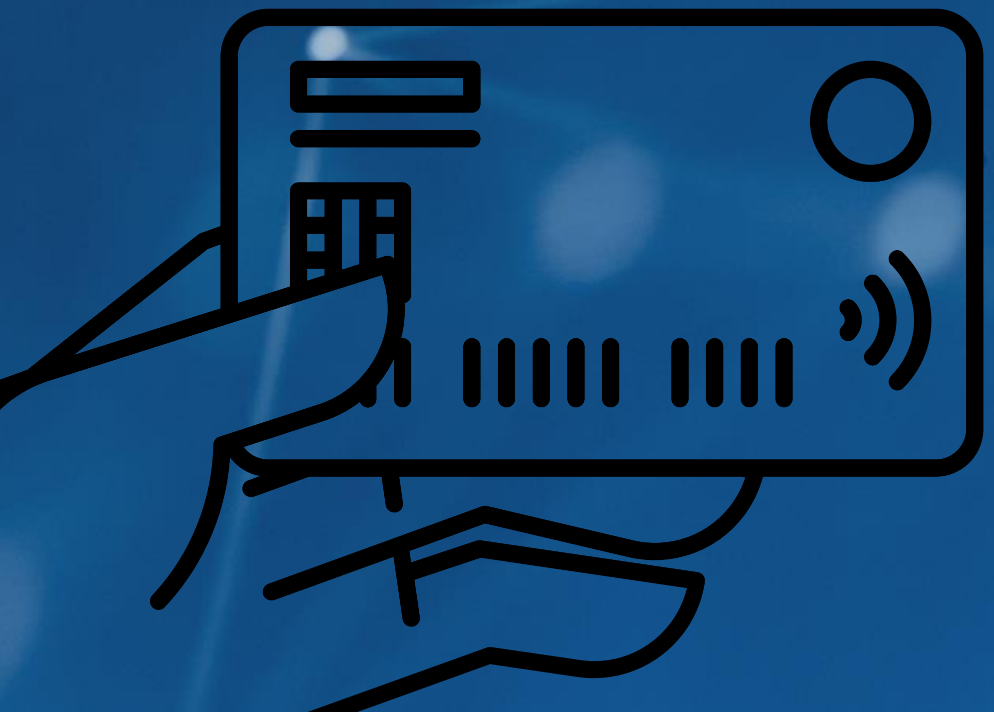
AI & Deep Learning

Credit Card Default Issue

Subham Sarangi



Business Understanding

Credit Card Default Issue

A single missed credit card payment does not constitute a default. A payment default happens when you fail to pay the Minimum Amount Due on your credit cardfor several months in a row. The card issuer usually sends the default notification after 6 consecutive missed payments. The bank, however, has the last say.

Problem Statement

A Taiwanese credit card provider aims to increase its capacity for forecasting consumer default and determine the key variables that influence this risk. This would enable the issuer to select the recipients of credit cards and the credit limits they will offer. In addition, it helps issuers or businesses to better understand their current and potential customers, guiding future strategies and plans for offering customized lending products to their customers. This can be predicted based on the customer's payment behaviour and default status over the last six months.

Data Understanding

We are preparing to understand the problems faced by the banks, when a credit card is being issued to customers to avoid the problem of default to understand, whether the credit card customer will make payment default in the next month or not based on number of features:

```
dataSet.head(10)
```

	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2	PAY_
0	1	20000[illegible]	2	2	1	24	[illegible]	[illegible]	[illegible]	[illegible]	...	NaN	NaN	NaN	0.0	689.0	
1	2	[illegible]20000.0	2	2	2	26	-1	2	[illegible]	0	...	3272.0	3455.0	3261.0	0.0	1000.0	1
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0	1500.0	1
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0	2019.0	1
4	5	50000.0	1	2	1	57	-1	0	-1	[illegible]	...	20940.0	19146.0	19131.0	2000.0	36681.0	10
5	6	50000.0	1	1	2	37	0	0	0	0	[illegible]	19394.0	19619.0	20024.0	2500.0	1815.0	
6	7	500000.0	1	1	2	29	0	0	0	0	...	542653.0	483003.0	473944.0	55000.0	40000.0	38
7	8	100000.0	2	2	2	23	0	-1	-1	0	...	221.0	-159.0	567.0	380.0	601.0	
8	9	140000.0	2	3	1	28	0	0	2	0	...	[illegible]2211.0	11793.0	3719.0	3329.0	0.0	
9	10	20000.0	1	3	2	35	-2	-2	-2	-2	...	[illegible]0.0	13007.0	13912.0	0.0	0.0	

10 rows × 25 columns

PAY_AMT
Amount of previous payment in different months

Gender
1= Male
2= Female

BILL_AMT
Amount of bill statement in different months

1=graduate school, 2=university, 3=high school, 4=others, 5=unknown, 6=unknown

MARRIAGE
1=married, 2=single, 3=others

Pay (Repayment status in different months)
(-1=pay duly, 1=payment delay for one month, 2=payment delay for two months, ... 8=payment delay for eight months, 9=payment delay for nine months and above)

Exploratory Data Understanding

Descriptive Statistics

```
dataSet.describe()
```

	ID	LIMIT_BAL	AGE	BILL_AMT1	BILL_AMT2	BILL_AMT3	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT
count	30000.000000	30000.000000	30000.000000	27992.000000	27494.000000	2.713000e+04	26805.000000	26494.000000	25980.000000	24751.00000
mean	15000.500000	167484.322667	35.485500	54897.825343	53661.608169	5.198653e+04	48419.640701	45645.883181	44886.559353	6864.66870
std	8660.398374	129747.661567	9.217904	74896.515914	72711.059216	7.113061e+04	66199.329394	62784.900318	61850.740349	18007.76437
min	1.000000	10000.000000	21.000000	-165580.000000	-69777.000000	-1.572640e+05	-170000.000000	-81334.000000	-339603.000000	1.00000
25%	7500.750000	50000.000000	28.000000	6059.750000	6239.000000	6.570500e+03	6569.000000	5962.500000	5418.500000	1610.00000
50%	15000.500000	140000.000000	34.000000	26732.500000	26848.000000	2.592550e+04	23437.000000	21319.000000	20818.500000	3000.00000
75%	22500.250000	240000.000000	41.000000	72184.250000	70286.500000	6.801950e+04	62630.000000	58516.250000	57462.500000	6005.00000
max	30000.000000	1000000.000000	79.000000	964511.000000	983931.000000	1.664089e+06	891586.000000	927171.000000	961664.000000	873552.00000

- The dataset has 30000 rows and 25 columns
- A total number of elements in the dataset:750000

```
data_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

We can infer that our dataset has 25 columns and 3000 rows (as previously shown in the report). The information on the dataset reported below show that there are no missing features for any of the $30,000$ samples.

ID	0
LIMIT_BAL	0
SEX	0
EDUCATION	0
MARRIAGE	0
AGE	0
PAY_1	0
PAY_2	0
PAY_3	0
PAY_4	0
PAY_5	0
PAY_6	0
BILL_AMT1	0
BILL_AMT2	0
BILL_AMT3	0
BILL_AMT4	0
BILL_AMT5	0
BILL_AMT6	0
PAY_AMT1	0
PAY_AMT2	0
PAY_AMT3	0
PAY_AMT4	0
PAY_AMT5	0
PAY_AMT6	0
def_pay	0
dtype: int64	

Null or Missing values in dataset

	ID	BILL_AMT2	PAY_AMT6	PAY_AMT5	PAY_AMT4	PAY_AMT3	PAY_AMT2	PAY_AMT1	BILL_AMT6	BILL_AMT5	...	PAY_5	PAY_4	PAY_3	PAY_2	PAY_
Total	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.
Percent	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.

This returns the number of null values in each column. As we can see, There is no missing data in the entire dataset. Also, we determined what percentage of the null values are present in each column. There are columns that have 0% null values.

Data Preprocessing

Client Personal Information

```
data_df[['LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE']].describe()
```

	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	167484.322667	1.603733	1.853133	1.551867	35.485500
std	129747.661567	0.489129	0.790349	0.521970	9.217904
min	10000.000000	1.000000	0.000000	0.000000	21.000000
25%	50000.000000	1.000000	1.000000	1.000000	28.000000
50%	140000.000000	2.000000	2.000000	2.000000	34.000000
75%	240000.000000	2.000000	2.000000	2.000000	41.000000
max	1000000.000000	2.000000	6.000000	3.000000	79.000000

```
df['EDUCATION'].value_counts().sort_i
0       14
1    10585
2    14030
3     4917
4      123
5      280
6       51
Name: EDUCATION, dtype: int64
```

```
df['MARRIAGE'].value_count
0       54
1    13659
2    15964
3      323
Name: MARRIAGE, dtype: int
```

Changing the incorrect attribute or removing the rows involved in the issue can fix errors in the dataset. We could be cautious and put the undocumented categories into another category, but since there aren't many anomalous entries (399, or 1.33% of total), we choose to remove the anomalies in the MARRIAGE column and combine the 0, 5, and 6 categories in EDUCATION into category 4, or "others" only.

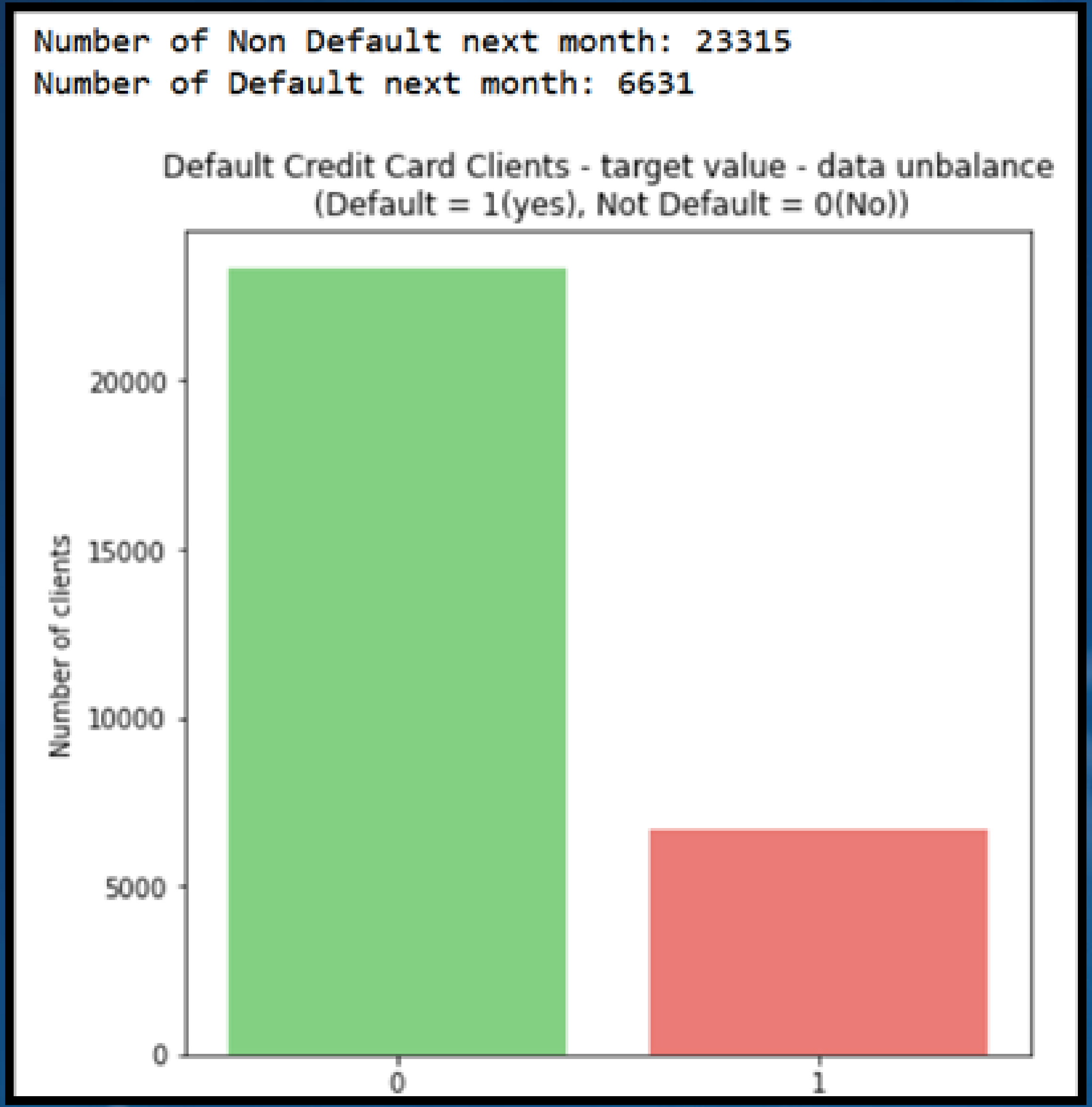
History of Past Payments

