Table of Contents

Problem Definition

Pat McGee is the CEO of a large multi-million dollar corporation with diverse business verticals, but he is facing a recurring problem in identifying the ideal employee candidates for promotion within his organization. The current promotion process only finalizes promotions after an extensive evaluation round, often resulting in significant delays in role transitions. Recognizing the need for a more efficient approach, Pat seeks a solution to identify eligible candidates at specific checkpoints, enabling the company to expedite the entire promotion cycle. To address this challenge, Pat has shared a comprehensive dataset encompassing various employee-related attributes regarding their performance and background and has tasked us to find the relationships within the dataset's features to develop a predictive model that should accurately identify the promotional worthy candidates.

Exploratory Data Analysis

This section highlights the key features that include Average_Training_Score, Awards_Won, Department and Previous_Year_Rating, and explores its relationship with the target variable Is_Promoted by providing visualizations and insightful findings. We will focus on summarizing the data presented in the original dataset, further examine its correlation with Is_Promoted, discuss how the data was prepared and treated for the modelling, and finally, describe the machine learning algorithms used to select the key features.

Summary of the Data

We will first examine the provided dataset and attributes. Figure 1 below showcases the first 5 rows of the provided dataset, comprising a total of 54808 rows of data and 13 columns of features. Shifting our attention to the key features, Average_ Training_Score represents the average score in current training evaluations, Awards_Won indicates whether the employee has won awards during their previous year or not, Department is the department of the employee, Previous_Year_Rating is the employee rating for the previous year, and finally, Is_Promoted serves as the target variable indicating whether the employee is recommended for promotion, with a binary value of 0 or 1 representing no or yes respectively.

Data Information:

	employee_id	department	region	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?	avg_training_score	is_promoted
0	65438	Sales & Marketing	region_7	Master's & above	f	sourcing	1	35	5.0	8	0	49	0
1	65141	Operations	region_22	Bachelor's	m	other	1	30	5.0	4	0	60	0
2	7513	Sales & Marketing	region_19	Bachelor's	m	sourcing	1	34	3.0	7	0	50	0
3	2542	Sales & Marketing	region_23	Bachelor's	m	other	2	39	1.0	10	0	50	0
4	48945	Technology	region_26	Bachelor's	m	other	1	45	3.0	2	0	73	0

Figure 1 — First 5 rows of the original dataset

A more intuitive representation of the features is presented in Figure 2 below, depicting a barplot distribution for each key feature against the target variable. Notably, in the plots for both Average_Training_Score and Previous_Year_Rating features, we can observe that there are slightly fewer occurrences when the target value is 0, suggesting a minor class imbalance. The Awards_Won plot also shows far fewer instances when the target value is 0, further underscoring the presence of class imbalance.

