which correlated strongly with other attributes. The more attributes a dataset has, the higher the risk of overfitting is, and so it was sensible to attempt to reduce dimensionality in this way. However, no attributes correlated particularly strongly with each other here, and so the decision was made to leave the 48 attributes for now.

In the final stage of pre-processing, principal component analysis (PCA) was performed. PCA is another way of reducing dimensionality, by trying to find a way of representing the data on a lower dimensional space. The PCA found that 90% of variance in *is_promoted* in the training set was explained by 27 principal components. New, final training and test datasets were created, using these 27 principal components.

With this final training set, two different machine learning methods were used to build two different models for predicting whether an employee is promoted. The building and testing of these two models is described in the following two subsections.

C. Experiment 1: Random forest classifier

The first machine learning method used was the random forest classifier. This method was chosen as it is robust to noise and has a lower risk of overfitting than a single decision tree.

A random forest classifier was trained on the final training set, firstly using the default parameters. This classified 96% of cases in the training set correctly. Some of the incorrectly classified 4% of cases may have been incorrectly classified because the principal components chosen in the training set only account for 90% of the variance in *is_promoted* in the training set.

It was also possible that a random forest classifier with different parameters may correctly classify more cases in the training set. Therefore, several different random forest classifiers with different parameters were then built, to see if a more accurate model could be found. The accuracy scores of these models are displayed in Figure 3.

Non-default parameter(s)	**Accuracy score**
None (all default parameters)	0.960
n_estimators=500	0.960
min_samples_split=4	0.959
min_samples_leaf=2	0.953
bootstrap=False	0.960

Fig. 3. Accuracy scores of different random forest classifier models on the training dataset.

No random forest classifier tested outperformed the random forest classifier with default settings, and so it was decided to use this default classifier for the prediction of whether an employee was promoted in the test set.

Due to the much higher chance of an employee not being promoted by the corporation compared to the chance of an employee being promoted, the model may have learned to be biased towards class 0 (no promotion). This potential bias was addressed by adjusting the threshold of the classifier at which it predicts an employee is promoted. Through plotting a precision recall curve (see Appendix C) and finding Youden's index, the optimal threshold was found to be at 0.365, which achieved an F-score of 0.725 on the training data. This threshold was used to classify employees on the test set.

The model was then tested on the test set. Several evaluation metrics were calculated, comparing the predictions of this random forest classifier to the ground truth on the test set. The model had an accuracy of 0.796 here, meaning that it correctly classified 79.6% of cases. A confusion matrix was created to show the classifications predicted by the model (see Fig. 4). The model had precision score of 0.229, meaning that most employees predicted as receiving a promotion did not actually receive a promotion. The model had a recall score of 0.599, meaning that it did successfully detect most employees who were promoted. The F1 score, the weighted average of precision and accuracy of the model, was 0.331.

		Predicted	
Ground truth		Not promoted	Promoted
	Not promoted	10217	2329
	Promoted	464	692

Fig. 4. Confusion matrix showing the class predicted by the random forest classifier, and the ground truth.

A ROC curve of false positive (FP) rate against true positive (TP) rate for the test data was plotted (see Appendix D), and the area under the curve (AUC) was found to be 0.76. The model clearly has room for improvement but does show some promise.

D. Experiment 2: Support vector machine classifier

The second machine learning method used was the support vector machine (SVM). This method was chosen as it is effective in high dimensional spaces, such as the promotion dataset.

A SVM was trained on the final training set, firstly using the default parameters. This classified 92% of cases in the training set correctly. Some of the incorrectly classified 4% of cases may have been incorrectly classified because the principal components chosen in the training set only account for 90% of the variance in *is_promoted* in the training set.

It was also possible that a SVM classifier with different parameters may correctly classify more cases in the training set. Therefore, several different SVM classifiers were then built, to see if a more accurate model could be found. The accuracy scores of these models are displayed in Figure 5.

Marginally, the best performing SVC classifier on the training set was the SVC with a 'poly' kernel and so it was decided to use this classifier for the prediction of whether an employee was promoted in the test set.