```
df[['PAY_1', 'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6']].describe()
```

	PAY_1	PAY_2	PAY_3	PAY_4	PAY_5	PAY_6
count	29601.000000	29601.000000	29601.000000	29601.000000	29601.000000	29601.000000
mean	-0.014932	-0.131313	-0.163440	-0.218303	-0.263978	-0.287558
std	1.124503	1.199642	1.199793	1.172220	1.136217	1.152206
min	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000	-2.000000
25%	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000	-1.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	8.000000	8.000000	8.000000	8.000000	8.000000	8.000000

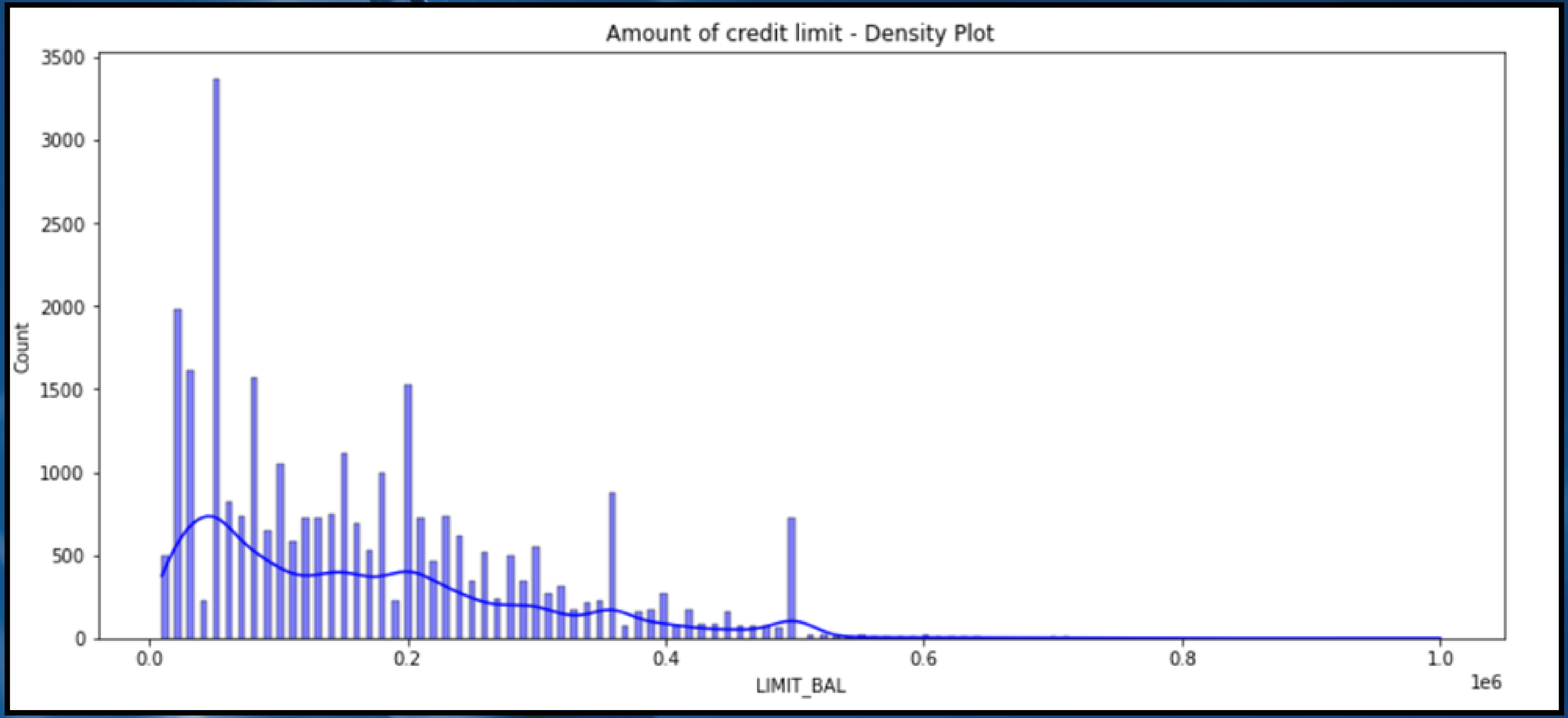
Data Exploration

Number of defaulters and non-defaulters in next month Interpretation



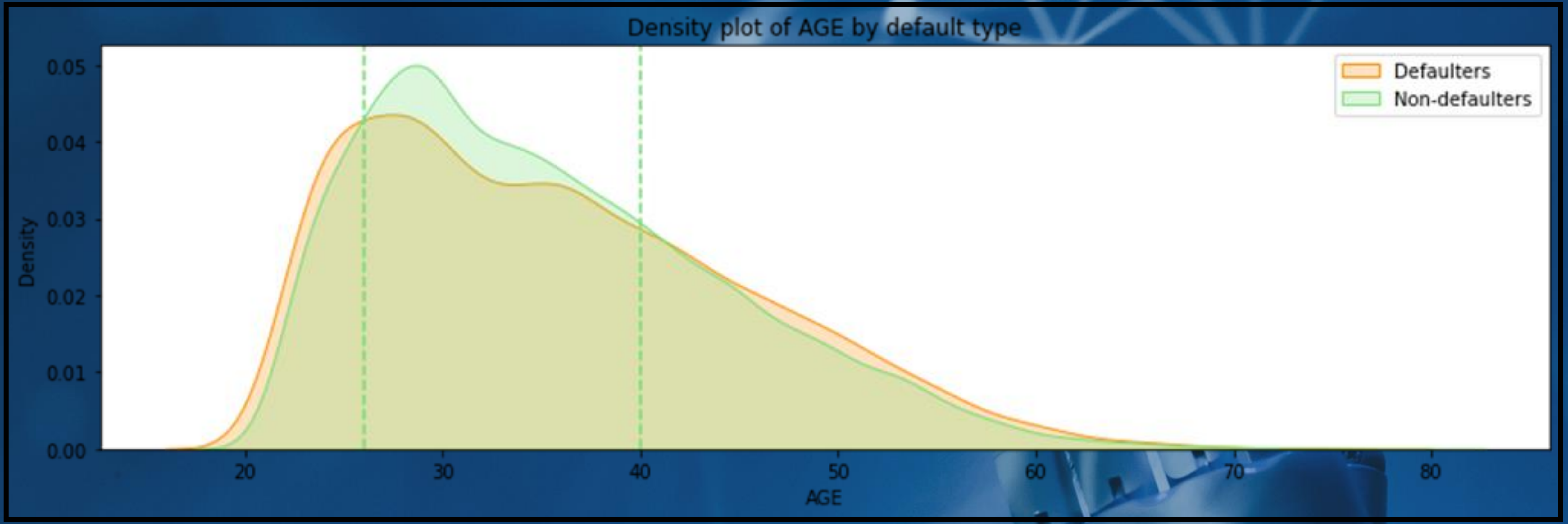
we found out that there are 23315 non-defaulters next month and 6631 defaulters next month, which is 3.5 times lower than the non-defaulters

Credit card limit distribution across data



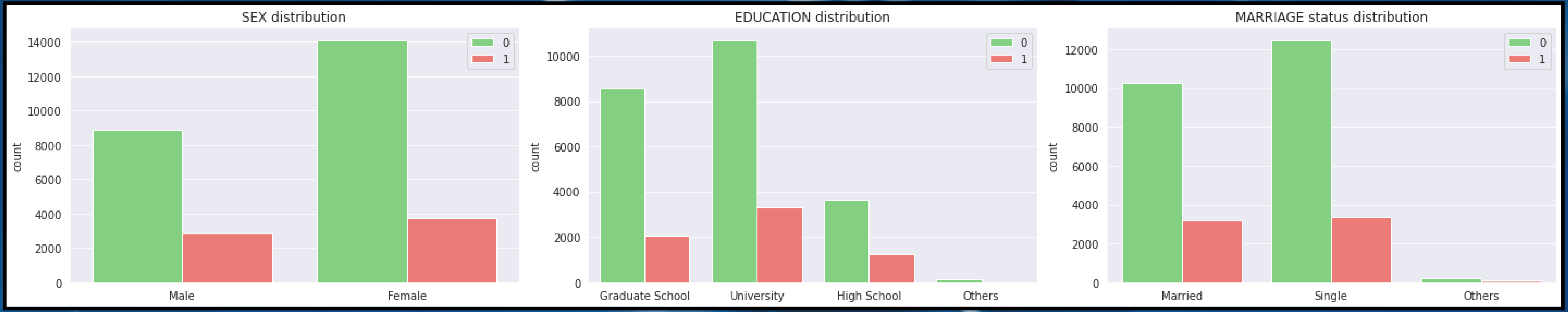
we observe that dataset consists of skewed data of limiting balance. We have more number of clients having limiting balance between 0 to 200000 currency

AGE Distribution by default type

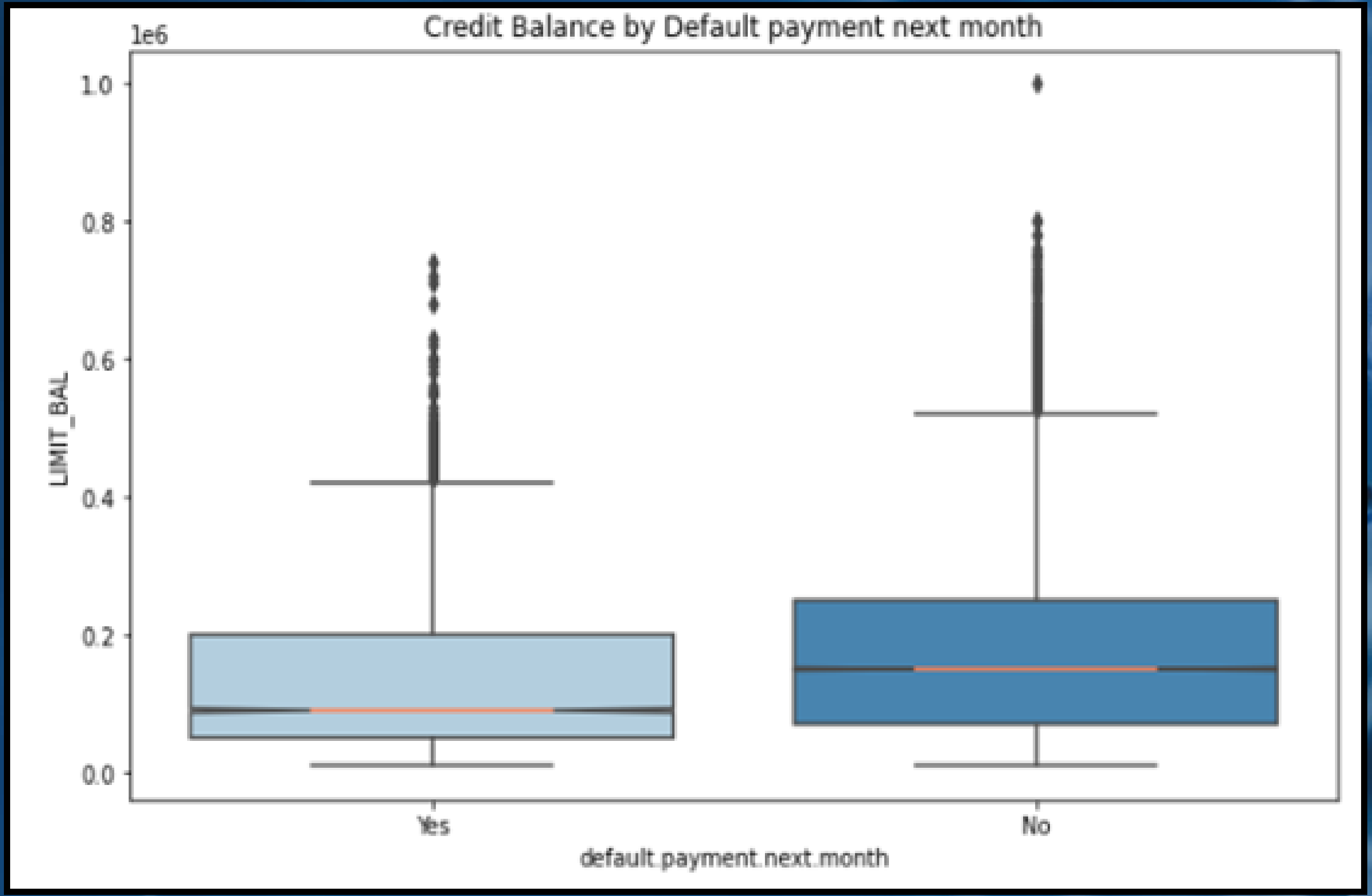


The probability of non-default of age between approximately 25 and 40 is higher, which indicates that consumers in this age group are more capable of repaying credit card loans

Relation between Sex, Education and Marraige

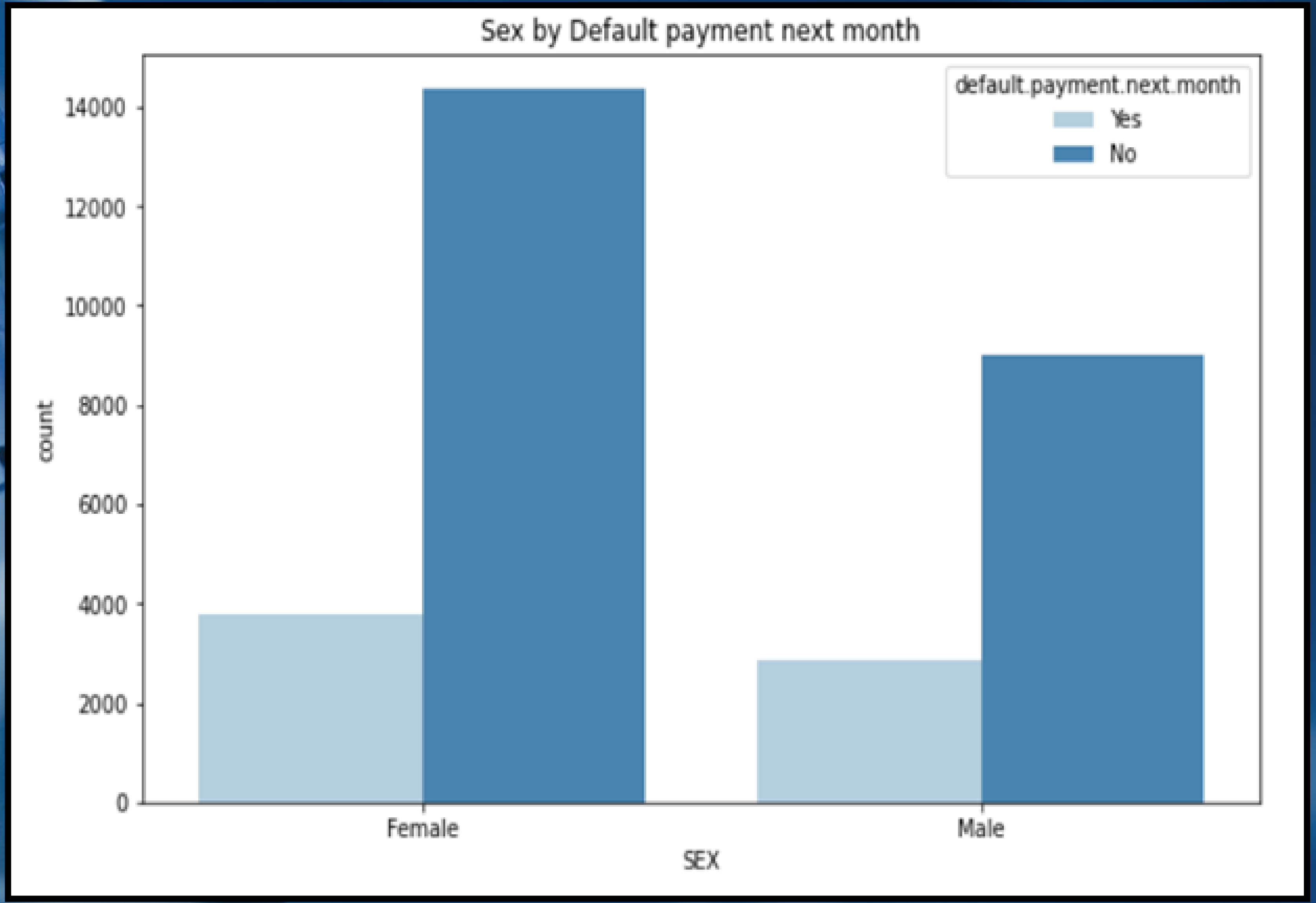


Relation: Limit or Credit Bal by Default Payment next month



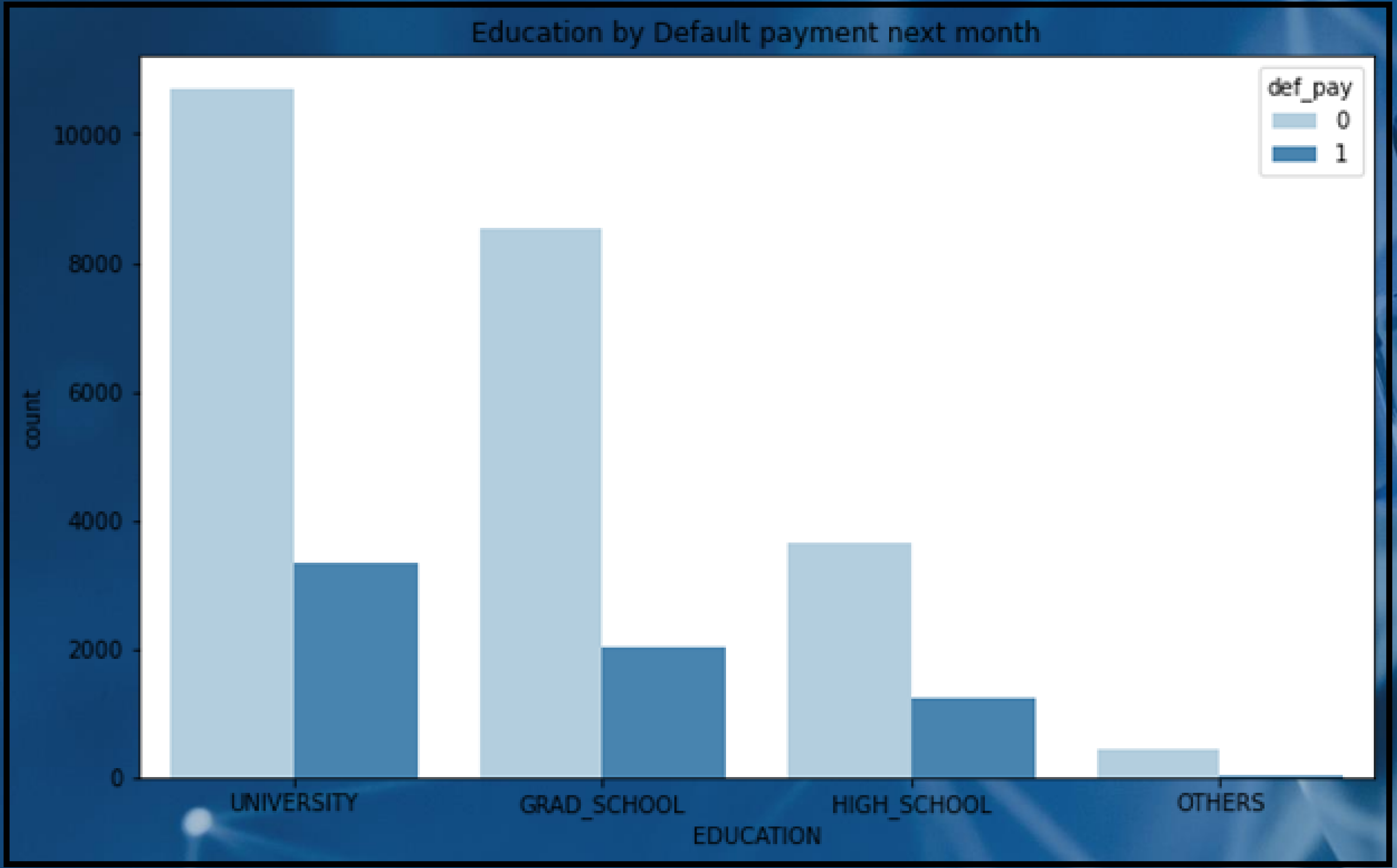
The median credit card limit balance of the customers who will default next month is less than the median credit card limit balance of the customers who will not default