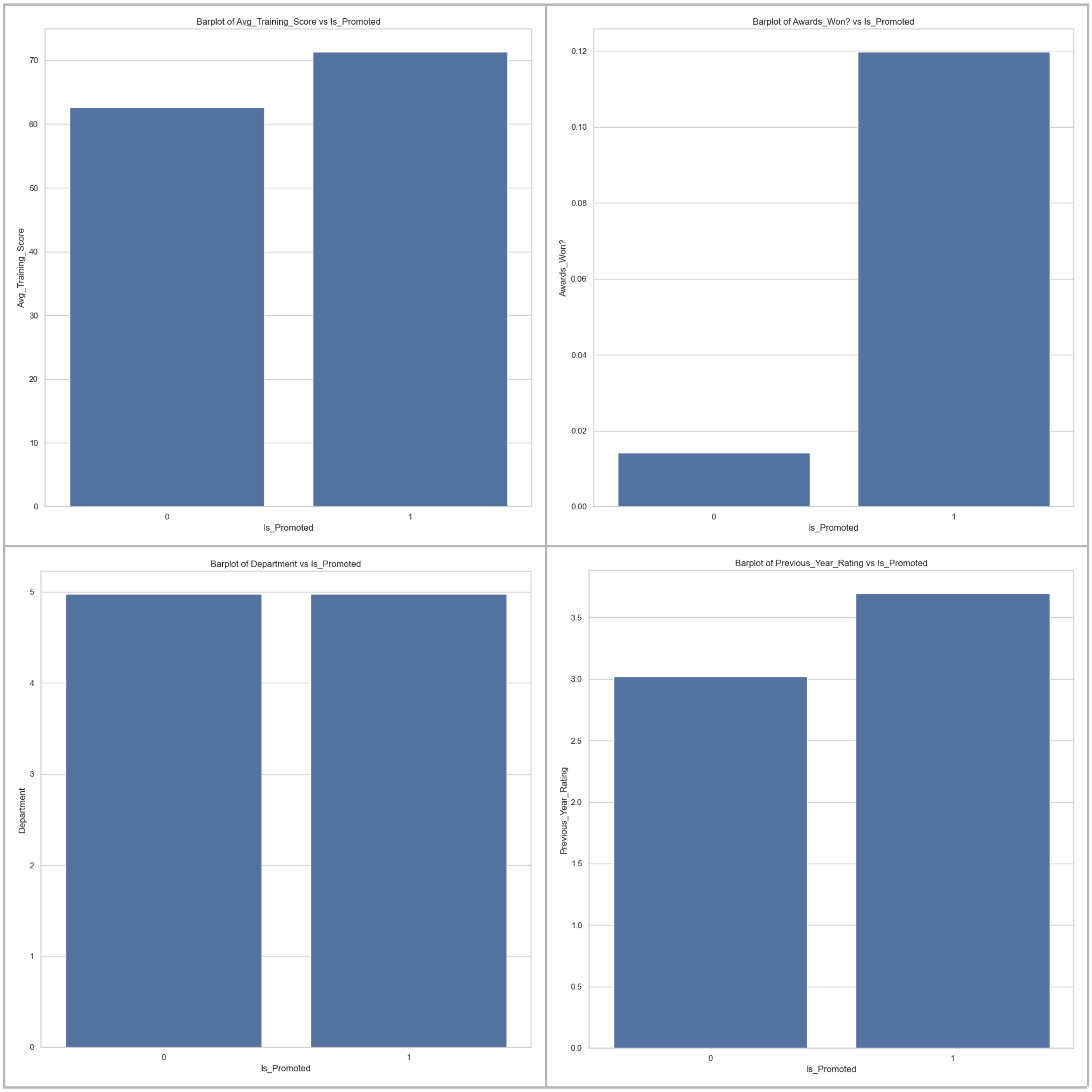

Figure 2 — Barplot of each key feature against Is_Promoted

Another way to visualize the data is through plotting a boxplot which is illustrated in Figure 3 below. Boxplots can also often compare the distribution of each key feature across different values of the target variable which can help identify any significant differences in those distributions. In the Average_Training_Score plot, we can observe that all quantiles are higher when the target variable has a value of 1 is higher than all quantiles when the target variable has a value of 0. The interquartile range for when the target value is 0 ranges approximately between 50 to 75 of an Average_Training_Score whereas when the target value is 1, the values range between 60 to 85. The boxplot also seems to be skewed for when the target value is 0 as we can see the median line in the middle of the box shifted lower than the central line. The Awards_Won feature holds binary values which explains the oddly depicted boxplot as shown below. The Department plot shows a relatively balanced distribution as we observe both boxplots tend to share the same interquartile range and median which was also depicted in the barplot. Nonetheless, the boxplot shows that the interquartile values range between 4 to 7 departments with a median of 5 departments. Lastly, the Previous_Year_Rating boxplot shows the boxplot with the target value of 0 having an interquartile range of 2 to 4 scale ratings which is lower than the boxplot with a target value of 1 having an interquartile range of 3 to 5 scale ratings, with no significant outliers detected.