Again, due to the much higher chance of an employee not being promoted by the corporation compared to the chance of an employee being promoted, it would be wise here to use Youden's index and a precision recall curve to find the best threshold value for this model. However, due to time constraints, it was not possible to do this for the SVC model.

Thus, the default classification threshold, 0.5, was used when testing the model.

Non-default parameter(s)	Accuracy score
None (all default parameters)	0.924
kernel='linear'	0.915
kernel='poly'	0.926
kernel='sigmoid'	0.859
kernel='poly', degree = 8	0.927
kernel='poly', degree = 2	0.919

Fig. 5. Accuracy scores of different random SVM models on the training dataset

The model was then tested on the test set. Several evaluation metrics were calculated, comparing the predictions of this random forest classifier to the ground truth on the test set. The model had an accuracy of 0.921 here, meaning that it correctly classified 92.1% of cases - a much higher rate than the random forest model. A confusion matrix was created to show the classifications predicted by the model (see Fig. 6). From this confusion matrix, it is possible to see that the model was heavily biased towards the more coomon class 0 (no promotion), which also explains the comparatively high classification accuracy. The model had precision score of 0.727, meaning that most employees predicted as receiving a promotion did receive a promotion. The model had a recall score of 0.108, meaning that it did not detect most employees who were promoted. The F1 score, the weighted average of precision and accuracy of the model, was 0.18.

		Predicted	
		Not promoted	Promoted
Ground truth	Not promoted	12499	47
	Promoted	1031	125

Fig. 6. Confusion matrix showing the class predicted by the SVM classifier, and the ground truth.

A ROC curve of FP rate against TP rate for the test data was plotted (see Appendix E), and the AUC was found to be 0.72. The model is pretty poor, as it largely predicts any case as belonging to class 0. This bias could be addressed by adjusting the classification threshold, as discussed earlier, if more time were available.

E. Reflection

So, two rather different classification models were trained on a training subset of the overall promotion dataset and tested on a testing subset. The first model, a random forest classifier, had the lower accuracy out of the two models, but had a higher AUC, as well as detecting a much higher proportion of class 1 (promoted) cases.

It is somewhat difficult to compare the models, as the random forest classifier's classification threshold was optimised to give the best possible precision and recall scores. The SVM model was not optimized in this way, due to time constraints, and so its predictions were heavily biased towards class 0. If this task were not completed under such extreme time constraints, it would have been possible to build both models to be optimized for precision and recall, which would allow for a more accurate comparison of the two classifiers.

However, the models which were built here could both be said to be useful, depending on the desired outcome of this automated first round promotion process. The random tree classifier model is more likely to predict an employee to be promoted who should be promoted, however it is also more likely to predict an employee to be promoted who should not be promoted, based off previous data. On the other hand, the SVM model is less likely to predict an employee to be promoted who should be promoted, however it is also less likely to predict an employee to be promoted who should not be promoted.

The random tree model is, arguably, a better model for the purpose. Though it correctly classified 79.6% of cases in the test set, compared to the 92.1% classified by the SVC model, this probably does not matter too much. The idea of the model is to automate the first round of the promotion process. The second round is likely to be more in-depth, and involve human opinion, and so if the model recommends somebody is promoted who should not be promoted, then this will likely be rectified in the second round of the promotion process.

Still, both models clearly have some room for improvement. However, some of the inaccuracies in both models could also be explained by the nature of the task. First, it is possible that some employees in the dataset should have been promoted by the corporation in the past, but have been overlooked for some reason, such as a grudge held by one of their superiors. This would distort the dataset, as some employees with data points closer to a promoted employee may have not been promoted, confusing the classification models. It is also possible that some employees who had not been promoted in the dataset were due a promotion soon, which would distort the data further.

Overall, neither model built for the purpose of automating the corporation's first round decision process were particularly good. The random tree classifier model had a lower accuracy rate but did at least detect most employees who should be promoted. The SVM model had a good accuracy rate, but upon closer inspection it was visible that the model was highly biased towards the most common class, which meant it did mot detect many employees who should be promoted. With that said, the best model for the purpose of the task is probably the random tree model, as it flags more employees who should be promoted, and any errors made by the model could be rectified in the second stage of the promotion process.