Relation: Sex by Default payment next month



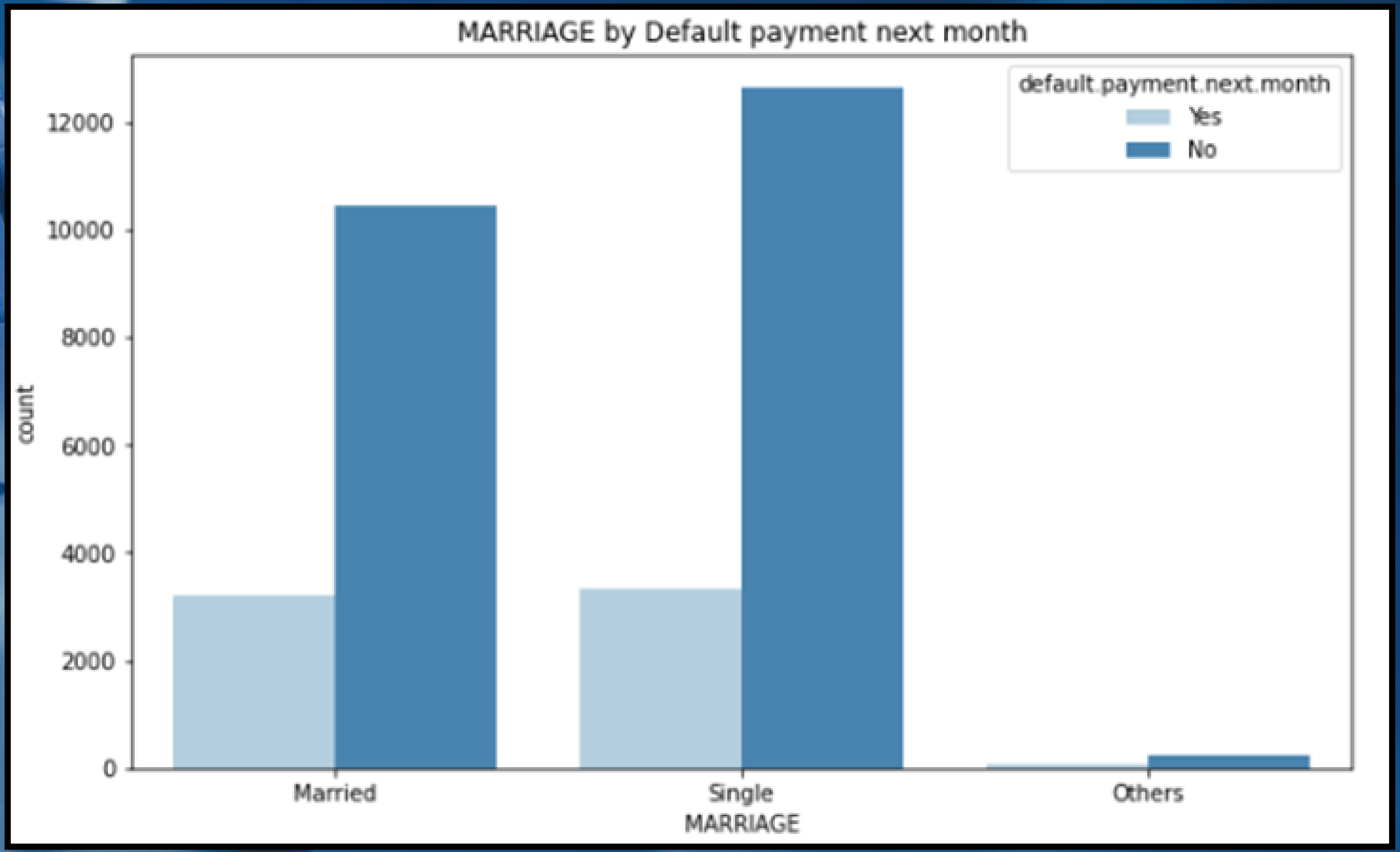
The number of defaults for the next month is almost the same for men and women, however non-defaulter payments for the next month are highest for the women compared to men by 7000

Relation: Education by Default payment next month

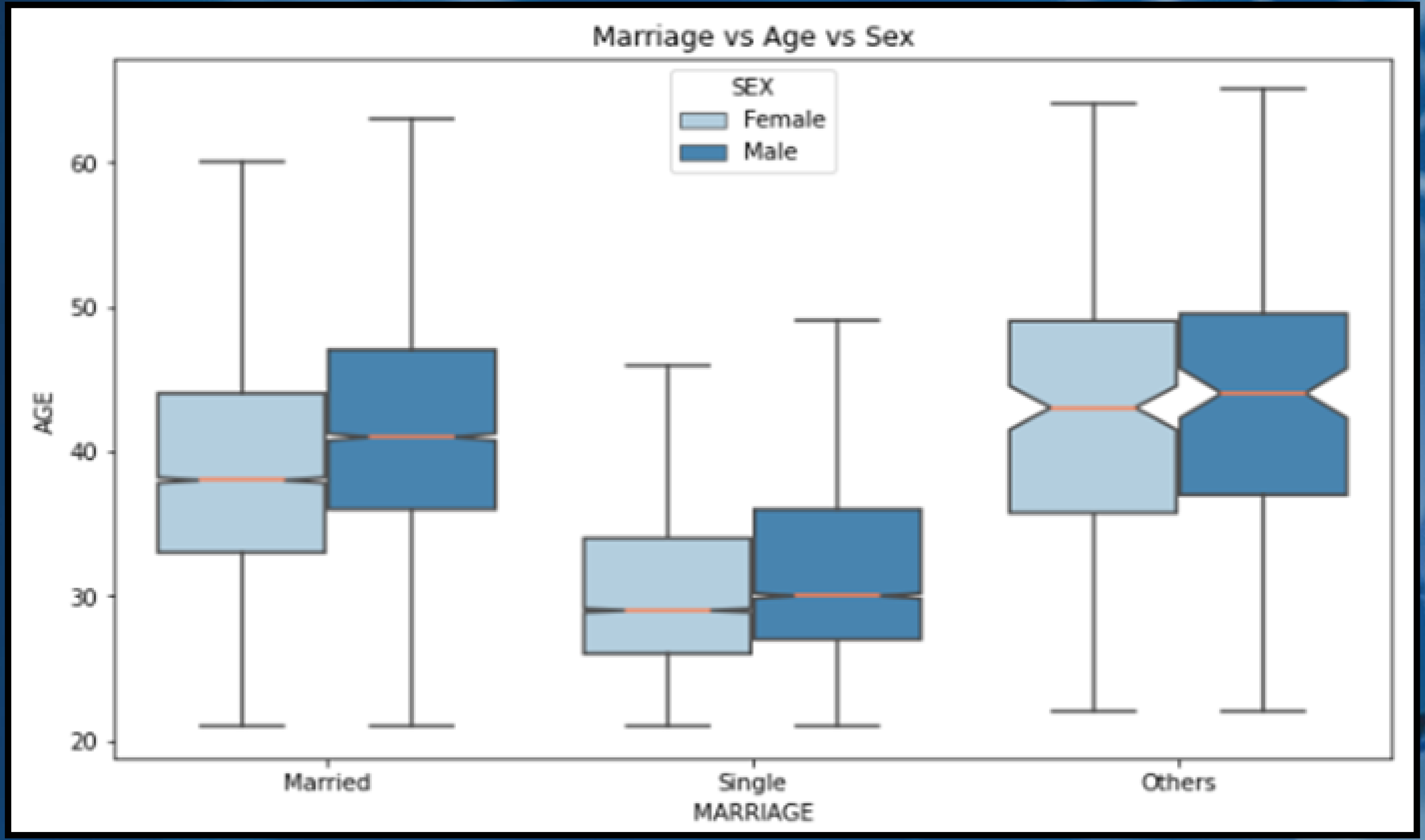


The qualification increases the non-defaulter list is increasing as well as the defaulter list, although defaulters are 1/3rd of the count compared to the non-defaulters

Relation: Marriage by Default payment next month



The number of defaulters in credit card payments is nearly the same for married credit card users as compared to the credit card users who are single

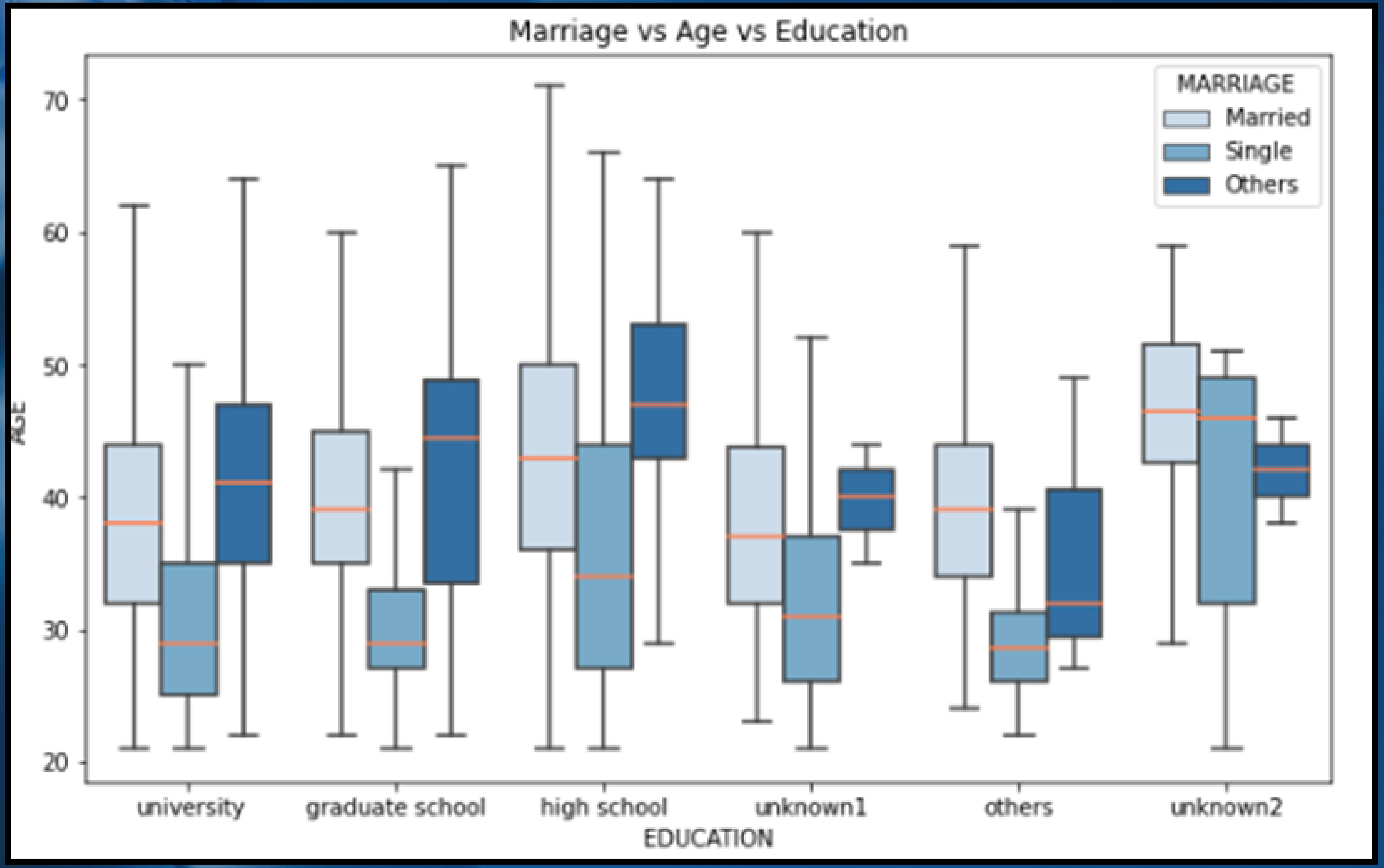


Relation: Marriage vs Sex vs Age

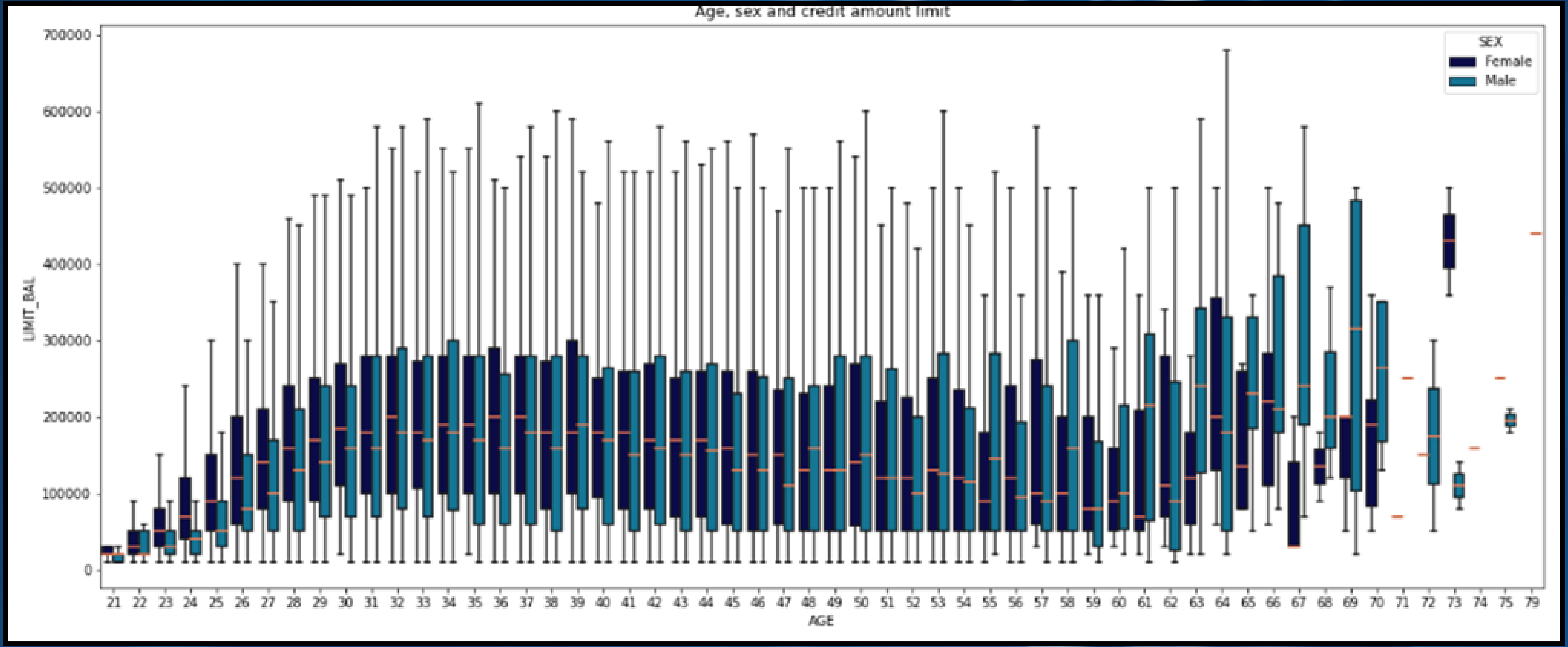
the mean age for both men and women is nearly the same in the case of singles and others but in the case of the Married category there is a notable difference in the mean with mean age of men being more than that of women

Relation: Marriage vs Age vs Education

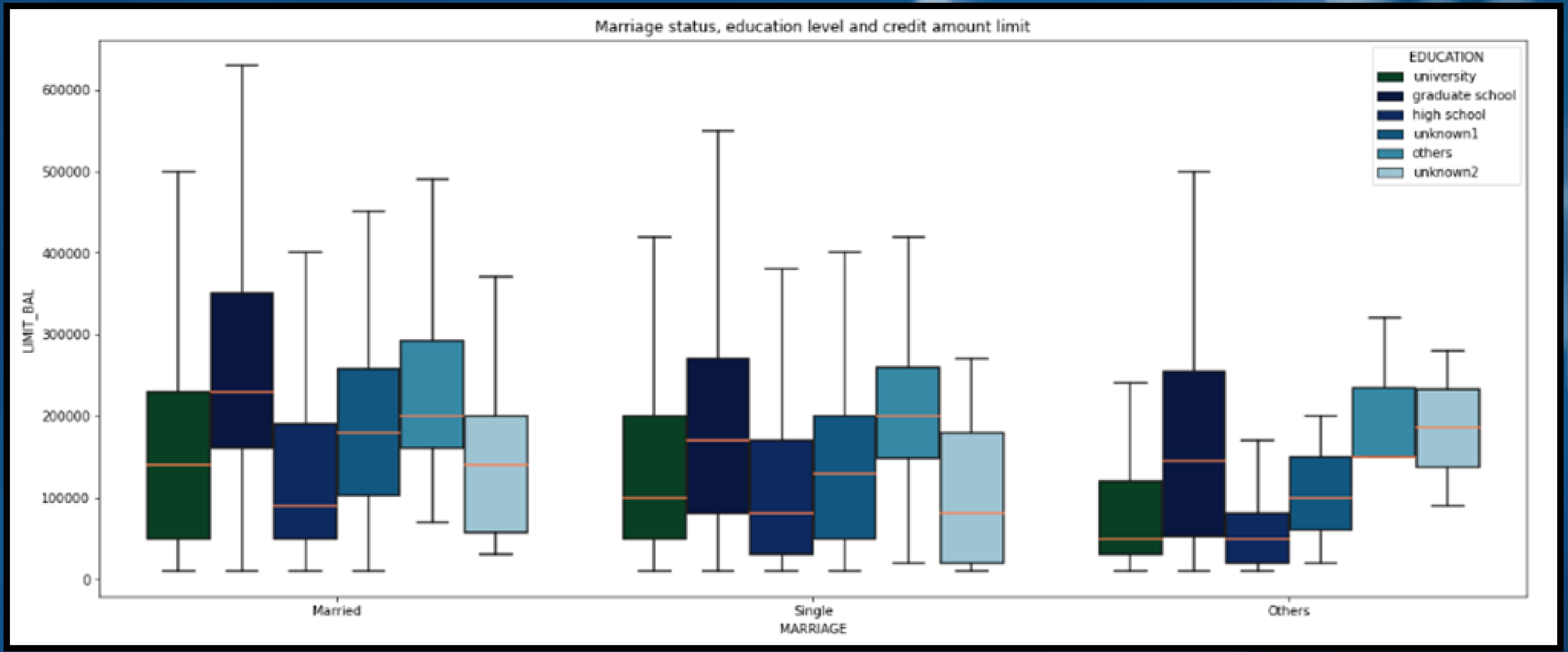
The distribution of the people based on Qualification and Marriage, here we can say that 40 to 50 years age are in the major mean areas, the highest distribution of age was for high school followed by university and graduate school



Relation: Age vs Sex vs Credit amount limit



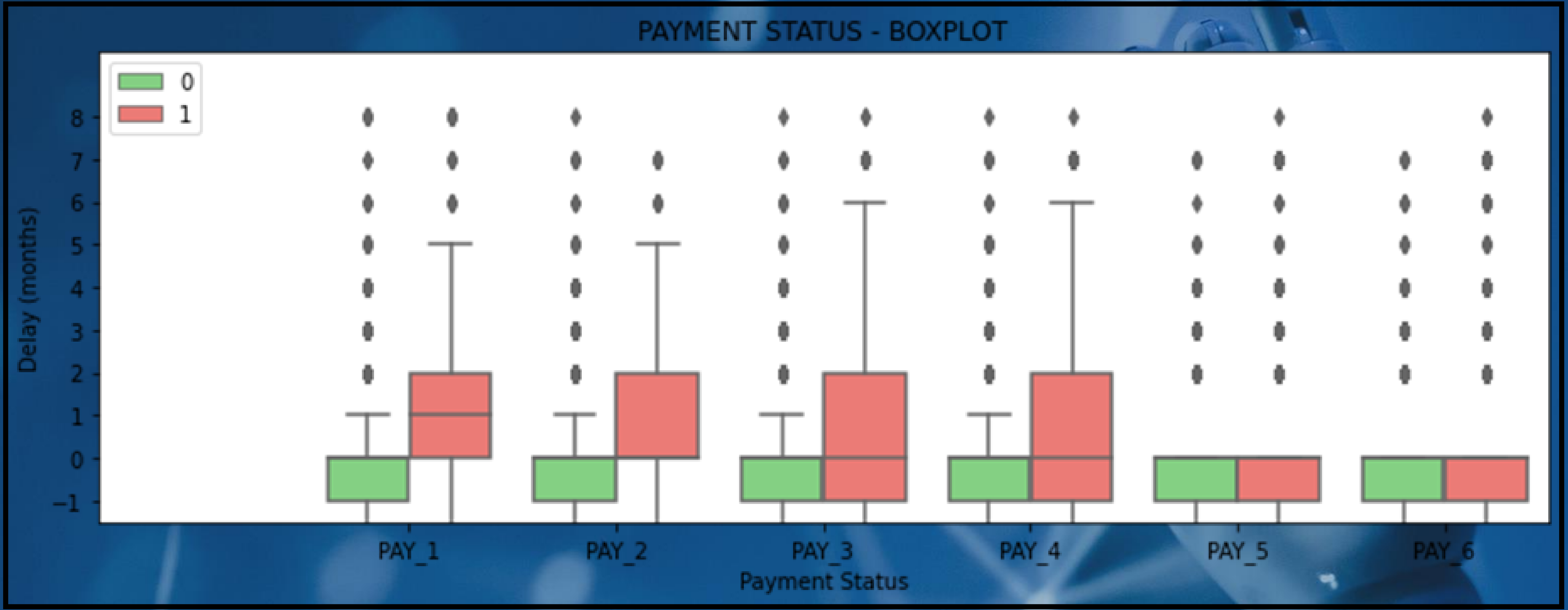
The relationship between credit card limit balances with the respective individual ages shown by male and female, and the distribution across the credit card limit balances spread over the respective range for those ages.