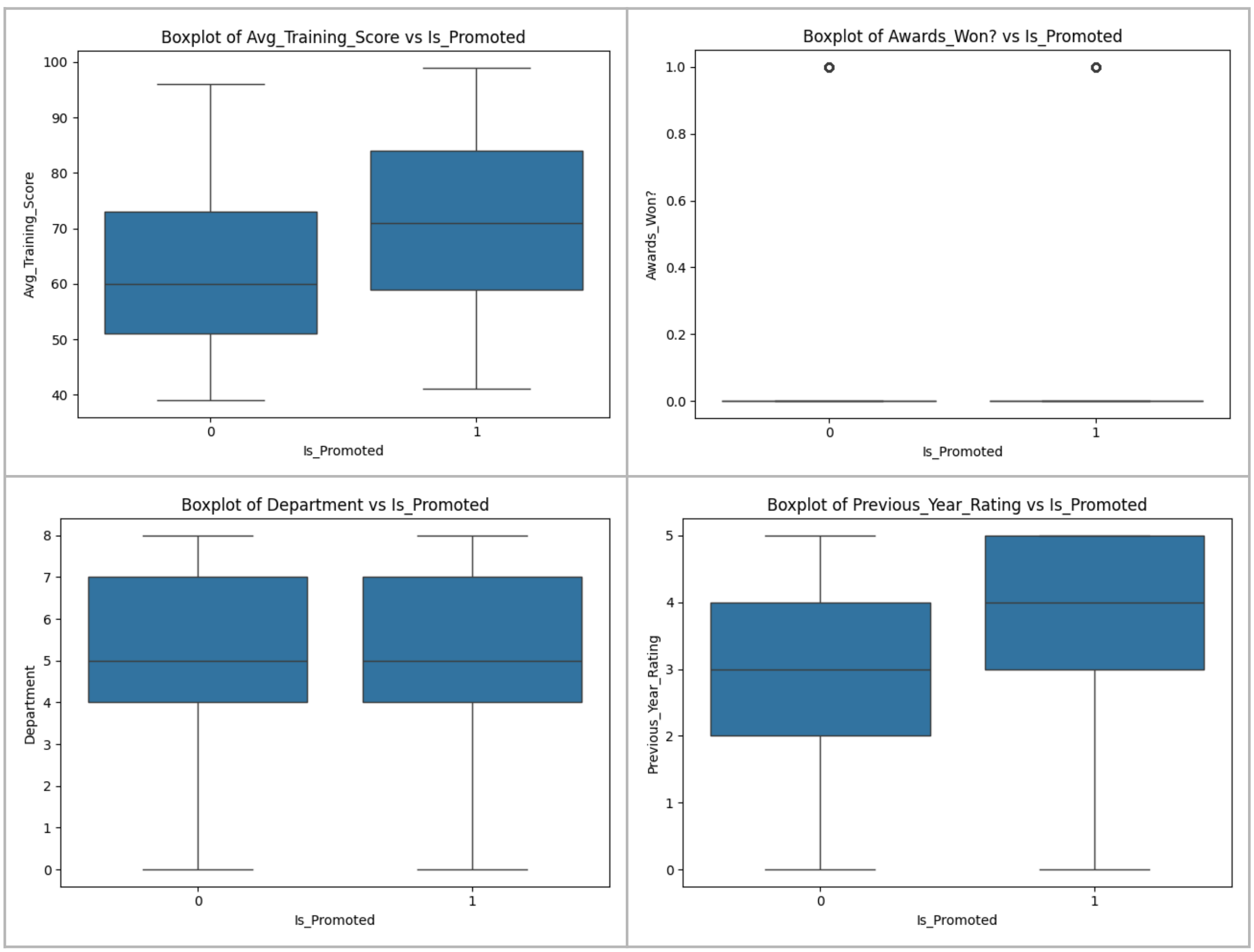

Figure 3 — Boxplot of each key feature against Is_Promoted

Correlation

A correlation heatmap is presented in Figure 4 below, illustrating the correlation coefficient between each key feature and the target variable, Is_Promoted. Unfortunately, we can see that all features exhibit a very low correlation with a coefficient value lower than 0.2. However, it's important to note that low correlation may simply imply the absence of linear relationships which may be irrelevant for a binary classification dataset. Tuning the hyperparameters of our models or scaling the data can often mitigate this issue effectively.