III. TASK 2: IMAGE CLASSIFICATION

A. Data and Preliminary Analysis

The dataset provided for the image classification task contained 9649 images in the training set, and 3925 in the validation set. Both the training and validation set of images were labelled, belonging to one of ten classes.

The training set and the testing set of images were loaded into two objects in Jupyter notebooks. Initially, the size of the images was set at 200x200 pixels, and batch size was set at 32 images - though different sizes for both values were used later in the experiment.

To check that images had been loaded into the datasets correctly, a subset of images from both the training set (see

Fig. 7), and the validation set (see Fig. 8) was displayed in the notebook, along with the class labels for each image. The images and their labels appeared to be correct, and so the next stage of the analysis was to start building the CNN models.

Fig 7. Nine of the images in the training set and their class labels.

Fig 8. Nine of the images in the validation set and their class labels.

B. *Methods*

Three types of CNN architecture were applied to the image dataset. The first was a simple architecture adapted from a tutorial on the internet [3]. This architecture was chosen as it was a simple architecture which did not take much time to implement, and the tutorial it was sourced from offered a good template for building and testing different CNN models. This first CNN architecture consisted of three convolutional layers, two pooling layers, and two fully connected layers. The input size of images into this model was 200x200 pixels. A summary of the model is displayed in Appendix F.

The second CNN architecture used was LeNet, another simple architecture which takes 32x32 images as input. Due to the simple nature of the architecture, and the lower size of input images, this model trained extremely fast, and so was an ideal architecture to try on this image set. A summary of the model is displayed in Appendix G.

The third architecture used was AlexNet, a more complex neural network that used dropout layers to reduce overfitting and took images of size 227x227 pixels as input. The AlexNet architecture also has a larger number of layers, with more nodes in the fully connected layers. Due to the added depth and complexity of the architecture, it takes much longer to train it, but it was hoped that this architecture would be the best performing architecture in the experiment here. A summary of the model is displayed in Appendix H.

All models were compared on their validation accuracy and training accuracy after a specified number of epochs.

C. *Experiments*

The first model built, an adaptation of a model found in a tutorial on the internet [3], achieved an accuracy score of 0.192 on the validation set, and 0.700 on the training set after 10 epochs. The model was clearly overfitted to the training data at this point, with the training accuracy becoming much higher than the validation accuracy almost straight away. The validation and training accuracy of the model over the ten epochs is visualized in Figure 9.

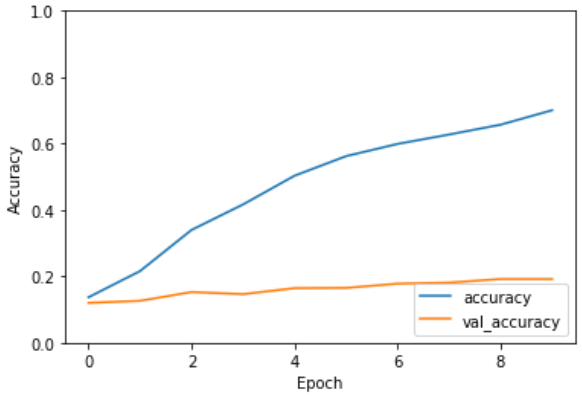


Fig 9. Plot of validation and training accuracy of the first model over ten epochs.

The second model built, using the LeNet architecture, achieved an accuracy score of 0.457 on the validation set, and 0.825 on the training set after ten epochs. Again, this model does show signs of overfitting, with the model being much more accurate on the training data compared to unseen data after around 2 epochs. Still, the fact that such a simple architecture, which only took 32x32 pixel images as its input and a short time to train, could correctly classify around half of the images in an unseen dataset is promising. The validation and training accuracy of the model over the ten epochs is visualized in Figure 10.