Relation: Marriage status vs Education vs Credit limit

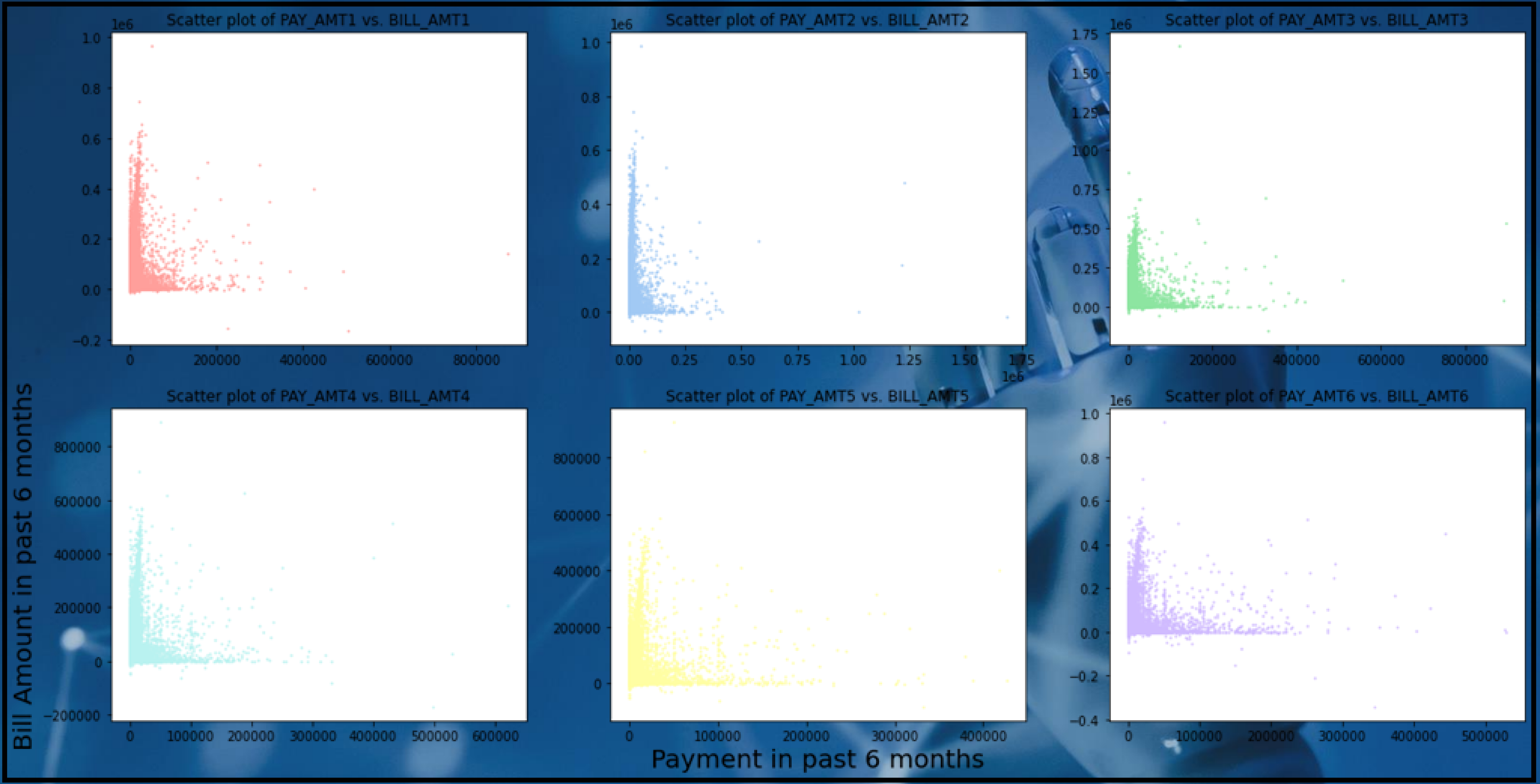
the averages for the different age groups for the limit balances do not coincide for the respective marital statuses. Also, the credit limit is highest for the married.

PAYMENT STATUS FEATURES (BY TARGET)



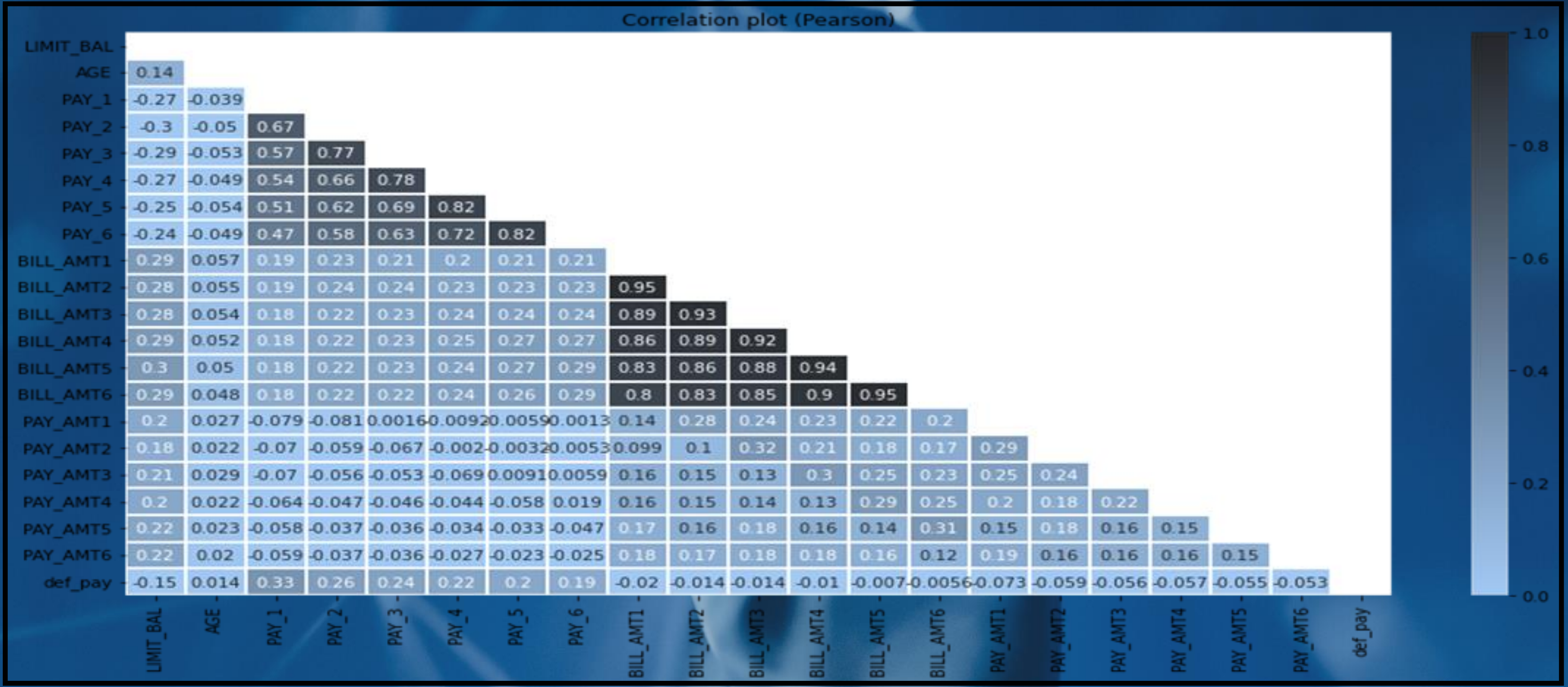
Clients who delay payment by one month or less have fewer credit card defaults. In particular, the repayment status in September, i.e., PAY_1, holds a greater discriminatory power than the repayment status in the other months

Past six months bill amount effect on the payment default next month or not:



There is higher proportion of clients for whom the bill amount is high but payment done against the same is very low. This we can infer since maximum number of datapoints are closely packed along the Y-axis near to 0 on X-axis.

Correlation between variables



There is a strong positive correlation between the BILL_AMTn features, which may indicate a redundancy of information. We see that despite BILL_AMTX variables having high correlation, it would be a mistake to just drop all of them because they are important for the explanation of dependent variables.

Data Modelling

Logistic Regression (Baseline Model)

Logistic Regression predicts event log odds using independent predictor variable

Rebuilding the model with only the significant variables

Dependent Variable:	def_pay	Pseudo R-squared:	0.127
Date:	2023-09-10 14:39	AIC:	19395.2421
No. Observations:	20962	BIC:	19514.4991
Df Model:	14	Log-Likelihood:	-9682.6