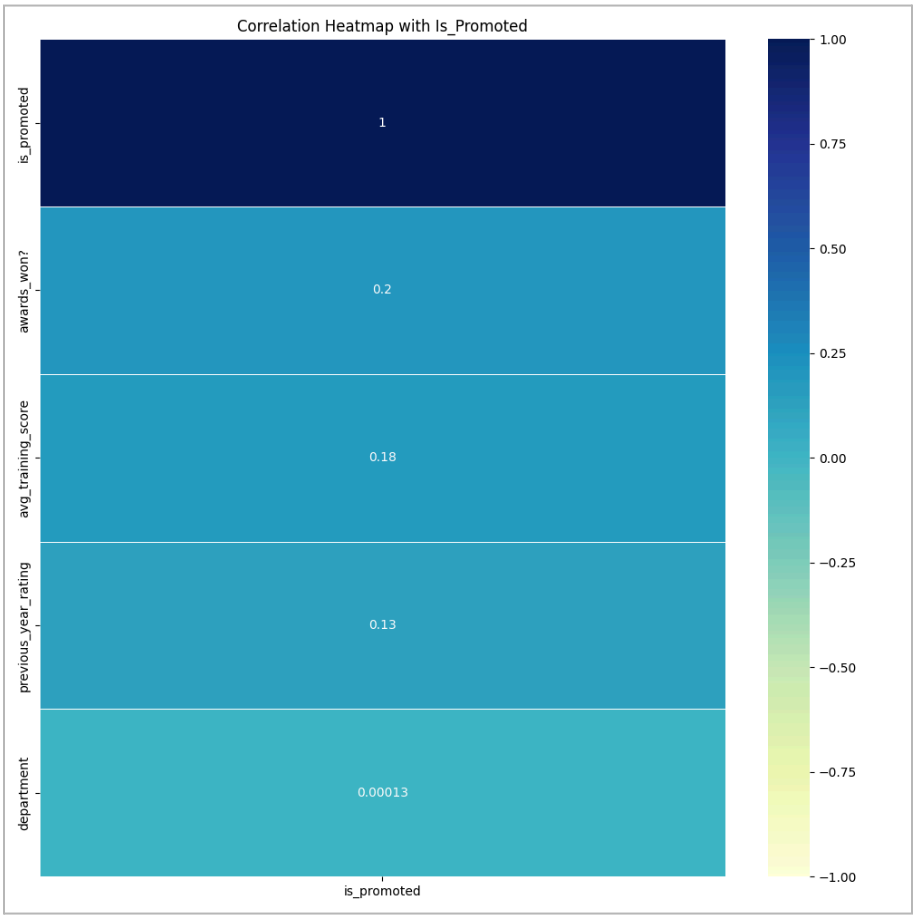

Figure 4 — Correlation heatmap of key features with Is_Promoted

Data Treatment

Prior to model development, data preparation is essential. Following comprehensive research and data analysis, we have identified several columns containing missing values and non-numeric features that need to be treated. Firstly, we can observe the column data types and non-null counts representing the absence of missing values revealed in Figure 5 below. Focusing on the key features, the Department column appears to be an Object data type, requiring conversion to a numerical format for model compatibility. Moreover, the Previous_Year_Rating column exhibits missing values with a non-null count of 50684 out of the total 54808 rows. Consequently, some form of imputation would be necessary to address these missing data points, enabling their utilization in the development of the model.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 54808 entries, 0 to 54807
Data columns (total 13 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   employee_id           54808 non-null  int64
 1   department            54808 non-null  object
 2   region                54808 non-null  object
 3   education             52399 non-null  object
 4   gender                54808 non-null  object
 5   recruitment_channel   54808 non-null  object
 6   no_of_trainings       54808 non-null  int64
 7   age                   54808 non-null  int64
 8   previous_year_rating  50684 non-null  float64
 9   length_of_service     54808 non-null  int64
 10  awards_won?           54808 non-null  int64
 11  avg_training_score    54808 non-null  int64
 12  is_promoted           54808 non-null  int64
dtypes: float64(1), int64(7), object(5)
```

Figure 5 — Dataset feature data types and null counts

Considering the limited variations in the Department column as illustrated in Figure 6 below with only 9 distinct department values, we can employ label encoding to assign numerical indices indexed by 0 (0 to 8) to each unique department. This transformation will facilitate the conversion of the column into a numerical feature, ensuring compatibility with the model.

```
department
Sales & Marketing    16840
Operations           11348
Technology            7138
Procurement           7138
Analytics             5352
Finance               2536
HR                    2418
Legal                 1039
R&D                    999
Name: count, dtype: int64
```

Figure 6 — Dataset feature data types and null counts

After conducting research, we have identified the cause of missing values in the Previous_Year_Rating column. These missing values correspond to the workers who have only served the firm for 1 year, as depicted in Figure 7 below. Since these workers were not part of the firm in the previous year, there are no ratings available for them. To remedy this issue, we will fill in the missing values with 0, indicating that these workers did not have a Previous_Year_Rating due to their absence from the firm during that period. This approach ensures that the dataset is appropriately handled considering the unique circumstances of these individuals.