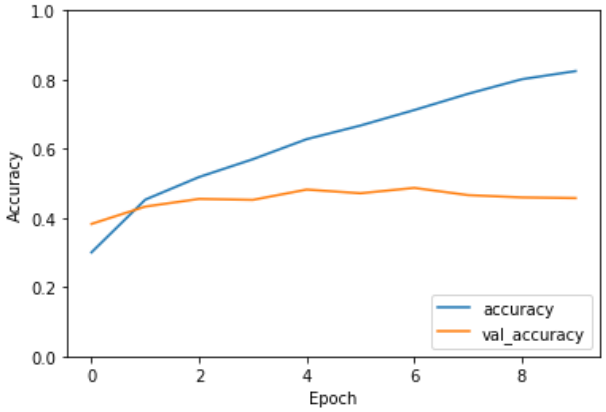


Fig 10. Plot of validation and training accuracy of the LeNet model over ten epochs.

The third model built, using the AlexNet architecture, achieved an accuracy score of 0.495 on the validation set, and 0.598 on the training set after ten epochs. It appears that the model had still not converged after these ten epochs so, if there was more time to train the model more extensively, it is possible that the accuracy of the model may have improved. Still though, after 10 epochs the AlexNet model could classify almost half of the images in the validation set correctly, and slightly more than the LeNet model attempted previously. Also, the issue of overfitting did not appear to be as problematic here, as the model performed similarly on both the training and validation sets over the ten epochs (see Fig. 11).

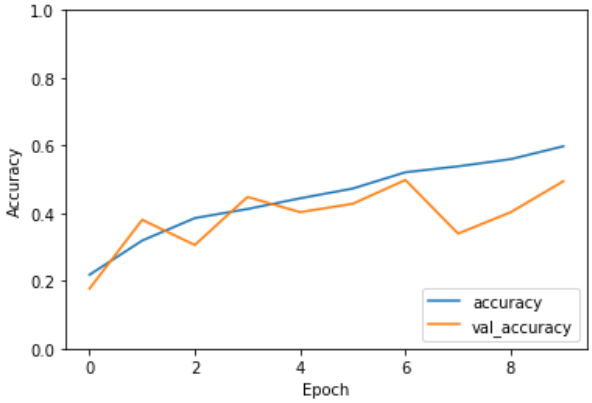


Fig 11. Plot of validation and training accuracy of the AlexNet model over ten epochs.

After these initial three models had been built, attempts were made to improve upon their validation accuracy scores. First, several parameters of the LeNet model were changed, one at a time. The results of this are displayed in Figure 12. None of the changes made to the standard model appeared to massively improve validation accuracy. Using a 2x2 kernel rather than the standard 3x3 kernel appears to have improved the classification accuracy slightly, but it is uncertain whether this improvement is significant.

Finally, the AlexNet model was trained over 25 epochs, to see whether validation accuracy would increase. After 25 epochs, the model achieved validation accuracy of 0.542, and training accuracy of 0.829. This is somewhat of an improvement to what it was after 10 epochs, though it does appear the model had started to become overfitted to the training data (see Fig 13). In previous epochs, the model had achieved accuracies as high as 0.693.

Change from standard LeNet architecture	Training Accuracy	Validation Accuracy
Batch size = 128	0.744	0.439
Batch size = 8	0.859	0.437
Image size = 128x128	0.988	0.454
Kernel size = 2x2	0.7464	0.500
Adding dropout layer	0.9791	0.4326

Fig 12. Training and validation accuracy of several modified LeNet models.

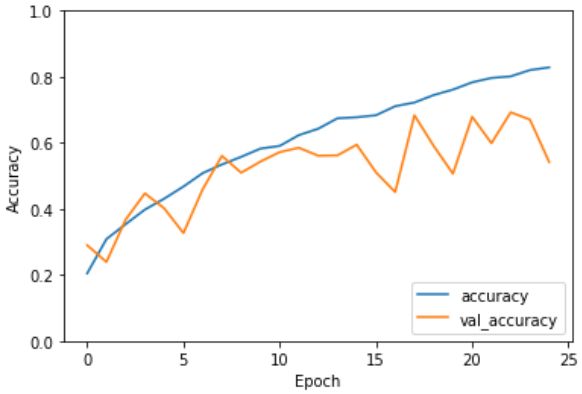


Fig 13. Plot of validation and training accuracy of the AlexNet model over 25 epochs.