Df Residuals:	20947	LL-Null:	-11096.
Converged:	1.0000	LLR p-value:	0.0000
No. Iterations:	7.0000	Scale:	1.0000

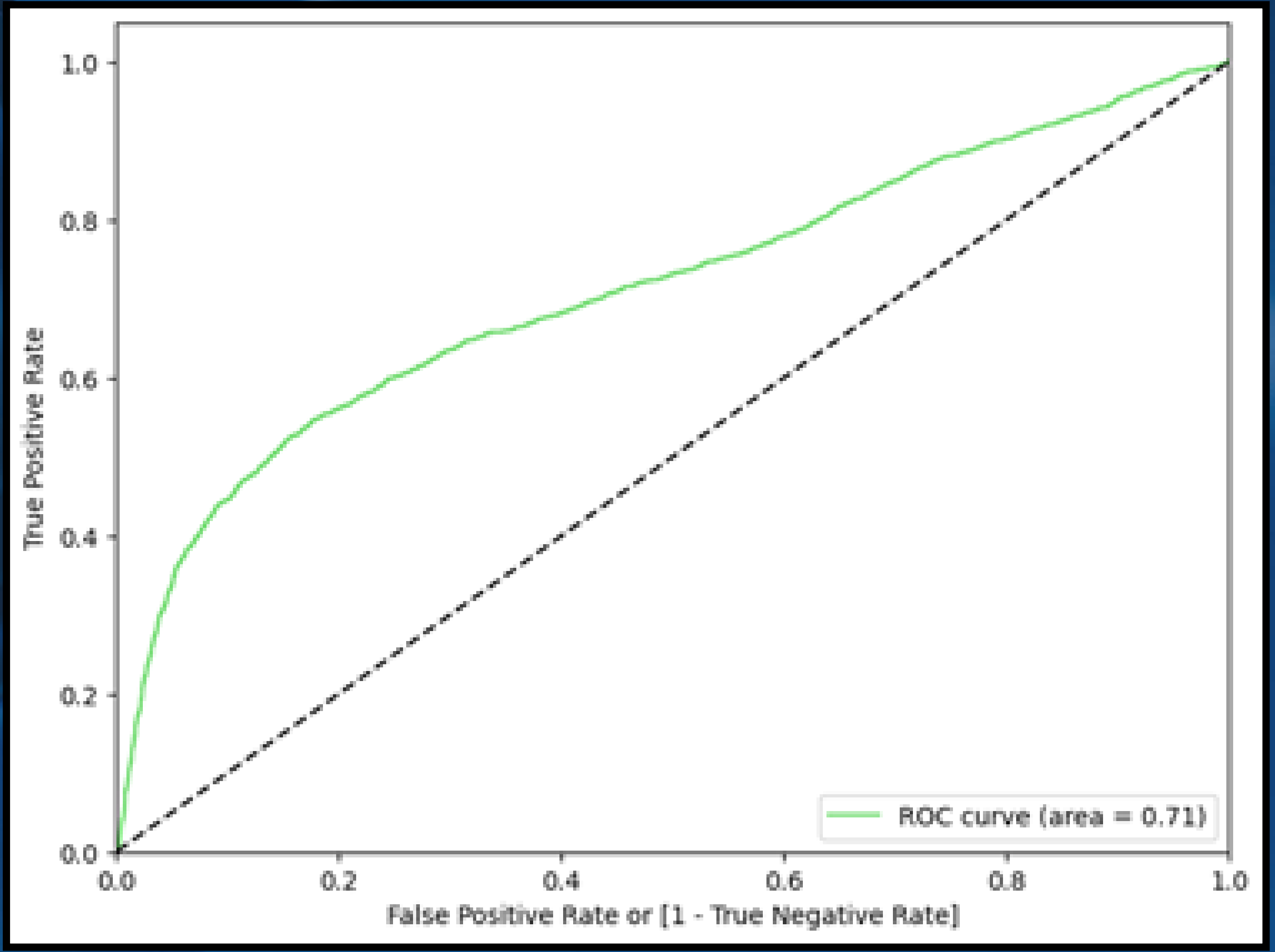
	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-1.3547	0.0436	-31.1034	0.0000	-1.4401	-1.2693
LIMIT_BAL	-0.1053	0.0240	-4.3920	0.0000	-0.1523	-0.0583
AGE	0.0510	0.0205	2.4876	0.0129	0.0108	0.0913
PAY_1	0.6917	0.0235	29.4635	0.0000	0.6457	0.7378
PAY_2	0.1001	0.0284	3.5231	0.0004	0.0444	0.1558
PAY_3	0.1340	0.0265	5.0640	0.0000	0.0821	0.1859
BILL_AMT1	-0.1477	0.0241	-6.1371	0.0000	-0.1948	-0.1005
PAY_AMT1	-0.2398	0.0452	-5.3073	0.0000	-0.3283	-0.1512
PAY_AMT2	-0.2527	0.0568	-4.4478	0.0000	-0.3640	-0.1413
PAY_AMT4	-0.0945	0.0328	-2.8857	0.0039	-0.1587	-0.0303
EDUCATION_HIGH_SCHOOL	-0.1685	0.0573	-2.9402	0.0033	-0.2809	-0.0562
EDUCATION_OTHERS	-1.0789	0.2208	-4.8857	0.0000	-1.5117	-0.6461
EDUCATION_UNIVERSITY	-0.1137	0.0426	-2.6711	0.0076	-0.1972	-0.0303

Extracting Predicted Probabilities

	actual	predicted_prob
23553	0	0.224467
5511	0	0.223787
13114	0	0.167725
20718	1	0.264988
20243	0	0.203457
18717	0	0.411267
8248	1	0.158560
19958	0	0.393926
6672	0	0.087302
25645	1	0.030971

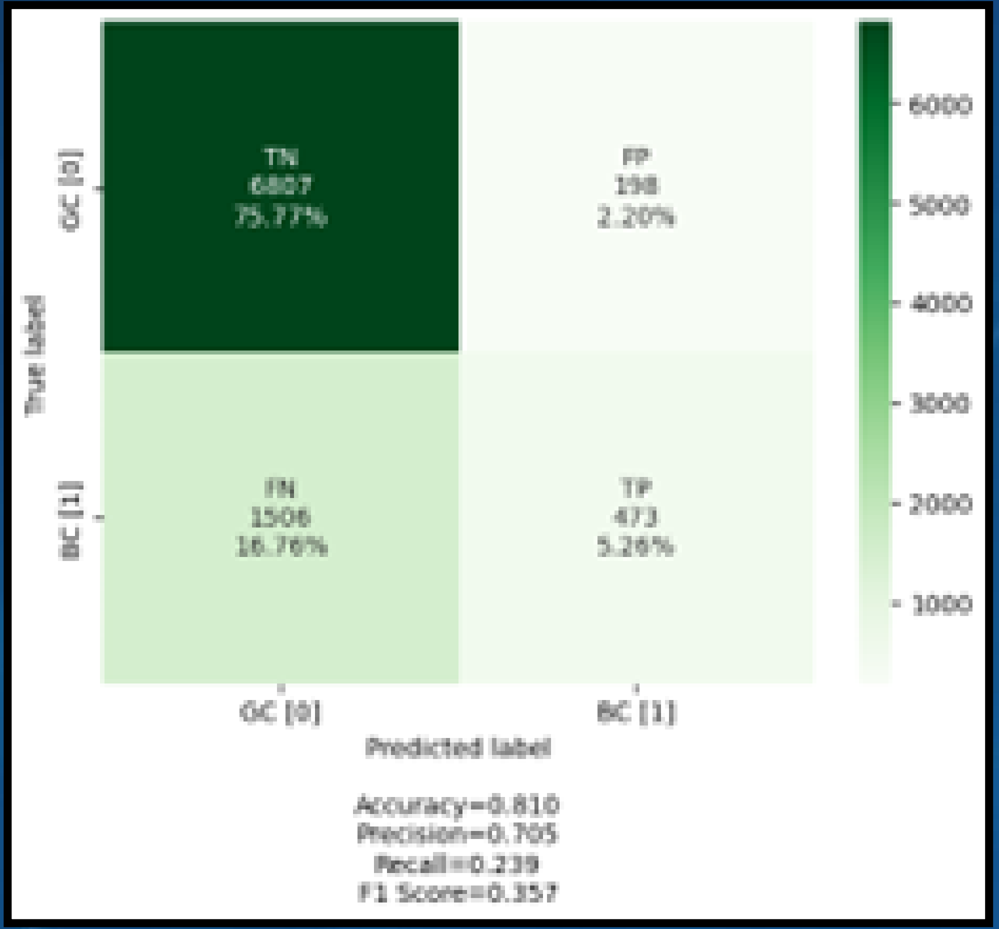
Measuring performance via the RoC Curve:

The AUC achieved here = 0.713

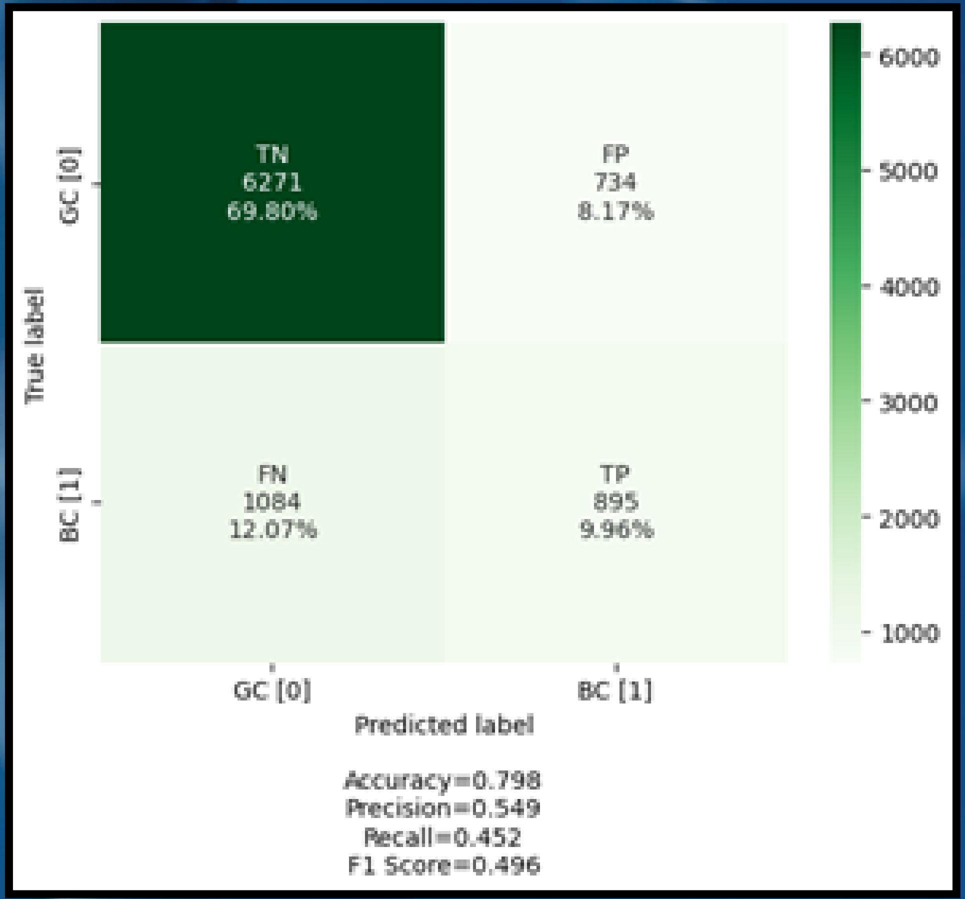


Measuring Performance of the Confusion Matrix

Accuracy = (TP + TN) / (TP + TN + FP + FN) - Proportion of objects rightly classified
Recall or Sensitivity = TP / (TP + FN) - Proportion of positives rightly classified
Specificity = TN/(TN + FP) - Proportion of negatives rightly classified
Precision = TP/(TP + FP) - Proportion of positives among all those classified a positives