```
Previous Year Rating Rows with Null Values:
      length_of_service  previous_year_rating
10                    1                   NaN
23                    1                   NaN
29                    1                   NaN
56                    1                   NaN
58                    1                   NaN
62                    1                   NaN
66                    1                   NaN
67                    1                   NaN
84                    1                   NaN
89                    1                   NaN
90                    1                   NaN
96                    1                   NaN
111                   1                   NaN
123                   1                   NaN
125                   1                   NaN
127                   1                   NaN
135                   1                   NaN
141                   1                   NaN
160                   1                   NaN
178                   1                   NaN
214                   1                   NaN
220                   1                   NaN
232                   1                   NaN
242                   1                   NaN
245                   1                   NaN
255                   1                   NaN
272                   1                   NaN
```

Figure 7 — Previous_Year_Rating column rows with missing values

Upon handling the missing values, converting all columns to numerical features and removing irrelevant columns representing unique identifiers such as Employee_Id and Region, the final prepared dataset is depicted in Figure 8 below.

Preprocessed Data:

	department	education	gender	recruitment_channel	no_of_trainings	age	previous_year_rating	length_of_service	awards_won?	avg_training_score	is_promoted
0	7	2	0	2	1	35	5.0	8	0	49	0
1	4	0	1	0	1	30	5.0	4	0	60	0
2	7	0	1	2	1	34	3.0	7	0	50	0
3	7	0	1	0	2	39	1.0	10	0	50	0
4	8	0	1	0	1	45	3.0	2	0	73	0

Figure 8 — First 5 rows of the prepared dataset

Feature Selection Comparison

During our analysis, we employed 3 machine-learning algorithms: Recursive Feature Elimination, Forward Feature Selection, and Feature Important using a Random Forest Classifier. These techniques were utilized to identify significant features for our model development. Table 1 below displays the results of the automated feature selection methods.

Feature Selection	**Recursive Feature Elimination**	**Forward Feature Selection**	**Feature Importance (Random Forest)**
Features	department, no_of_trainings, previous_year_rating, awards_won?, avg_training_score	awards_won?, avg_training_score, previous_year_rating	avg_training_score, age, length_of_service, department, previous_year_rating
# of Features	5	3	5

Table 1 — Comparison table of automated feature selection routines and selected features.

Notably, Average_Training_Score and Previous_Year_Rating emerged as common features selected by all 3 algorithms highlighted in green. Additionally, features Department and Awards_Won were consistently identified across the algorithms, either individually or in combination with other features. These findings provide valuable insights into the most influential features that hold a strong relationship with the target variable for our model development process.

Model Analysis Breakdown

This section aims to assess the performance of 3 predictive models: Logistic Regression, Stacked Classifier, and Artificial Neural Network, in predicting our target variable, Is_Promoted. We will conduct a comparative analysis of their metrics across multiple cross-fold validation iterations to determine which of the 3 models presents the most optimal model that demonstrates superior predictive capabilities and optimal feature selection.

Model Selection Comparison

After evaluating the performance of the 3 machine-learning algorithms to identify significant features, it's evident that both features Average_Training_Score and Previous_Year_Rating played a significant role in the models as we can see it being commonly selected in Table 2 below. Among the metrics, our custom parameter-tuned Artificial Neural Network emerged with the highest average accuracy, precision, recall and F1 score out of the 3 models. These scores reflect its excellent fit and high predictive power. However, it's worth noting that the Logistic Regression model exhibited the lowest standard deviation in accuracy, recall and F1 score, indicating a remarkable consistency in its performance metrics. Overall, all 3 models demonstrate robust statistics and share common feature selections, emphasizing their strong predictive capabilities.

Model	Logistic Regression	Stacked Classifier	Artificial Neural Network
Features	avg_training_score, awards_won?, no_of_trainings, previous_year_rating	avg_training_score, department, length_of_service, previous_year_rating	avg_training_score, awards_won?, department, previous_year_rating
# of Features	4	4	4
Model Parameters and Tuning	N/A	Stacked Models = "**LogisticRegression**", "**DecisionTreeClassifier**", "**AdaBoostClassifier**", "**RandomForestClassifier**"	Batch Size: **100** Epochs: **50** Optimizer: **RMSprop** Learning Rate: **0.01** Kernel Initializer: **he_normal** Number of Neurons: **150** Additional Layers: **3** Activation Function: **softsign**
Average Accuracy	0.9185885747312881	0.9389687899938342	0.9412129892648393
Std of Accuracy	0.002629706639223549	0.0035629513866176155	0.0031201707117856415
Average Precision	0.9025683379925674	0.9350372095719773	0.9408872400464634
Std of Precision	0.004135920977694782	0.003655829080103445	0.003740932074448582
Average Recall	0.9185885747312881	0.9389687899938342	0.9412129892648393
Std of Recall	0.002629706639223549	0.0035629513866176155	0.0031201707117856415
Average F1 Score	0.8872539753993897	0.9263381581380816	0.9280688293793432
Std of F1 Score	0.003956165723142604	0.004392024778758191	0.004175838446319676

Table 2 — Comparison table for the models and their respective features, metrics and standard deviations.

Stacked Model Development and Tuning

The stacked model employed several base classifiers including Logistic Regression, Decision Tree Classifier, AdaBoost Classifier, and Random Forest Classifier, each operating independently on the features. This ensemble approach was designed to leverage the diverse strengths of individual classifiers. The model's performance was evaluated using k-fold cross-validation by splitting the training dataset into train and validation sets. For each iteration, we fit and made the predictions for each base model, which provided robust estimates of accuracy, precision, recall, and F1 score across multiple iterations. The stacked model was fitted by a simple Logistic Regression model and evaluated with the predictions made on the validation set of data.

The base models all performed very well with remarkable statistical metrics having to score above 86% for all performance categories as shown in Figure 9 below. The stacked model in comparison took the weighted average across the metrics and also excelled in performance as shown in Table 2 above. The average standard deviations for the metrics average approximately from 0.003 to 0.004 showing a very strong consistent model indicating a good fit.