D. Reflection

Overall, the two most accurate models on the validation set were the AlexNet and LeNet models. Both of these models achieved accuracies of around 0.5, and the AlexNet actually achieved close to 70% accuracy on the validation set, before it started to become overfitted to the training data.

The models were not trained with a hugely extensive set of images, and there are currently much more complex CNN architectures that would likely make more accurate models. Still, achieving around 50% accuracy on unseen data, in models that can be trained in a couple of minutes, is a reasonably good result.

When selecting the best model for image classification, there is a trade-off which must be assessed between fast training times and higher accuracy. In the two best performing models in this experiment, training the LeNet architecture took less than a tenth of the time it took to train the AlexNet model, but achieved only slightly worse accuracy. Depending on the use case of a model, it may sometimes be wise to build a slightly less accurate model if it takes much less time to train it.

REFERENCES

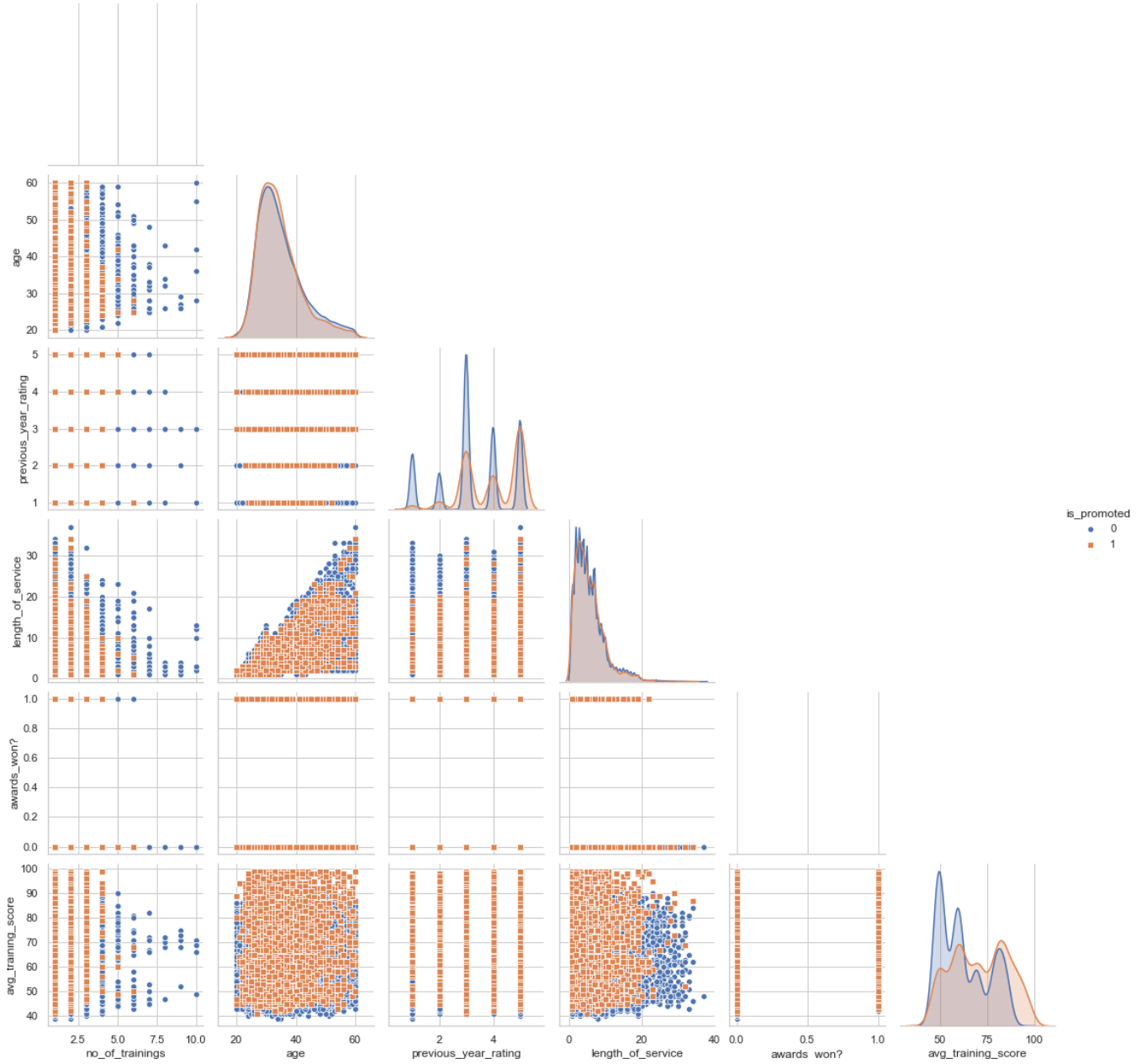
[1] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation*, *1*(4), pp.541-551.