F-Score : 2*Recall*Precision / (Recall + Precision) - Harmonic mean of precision and recall



Classifying on default threshold 0.5 and building the Confusion Matrix. The accuracy at threshold 0.5 = 0.810

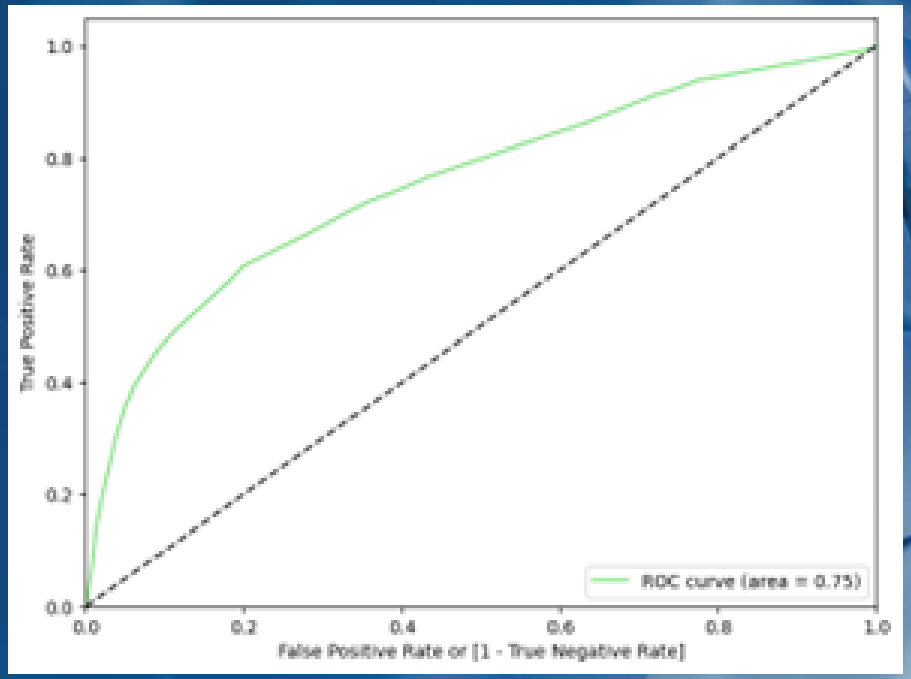


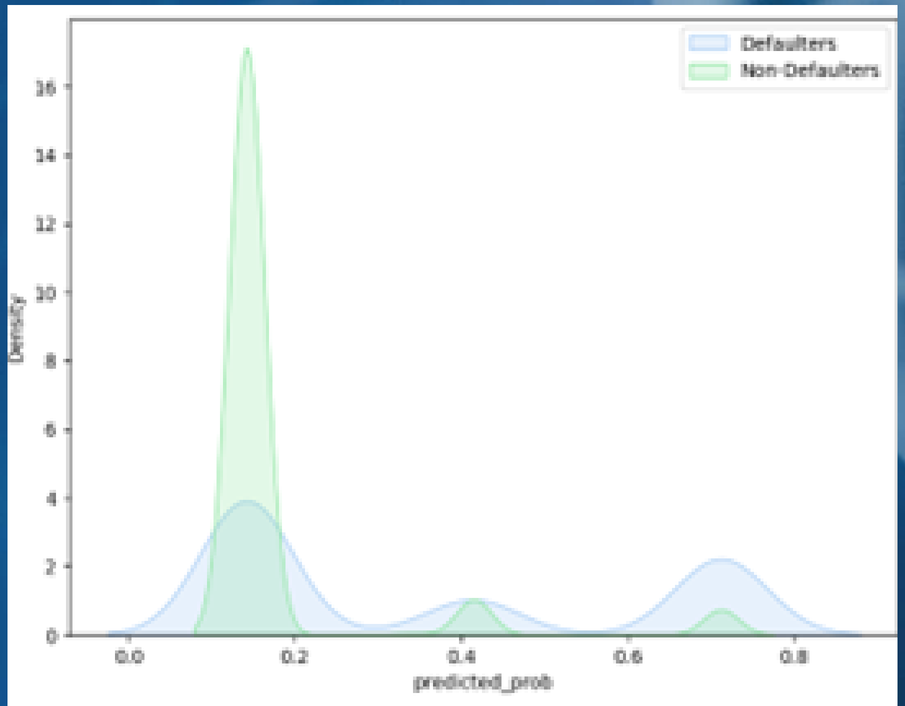
After rebuilding with better threshold, accuracy is 0.798

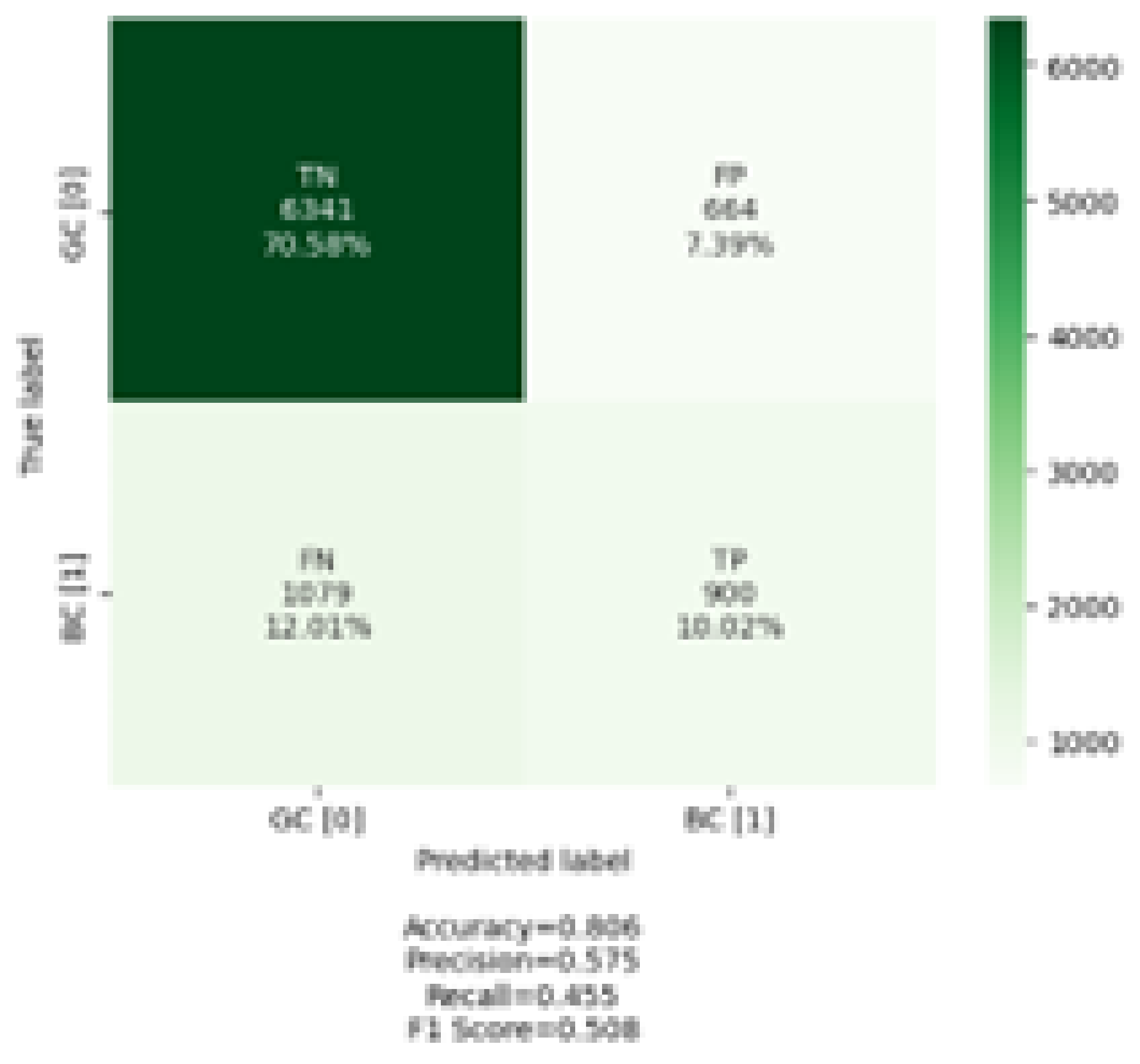
Decision Tree Classifier

Building the tree using Gini Criteria and extracting probabilities - Classifier performance = 75.4%

	actual	predicted_prob
5070	1	0.408271
6308	0	0.052035
22293	0	0.121332
18382	0	0.121332
6914	1	0.720482
15637	0	0.121332
22730	0	0.159046
17981	1	0.108191
29609	0	0.121332
11615	0	0.087193





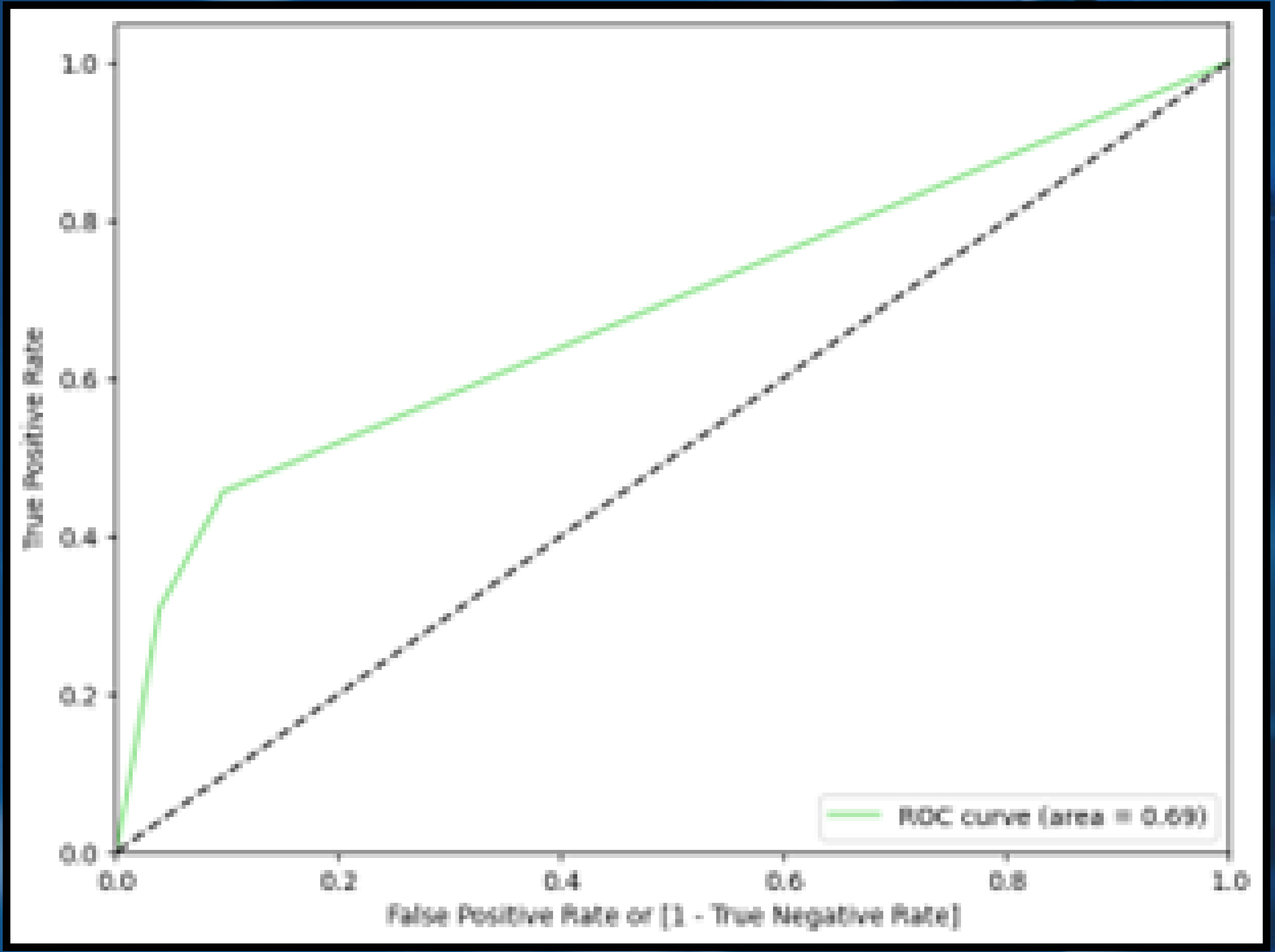


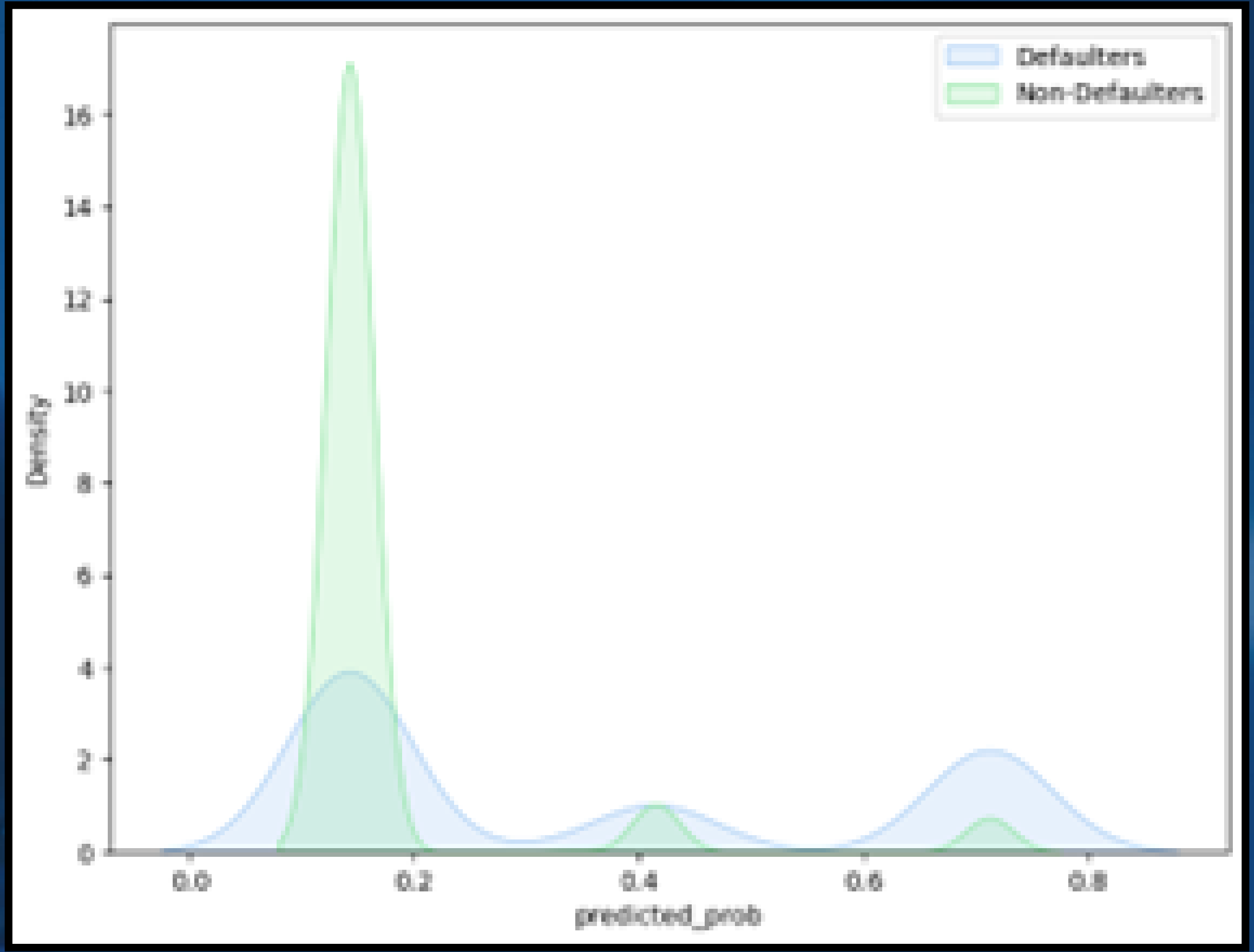
Interpretation: 7241 were correct predictions and 1743 were incorrect. Accuracy is 80.6%

GINI impurity of node 1 = 0.276
Entropy of Node 1 = 0.65

Extracting Probabilities and measuring Classifier performance, Plotting Distributions and Identifying Optimal Probability

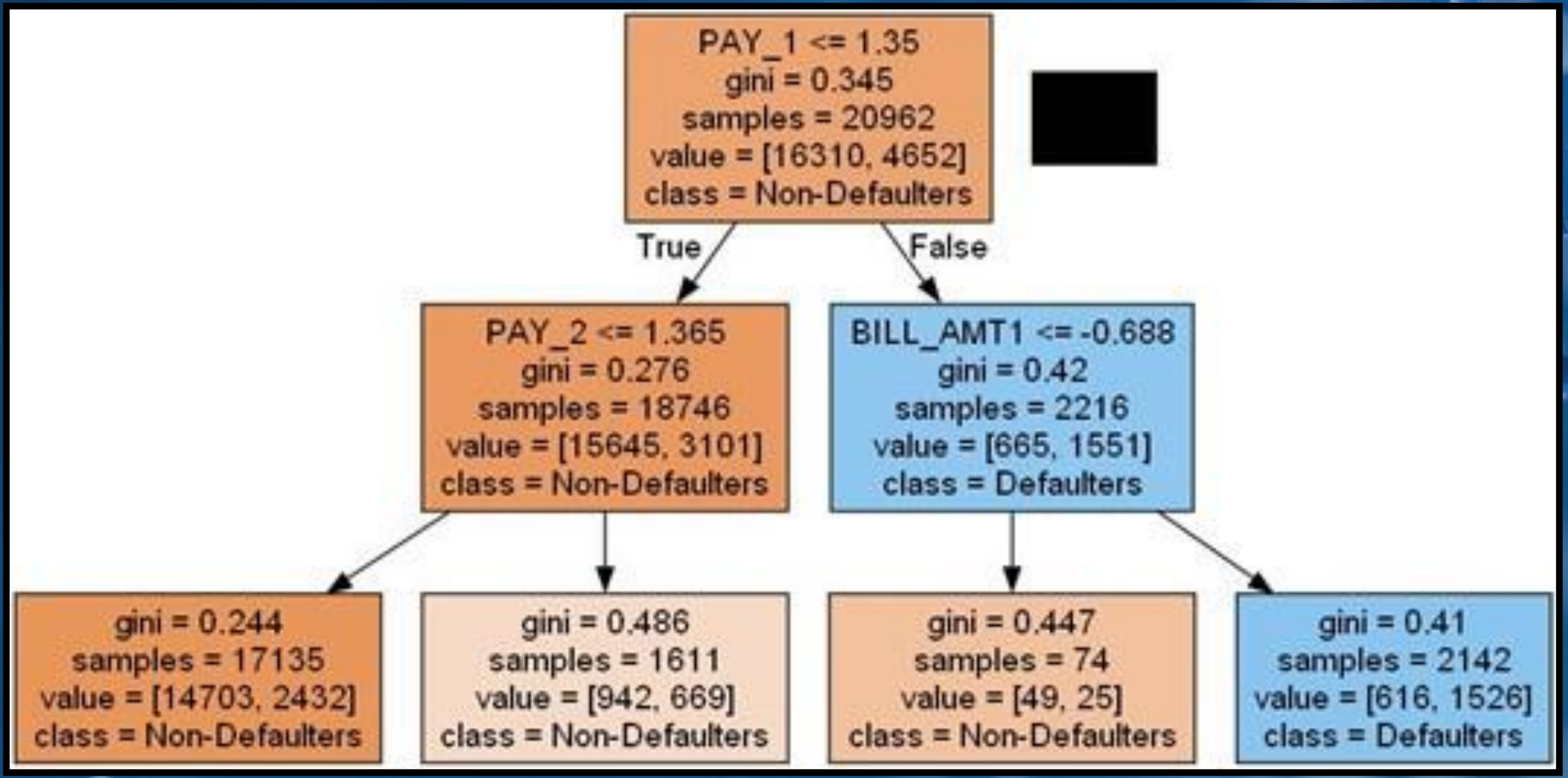
	actual	predicted_prob
5070	1	0.415270
6308	0	0.141932
22293	0	0.141932
18382	0	0.141932
6914	1	0.712418
15637	0	0.141932
22730	0	0.141932
17981	1	0.141932
29609	0	0.141932
11615	0	0.141932



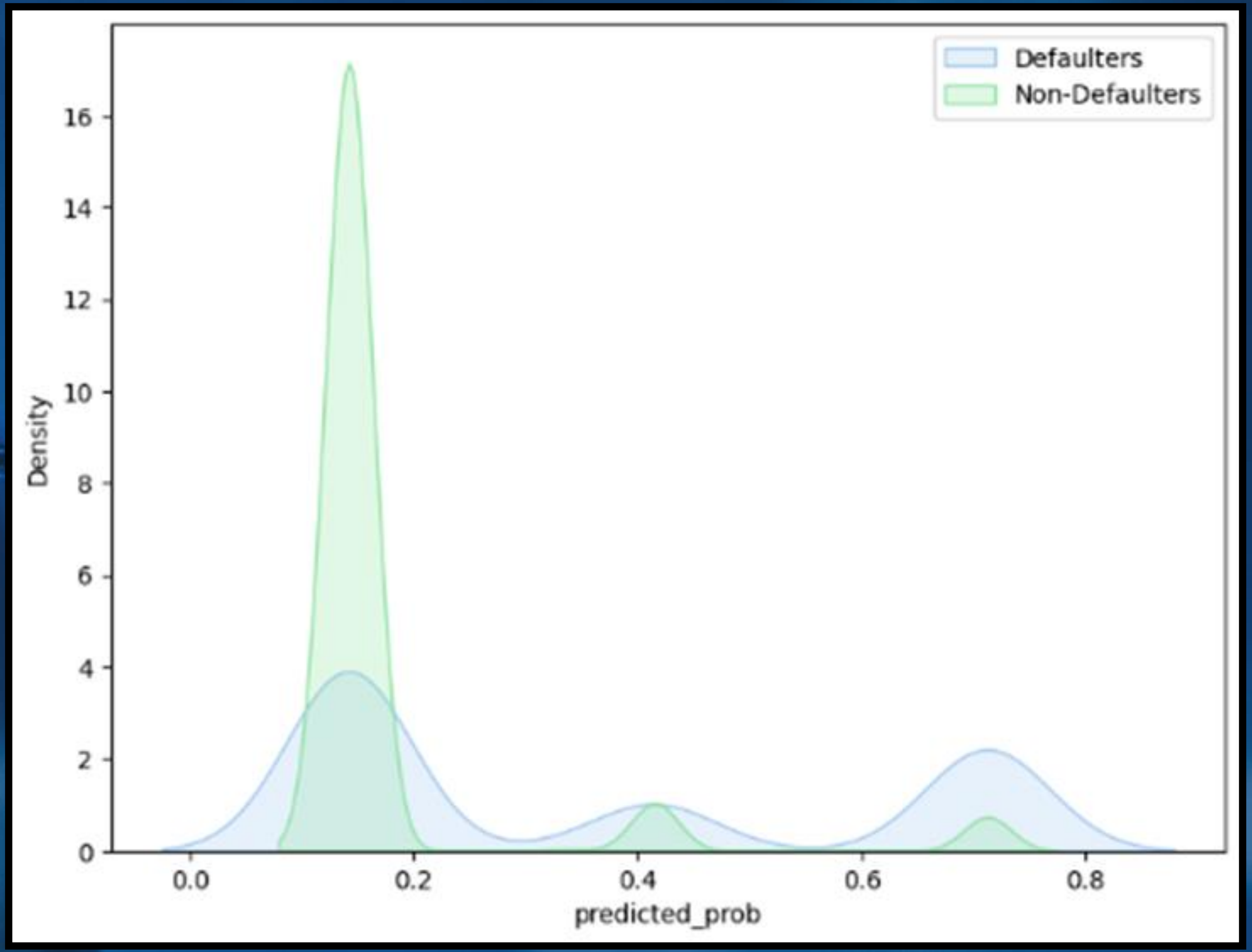


The Classifier performance is found to be 0.69

Decision Tree using Entropy Criteria



Plotting Distributions and Identifying Optimal Probability

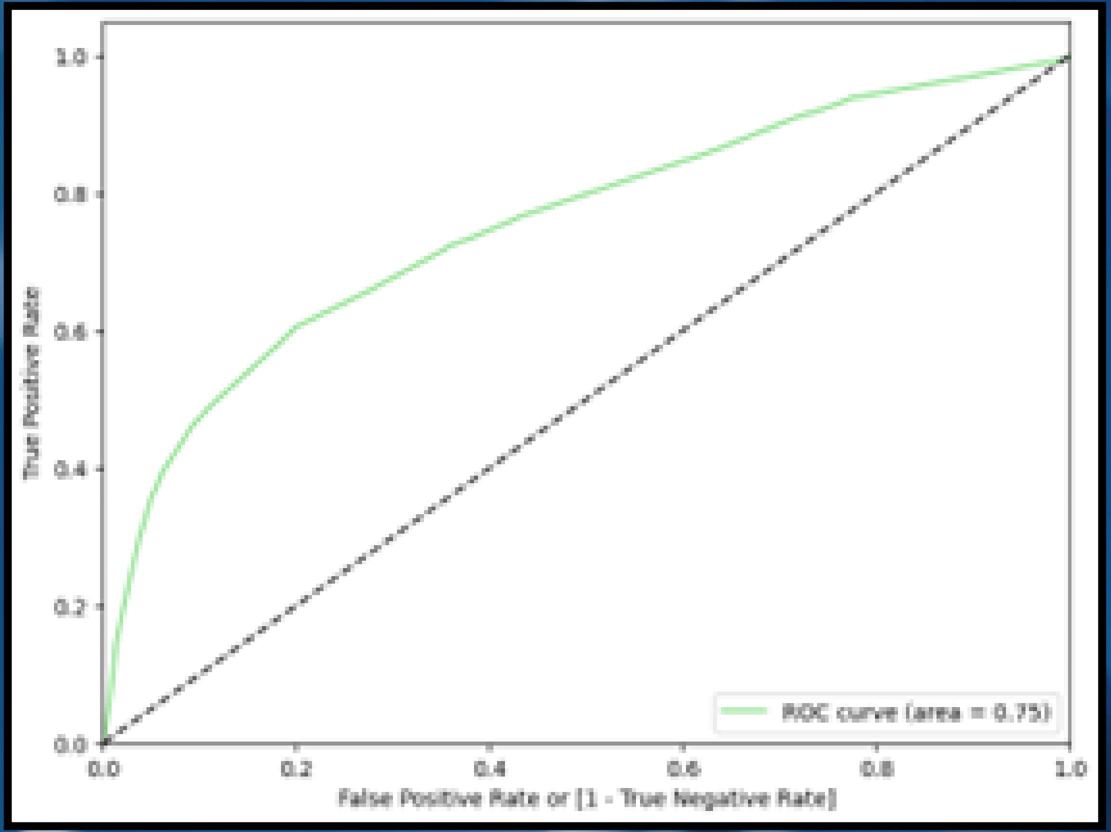


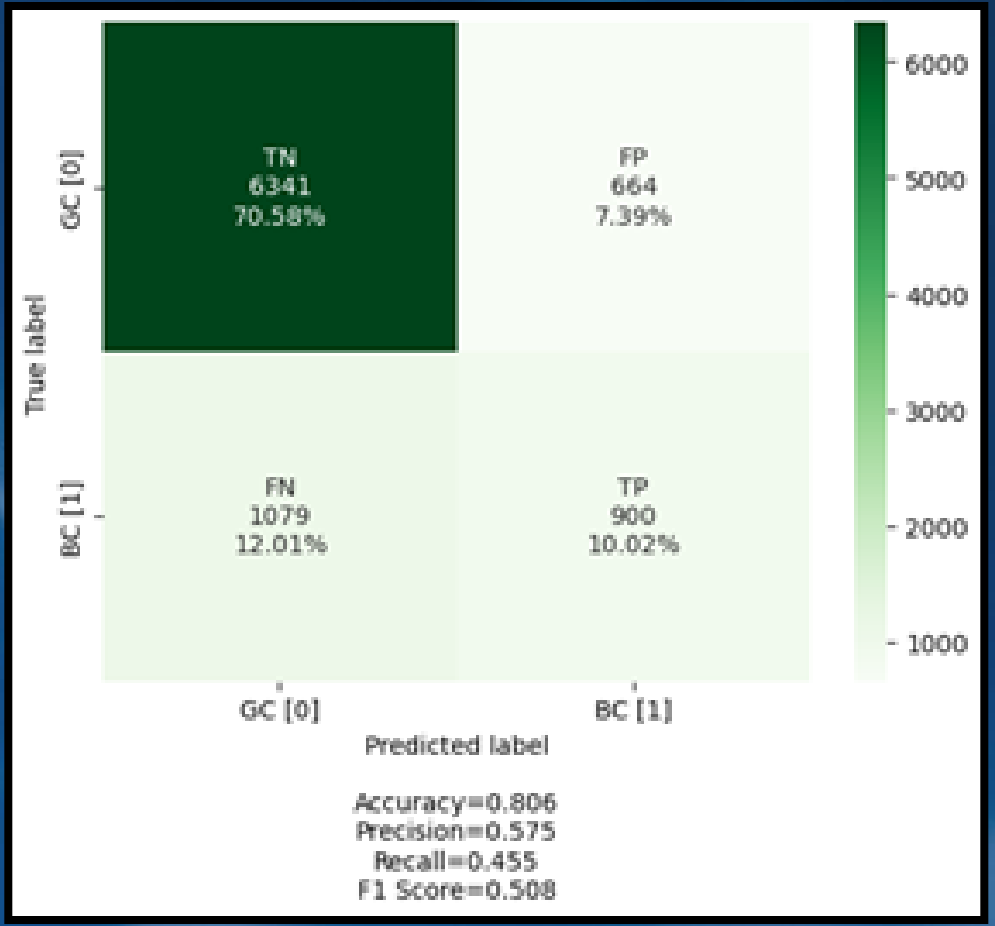
The threshold value is obtained by intersecting the Yes and No plots and is approximately 0.18.