```
Model: LogisticRegression
Accuracy: 0.9116788321167884
Precision: 0.8311582929298311
Recall: 0.9116788321167884
F1 Score: 0.8695585042488496
Classification Report:
              precision    recall  f1-score   support

           0       0.91      1.00      0.95      4996
           1       0.00      0.00      0.00       484

    accuracy                           0.91      5480
   macro avg       0.46      0.50      0.48      5480
weighted avg       0.83      0.91      0.87      5480
```

```
Model: DecisionTreeClassifier
Accuracy: 0.935948905109489
Precision: 0.9312600692162737
Recall: 0.935948905109489
F1 Score: 0.9228293565288047
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97      4996
           1       0.85      0.33      0.48       484

    accuracy                           0.94      5480
   macro avg       0.90      0.66      0.72      5480
weighted avg       0.93      0.94      0.92      5480
```

```
Model: AdaBoostClassifier
Accuracy: 0.9182481751824818
Precision: 0.9070129078706463
Recall: 0.9182481751824818
F1 Score: 0.8888238008091603
Classification Report:
              precision    recall  f1-score   support

           0       0.92      1.00      0.96      4996
           1       0.77      0.11      0.19       484

    accuracy                           0.92      5480
   macro avg       0.85      0.55      0.57      5480
weighted avg       0.91      0.92      0.89      5480
```

```
Model: RandomForestClassifier
Accuracy: 0.9361313868613139
Precision: 0.9314907548146787
Recall: 0.9361313868613139
F1 Score: 0.9231143490543309
Classification Report:
              precision    recall  f1-score   support

           0       0.94      0.99      0.97      4996
           1       0.85      0.33      0.48       484