[2] Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, *25*, pp.1097-1105.

[3] TensorFlow. 2021. Convolutional Neural Network (CNN). [online] Available at: <https://www.tensorflow.org/tutorials/images/cnn> [Accessed 13 May 2021].

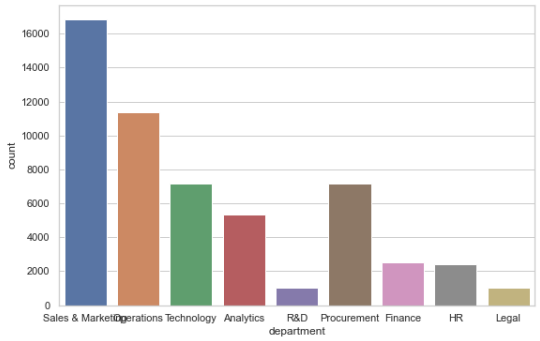
APPENDIX A

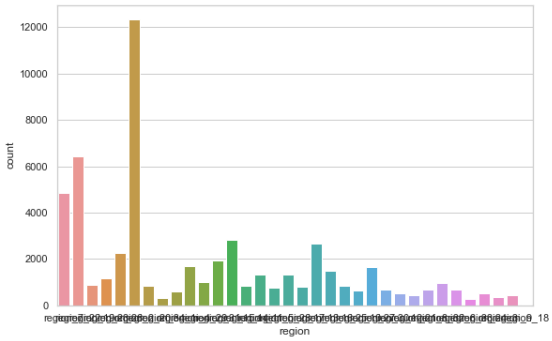
PAIR PLOT OF RELATIONSHIPS BETWEEN ATTRIBUTES AND THE TARGET VARIABLE

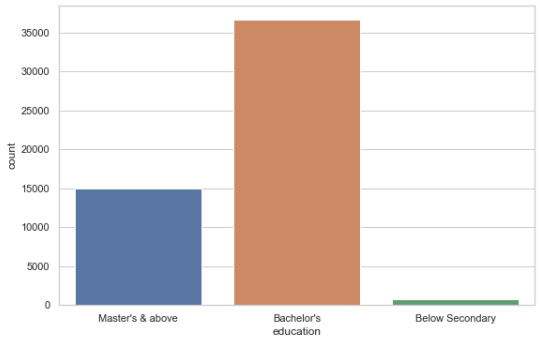


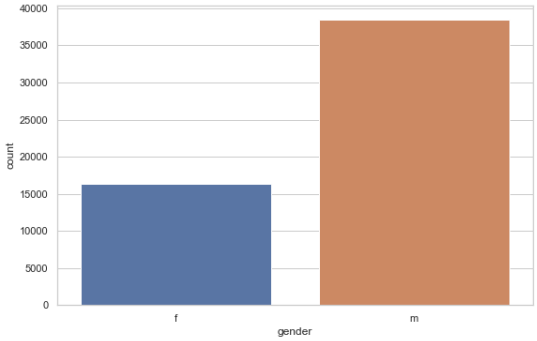
APPENDIX B

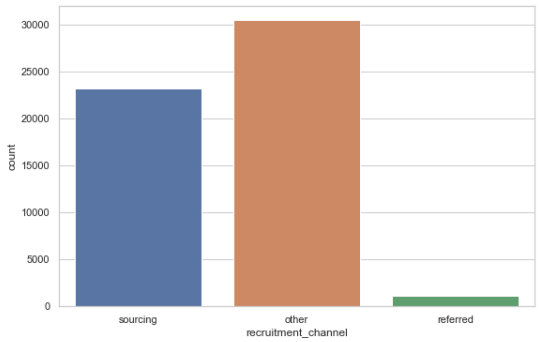
BAR CHART OF CATEGORICAL DATA

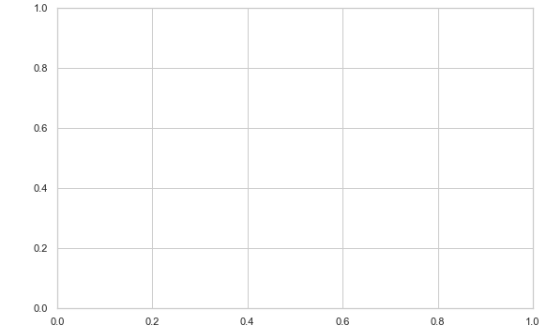






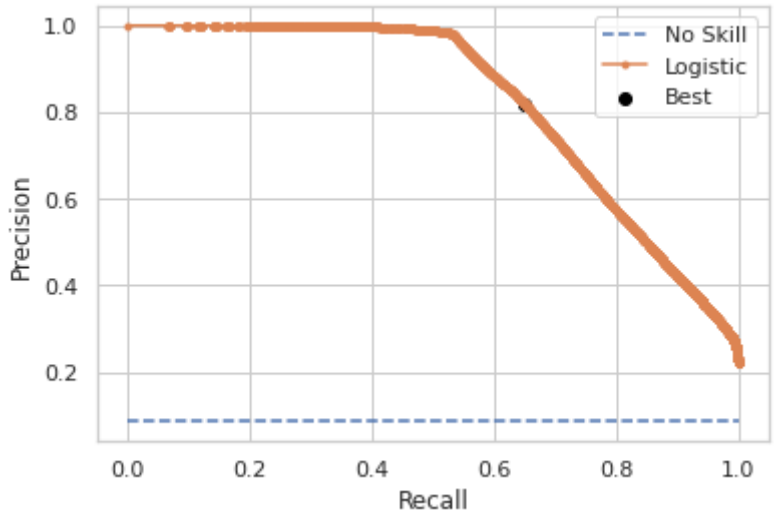






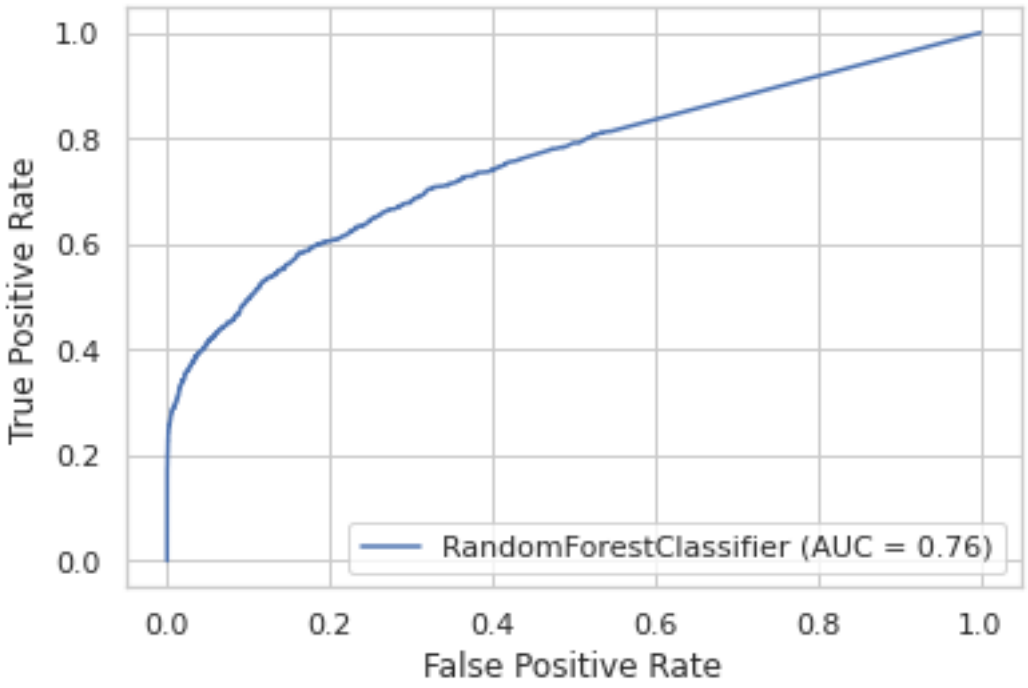
APPENDIX C

PRECISION RECALL CURVE FOR RANDOM FOREST CLASSIFIER ON TRAINING SET



APPENDIX D

ROC CURVE FOR RANDOM FOREST CLASSIFIER ON TEST SET



APPENDIX E

ROC CURVE FOR SVM ON TEST SET

APPENDIX F

SUMMARY OF FIRST CNN MODEL