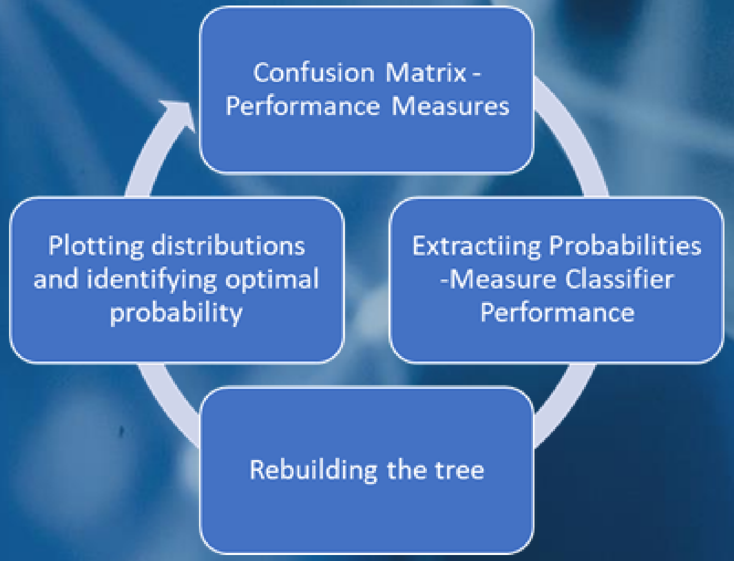
Therefore, probability exceeding 0.18 will be considered to be of class 1, and those below 0.18 will be considered to be of class 0.

Finding optimal criteria and max depth, rebuilding the tree with entropy criteria and max depth 6

	actual	predicted_prob
5070	1	0.408271
6308	0	0.052035
22293	0	0.121332
18382	0	0.121332
6914	1	0.720482
15637	0	0.121332
22730	0	0.159046
17981	1	0.108191
29609	0	0.121332
11615	0	0.087193

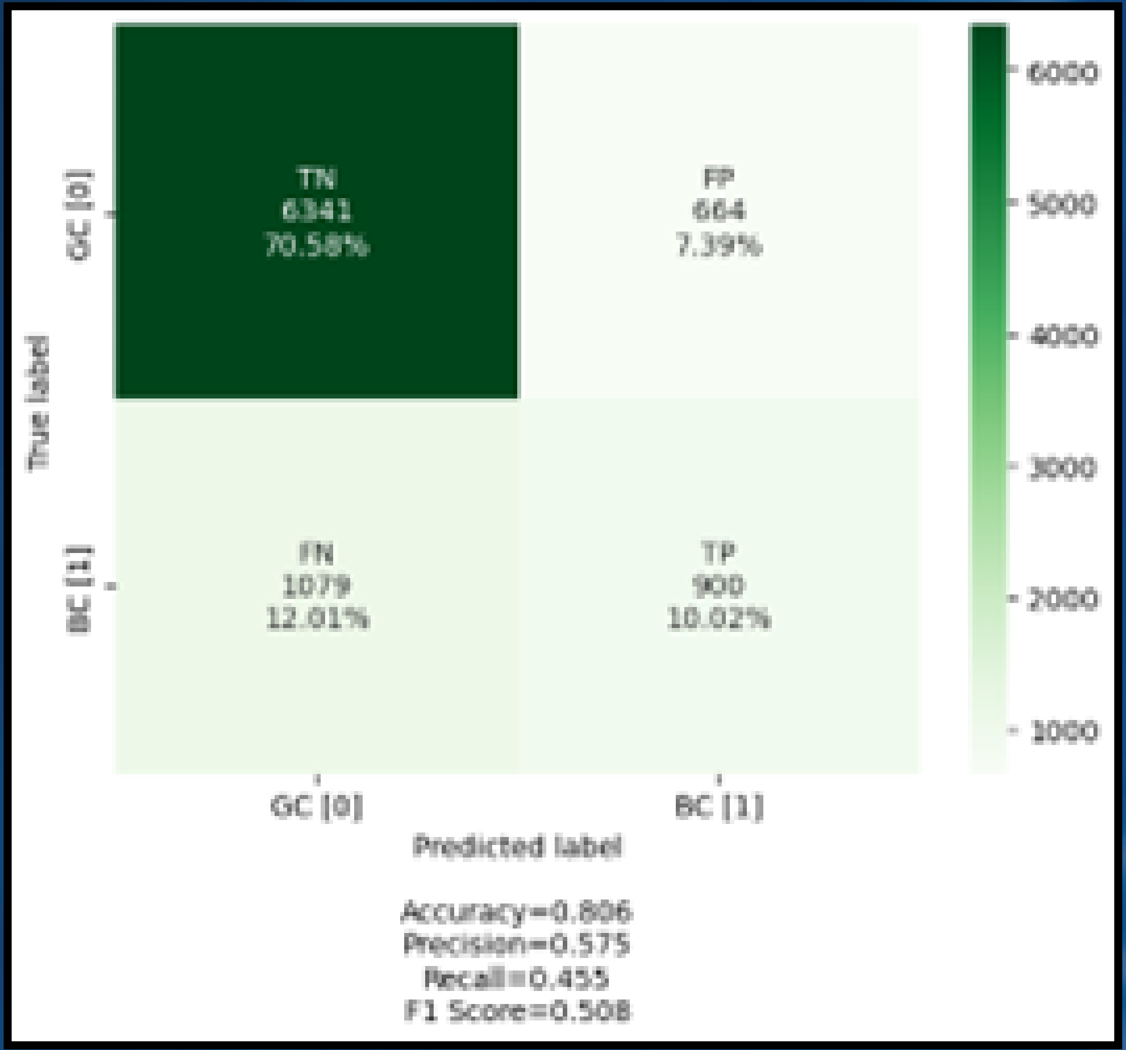






Interpretation: 7241 were correct predictions and 1743 were incorrect. Accuracy is 80.6%

Final Confusion Matrix - Classifier Interpretations



	precision	recall	f1-score	support
0	0.88	0.80	0.84	7005
1	0.46	0.60	0.52	1979
accuracy			0.76	8984
macro avg	0.67	0.70	0.68	8984
weighted avg	0.79	0.76	0.77	8984

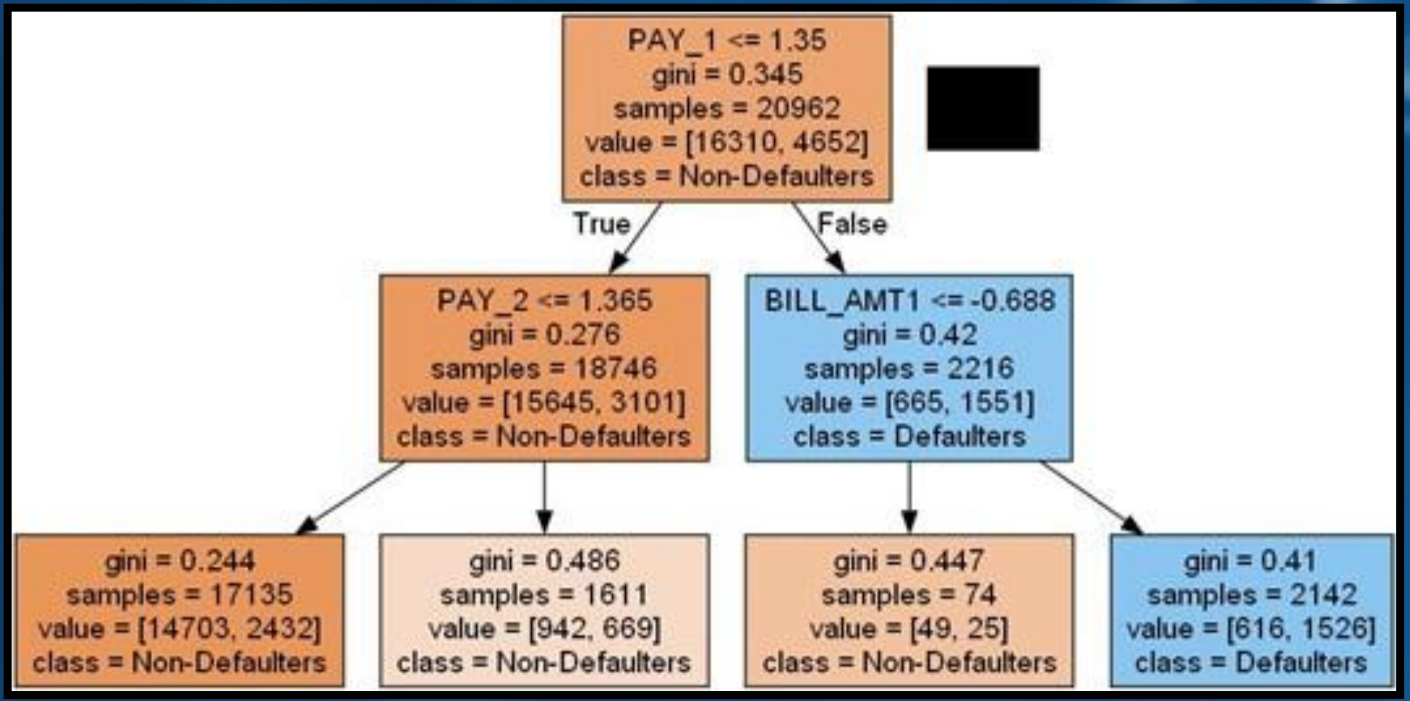
Interpretation: 7241 were correct predictions and 1743 were incorrect

Precision: 58% for defaulting clients, 85% for non-defaulting clients.

Recall: 45% for defaulting clients, 91% for non-defaulting clients.

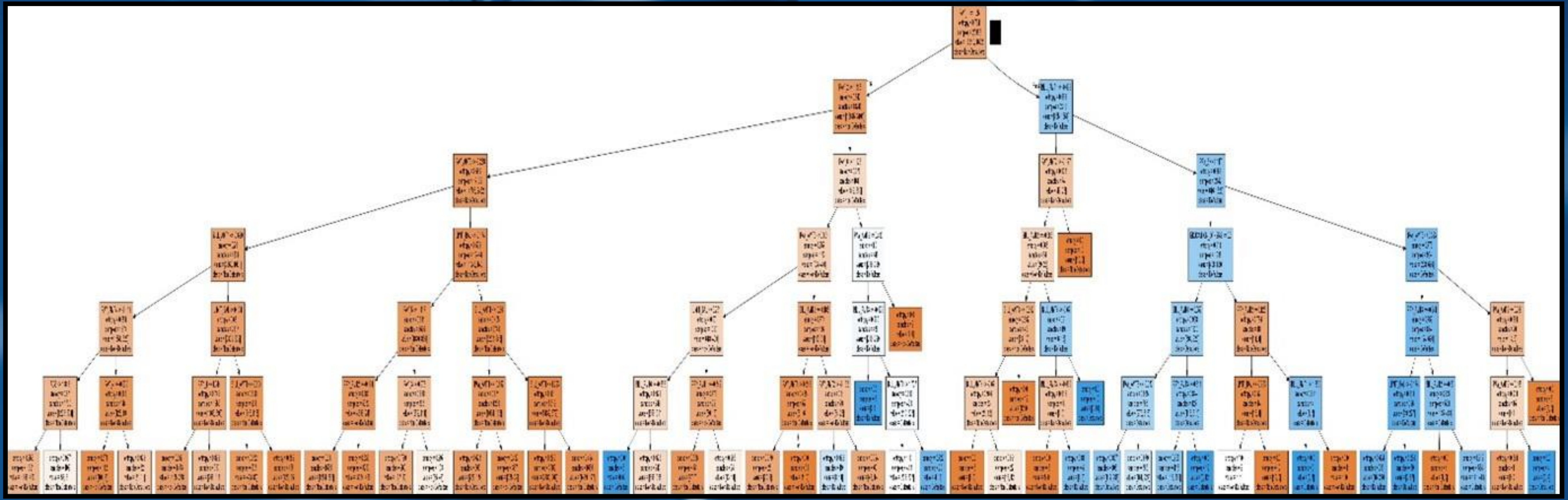
F-Score indicates the model does fairly well in predicting defaults.

Accuracy stands at 76%.



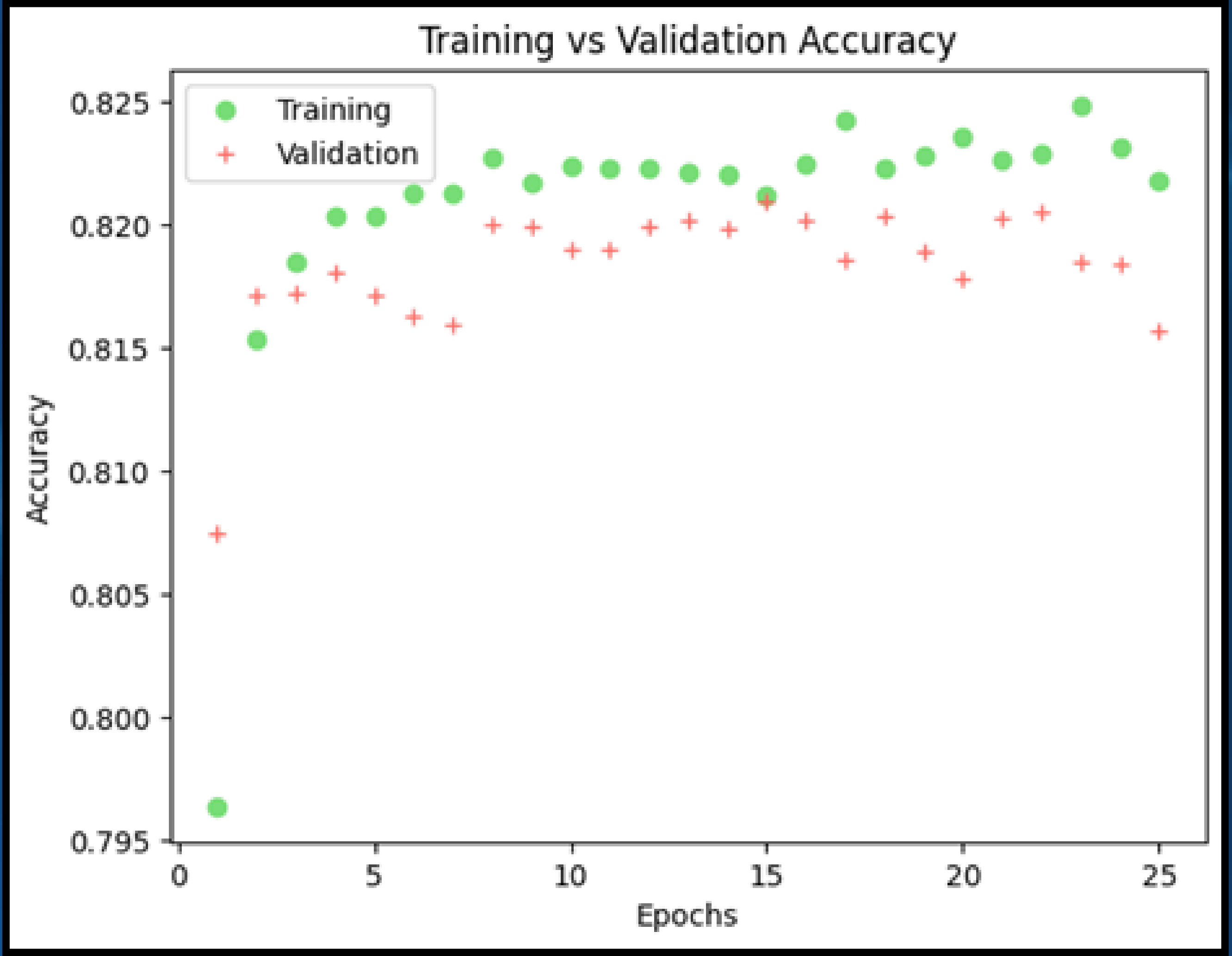
The threshold value is obtained by intersecting the Yes and No plots and is approximately 0.22. Therefore, probability exceeding 0.22 will be considered to be of class 1, and those below 0.18 will be considered to be of class 0.

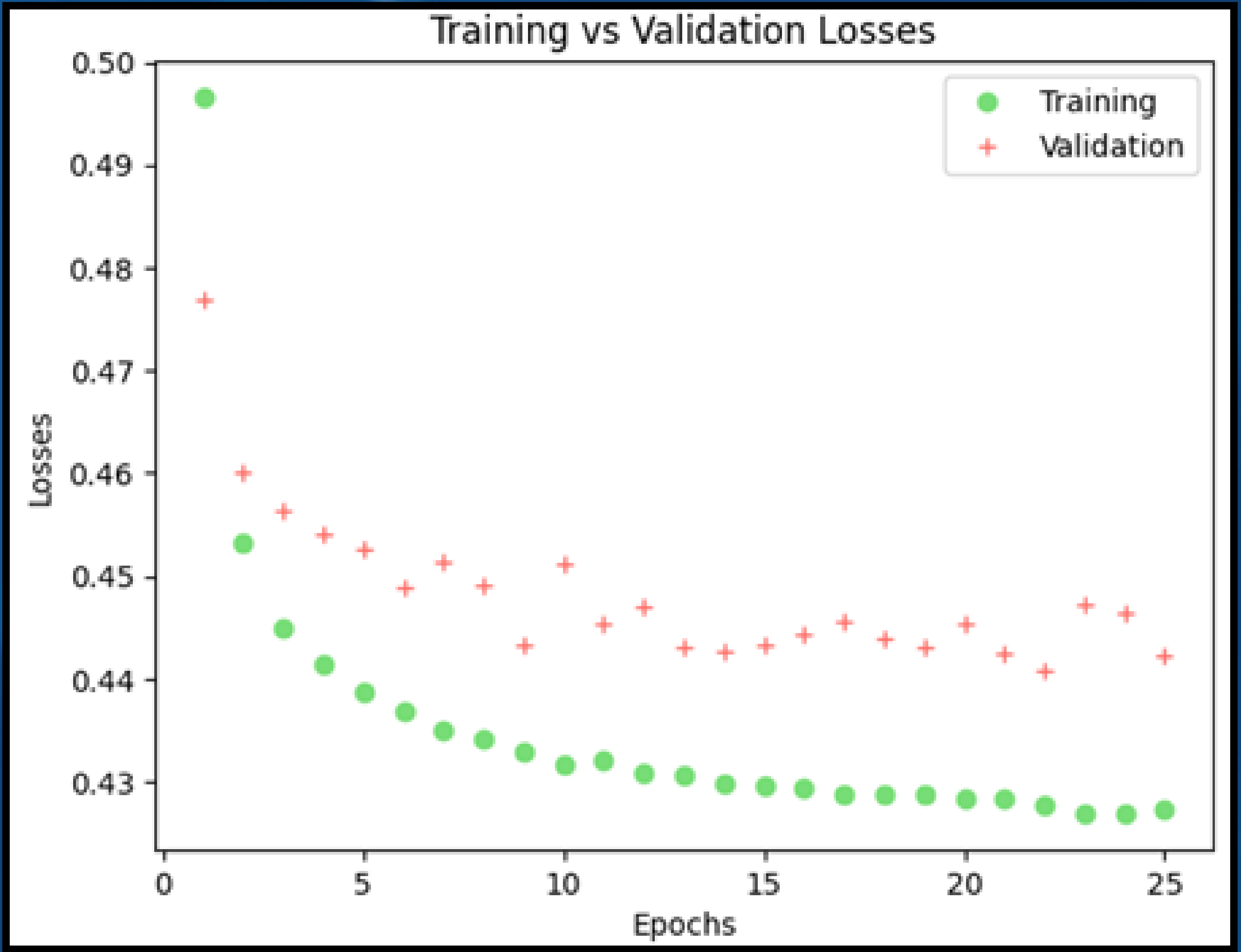
- Visualizing Decision Tree with depth=6



Dense Neural Network

Model: Dense Neural Network (Keras)
Task: Binary classification