    accuracy                           0.94      5480
   macro avg       0.90      0.66      0.72      5480
weighted avg       0.93      0.94      0.92      5480
```

Figure 9 — Stacked base models' metrics for one iteration of cross-fold validation

Artificial Neural Network Model Development and Tuning

To optimize the training process of the Artificial Neural Network (ANN) model, we conducted a grid search to experiment with different combinations of several parameters including the number of Epochs, Batch Size, Optimizer, Learning Rate, Kernel Initializer, number of Neurons, number of additional hidden layers, activation functions.

For batch sizes, we tested values of 10, 50, and 100, while exploring the number of epochs with values of 50, 100, and 200. Among the configurations tested, we observed that a batch size of 10 combined with 100 epochs consistently yielded the best results in terms of minimizing loss and maximizing accuracy. However, for the sake of easier development and time, we decided to choose 100 batches with 50 epochs as it yielded similar results for a shorter period of training.

Batch Size	Epochs	Loss	Accuracy	Precision	Recall	F1-Score
10	50	0.251140	0.925070	0.919282	0.925070	0.900515
10	100	0.241978	0.926286	0.922392	0.926286	0.902419
10	200	0.249797	0.925070	0.916027	0.925070	0.902097
50	50	0.243324	0.925070	0.916784	0.925070	0.901674
50	100	0.251743	0.922151	0.905967	0.922151	0.900925
50	200	0.275224	0.909257	0.884813	0.909257	0.892187
100	50	0.241250	0.924948	0.915804	0.924948	0.901868
100	100	0.237978	0.925557	0.918199	0.925557	0.902313
100	200	0.247560	0.925435	0.917707	0.925435	0.902224

Table 3 — ANN model grid searched Batch Size and Epoch results

To determine the most effective optimizer for training our Artificial Neural Network (ANN) model, we conducted a comprehensive grid search involving various optimizers, including Adam, SGD, RMSprop, Adagrad, Adadelta, Adamax, and Nadam. Among the optimizers tested, RMSprop emerged as the most suitable choice for our model, exhibiting favourable performance across all metrics. Specifically, RMSprop achieved a relatively low loss value of 0.233656, the highest accuracy of 0.928233, the highest precision of 0.923512, and the highest recall of 0.928233 among the tested optimizers.

Optimizer	Loss	Accuracy	Precision	Recall	F1-Score
Adam	0.231680	0.927138	0.915691	0.927138	0.904867

SGD	0.269369	0.918258	0.843198	0.918258	0.879129
RMSprop	0.233656	0.928233	0.923512	0.928233	0.904037
Adagrad	0.314114	0.918258	0.843198	0.918258	0.879129
Adadelta	0.603502	0.918258	0.843198	0.918258	0.879129
Adamax	0.237279	0.926165	0.914700	0.926165	0.902269
Nadam	0.232216	0.927259	0.915252	0.927259	0.905657

Table 4 — ANN model grid searched Optimizer results

To optimize the training process of our Artificial Neural Network (ANN) model, we conducted a systematic grid search to identify the most effective learning rate including values of 0.001, 0.005, 0.01, 0.05, 0.1, and 0.2. Among the learning rates tested, 0.01 emerged as the most optimal choice for our model, yielding the lowest loss value of 0.223878, the highest accuracy of 0.93322, the highest precision of 0.929737, the highest recall of 0.93322, and the highest F1 score of 0.912689.

Learning Rate	Loss	Accuracy	Precision	Recall	F1-Score
0.001	0.231484	0.930300	0.922844	0.930300	0.908162
0.005	0.232660	0.929936	0.921893	0.929936	0.907609
0.01	0.223878	0.933220	0.929737	0.933220	0.912689
0.05	0.244577	0.930179	0.919883	0.930179	0.909610
0.1	0.249279	0.931030	0.924416	0.931030	0.909407
0.2	0.380763	0.911933	0.888236	0.911933	0.896513

Table 5 — ANN model grid searched Learning Rate results

To optimize the initialization of kernel weights in our Artificial Neural Network (ANN) model, we conducted a systematic grid search to identify the most effective kernel initializer. Various initializers, including uniform, lecun_uniform, normal, zero, glorot_normal, glorot_uniform, he_normal, and he_uniform, were evaluated based on their impact on model performance metrics. Among the kernel initializers tested, he_normal emerged as the most optimal choice for our model, yielding the lowest loss value of 0.222840 and dominating every statistical metric in the table.

Initializer	Loss	Accuracy	Precision	Recall	F1-Score

uniform	0.250072	0.924218	0.908709	0.924218	0.909358
lecun_uniform	0.241512	0.926530	0.915296	0.926530	0.905959
normal	0.236439	0.926530	0.913554	0.926530	0.908082
zero	0.289201	0.915704	0.838513	0.915704	0.875410
glorot_normal	0.234739	0.929692	0.929006	0.929692	0.907395
glorot_uniform	0.240927	0.927503	0.914351	0.927503	0.912310
he_normal	0.222840	0.935531	0.935260	0.935531	0.918155
he_uniform	0.242815	0.929814	0.919438	0.929814	0.913367

Table 6 — ANN model grid searched Kernel Initializer results