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 198, 198, 6)	168
max_pooling2d_6 (MaxPooling2	(None, 99, 99, 6)	0
conv2d_10 (Conv2D)	(None, 97, 97, 64)	3520
max_pooling2d_7 (MaxPooling2	(None, 48, 48, 64)	0
conv2d_11 (Conv2D)	(None, 46, 46, 64)	36928
flatten_1 (Flatten)	(None, 135424)	0
dense_2 (Dense)	(None, 64)	8667200
dense_3 (Dense)	(None, 10)	650

Total params: 8,708,466
Trainable params: 8,708,466
Non-trainable params: 0

APPENDIX G

SUMMARY OF LENET MODEL

Layer (type)	Output Shape	Param #
conv2d_30 (Conv2D)	(None, 30, 30, 6)	168
average_pooling2d_8 (Average	(None, 15, 15, 6)	0
conv2d_31 (Conv2D)	(None, 13, 13, 16)	880
average_pooling2d_9 (Average	(None, 6, 6, 16)	0
flatten_8 (Flatten)	(None, 576)	0
dense_22 (Dense)	(None, 120)	69240
dense_23 (Dense)	(None, 84)	10164
dense_24 (Dense)	(None, 10)	850

Total params: 81,302
Trainable params: 81,302
Non-trainable params: 0

APPENDIX H

SUMMARY OF ALEXNET MODEL

Layer (type)	Output Shape	Param #
conv2d_20 (Conv2D)	(None, 55, 55, 96)	34944
batch_normalization (BatchNo	(None, 55, 55, 96)	384
max_pooling2d_8 (MaxPooling2	(None, 27, 27, 96)	0
conv2d_21 (Conv2D)	(None, 27, 27, 256)	614656
batch_normalization_1 (Batch	(None, 27, 27, 256)	1024
max_pooling2d_9 (MaxPooling2	(None, 13, 13, 256)	0
conv2d_22 (Conv2D)	(None, 13, 13, 384)	885120
batch_normalization_2 (Batch	(None, 13, 13, 384)	1536
conv2d_23 (Conv2D)	(None, 13, 13, 384)	1327488
batch_normalization_3 (Batch	(None, 13, 13, 384)	1536
conv2d_24 (Conv2D)	(None, 13, 13, 256)	884992
batch_normalization_4 (Batch	(None, 13, 13, 256)	1024
max_pooling2d_10 (MaxPooling	(None, 6, 6, 256)	0
flatten_6 (Flatten)	(None, 9216)	0
dense_16 (Dense)	(None, 4096)	37752832
dropout (Dropout)	(None, 4096)	0
dense_17 (Dense)	(None, 4096)	16781312
dropout_1 (Dropout)	(None, 4096)	0
dense_18 (Dense)	(None, 10)	40970

Total params: 58,327,818
Trainable params: 58,325,066
Non-trainable params: 2,752