To optimize the architecture of our Artificial Neural Network (ANN) model, we conducted a systematic grid search to identify the most effective number of neurons in the hidden layers. We tested various configurations ranging from 5 to 200 neurons, evaluating each configuration based on its impact on model performance metrics. Among the configurations tested, a hidden layer with 150 neurons emerged as the most optimal choice for our model. This configuration yielded the second-lowest loss value of 0.221430, the highest accuracy of 0.939059, the highest recall of 0.939059, and the highest F1 score of 0.926647.

Neurons	Loss	Accuracy	Precision	Recall	F1-Score
5	0.235092	0.928233	0.919227	0.928233	0.906452
25	0.223662	0.933341	0.923798	0.933341	0.919024
50	0.218547	0.937356	0.936137	0.937356	0.920918
100	0.225821	0.937234	0.932244	0.937234	0.922686
150	0.221430	0.939059	0.933333	0.939059	0.926647
200	0.239393	0.933341	0.922834	0.933341	0.921935

Table 7 — ANN model grid searched the number of Neurons results

To optimize the architecture of our Artificial Neural Network (ANN) model further, we conducted a systematic grid search to identify the most effective number of additional hidden layers. We tested various configurations ranging from 0 to 5 additional hidden layers, evaluating each configuration based on its impact on model performance metrics. Among the configurations tested, adding 3 additional hidden layers emerged as the most optimal choice for our model. This configuration yielded the lowest loss value of 0.229829, the highest accuracy of 0.937234, the highest precision of 0.938534, the highest recall of 0.937234, and the highest F1 score of 0.922259.

Additional Hidden Layers	Loss	Accuracy	Precision	Recall	F1-Score
0	0.246010	0.928597	0.919644	0.928597	0.913055
1	0.262790	0.934436	0.931005	0.934436	0.919795
2	0.247990	0.932490	0.931464	0.932490	0.915199
3	0.229829	0.937234	0.938534	0.937234	0.922259
4	0.269605	0.935288	0.930924	0.935288	0.921869
5	0.235651	0.936747	0.936872	0.936747	0.921936

Table 8 — ANN model grid searched the number of additional hidden layers results

To determine the most effective activation function for our Artificial Neural Network (ANN) model, we conducted a systematic grid search, evaluating various activation functions including softmax, softplus, softsign, relu, tanh, and sigmoid. Among the activation functions tested, softsign emerged as the most optimal choice for our model, demonstrating superior performance across all evaluated metrics. This activation function yielded the lowest loss value of 0.201647 and dominated the metrics with the highest accuracy of 0.942586, the highest precision of 0.942957, the highest recall of 0.942586, and the highest F1 score of 0.930048.

Activation Function	Loss	Accuracy	Precision	Recall	F1-Score
softmax	0.291827	0.914609	0.836509	0.914609	0.873818
softplus	0.223648	0.937356	0.937205	0.937356	0.921703
softsign	0.201647	0.942586	0.942957	0.942586	0.930048
relu	0.205120	0.941248	0.941574	0.941248	0.927938
tanh	0.216874	0.941370	0.940354	0.941370	0.928738
sigmoid	0.205826	0.941248	0.940220	0.941248	0.928547

Table 9 — ANN model grid searched the Activation Function results

Model Evaluation

After examining the performance between the 3 models, the Artificial Neural Network model dominated the metrics having the highest accuracy, precision, recall, F1 score and minimizing losses with very favourable standard deviations for the respective metrics. The features that were used for the models seem to exhibit the features that were presented from the machine-learning feature selection techniques as well.

After grid searching the best and most optimal parameters to tune the ANN model, we split the training dataset into train and validation sets across 10 splits of k-fold cross-fold validations. We performed a standard scaling on the features for normalization to bring all features to the same level. Then, we compiled the model with binary-crossentropy for the loss function and evaluated the metrics based on its weighted average. To save the model, we added an early stopping with parameters of min_delta value set to 0.000001 and patience level to 200 epochs and used model checkpoint to save the binary model weights using Keras.

To visualize the model performance, we plotted the training and validation losses and accuracy as depicted in Figure 10 below. Although the validation losses curve spikes up at certain points in time, it can be mitigated by applying more epochs to smoothen the decline and enhance the performance. We can also observe that the validation losses are higher than the training sets with slightly higher accuracy. The accuracy plot shows that the accuracy peaks at approximately 5 epochs while suggesting that potentially 5 to 20 epochs are required for reaching peak performance during training with the current set of parameters which is low and favourable.

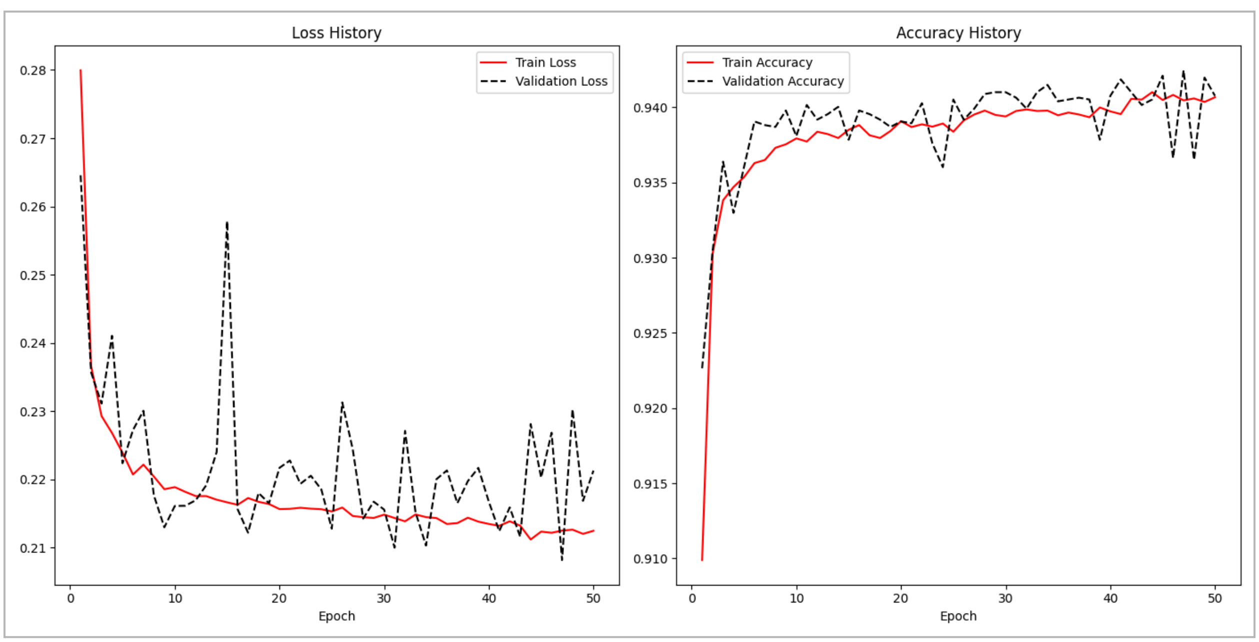

Figure 10 — ANN model training and validation loss and accuracy plots