Data: 70/30 split, standardized
Architecture: 16-8-1 neurons, sigmoid output
Compilation: Binary cross-entropy loss, RMSprop optimizer
Training: 25 epochs, batch size 32
Evaluation: Accuracy and loss monitored (plots available)
Next Steps: Hyperparameter tuning



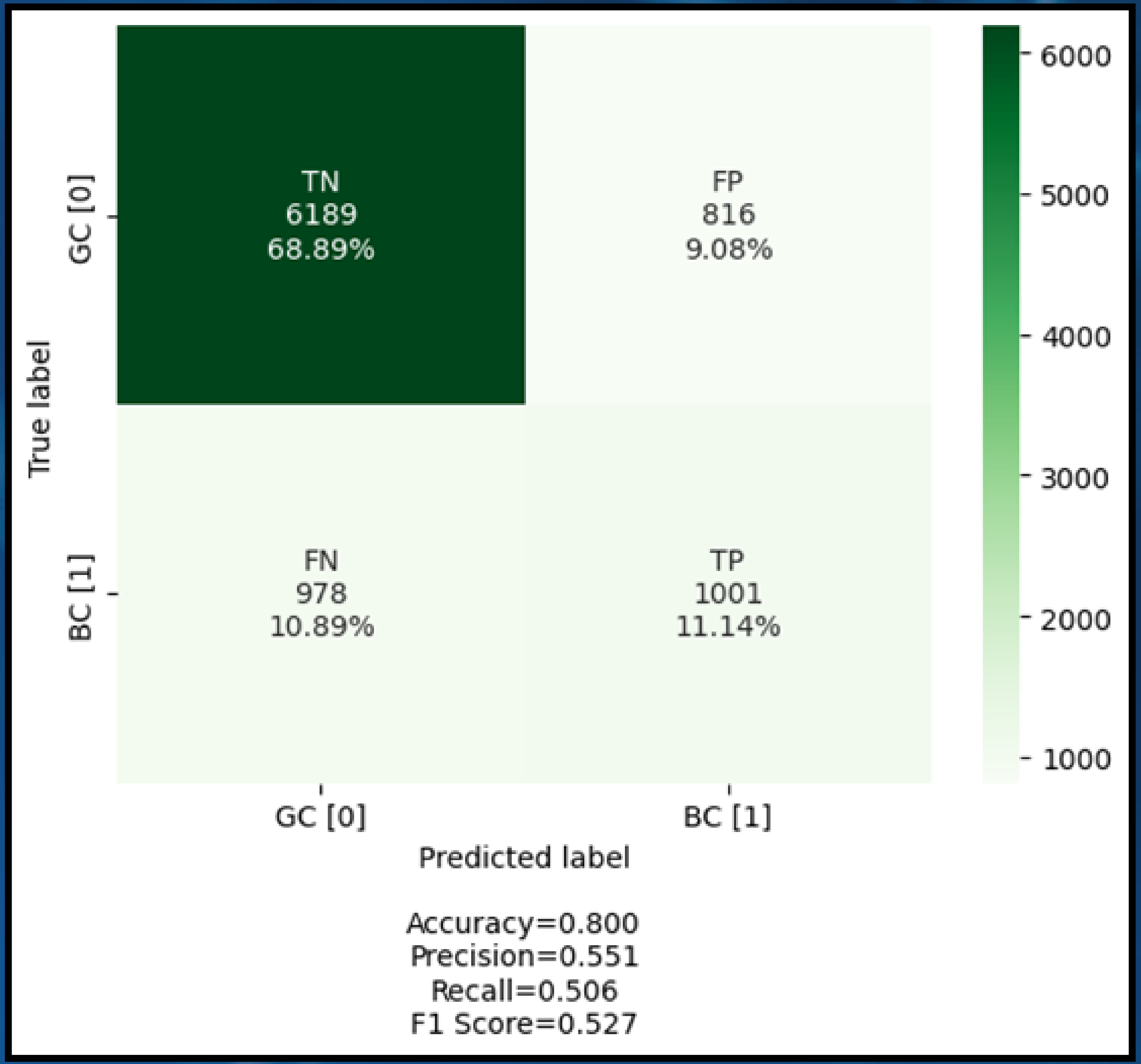


From all 7005 NO's, 6189 NO's are correctly predicted, and 816 NO's are predicted as YES.

From all 1979 YES, 1001 YES are correctly predicted, and 978 YES are predicted as NO.

The confusion matrix shows a solid performance for model accuracy (0.8) but moderate scores for precision (0.55) and recall (0.5).

This shows that while the model in general can identify labels well, the rate of false positives over totality of predicted positives is quite high. That can be a significant model shortcoming in the case of identifying defaulting customers.



	precision	recall	f1-score	support
0	0.86	0.88	0.87	7005
1	0.55	0.51	0.53	1979
accuracy			0.80	8984
macro avg	0.71	0.69	0.70	8984
weighted avg	0.79	0.80	0.80	8984

Hyperparameter Tuning Methods

- **Standardization**
- **Dividing Data: into training and testing sets (70% and 30%, respectively).**

Hyperparameter Tuning - Grid Search

Grid Search exhaustively explores hyperparameter combinations, assessing training/scoring times, hyperparameter values, cross-validation test scores, and ranks based on mean_test_score, making it time-consuming and resource-intensive.

Interpretation: The Grid Search identified a combination of batch_size=16, epochs=30, and unit=8 as the one offering the highest average test score of 0.816764

In [94]:
```
results_df = pd.DataFrame(grid_result.cv_results_)
results_df.head()
```

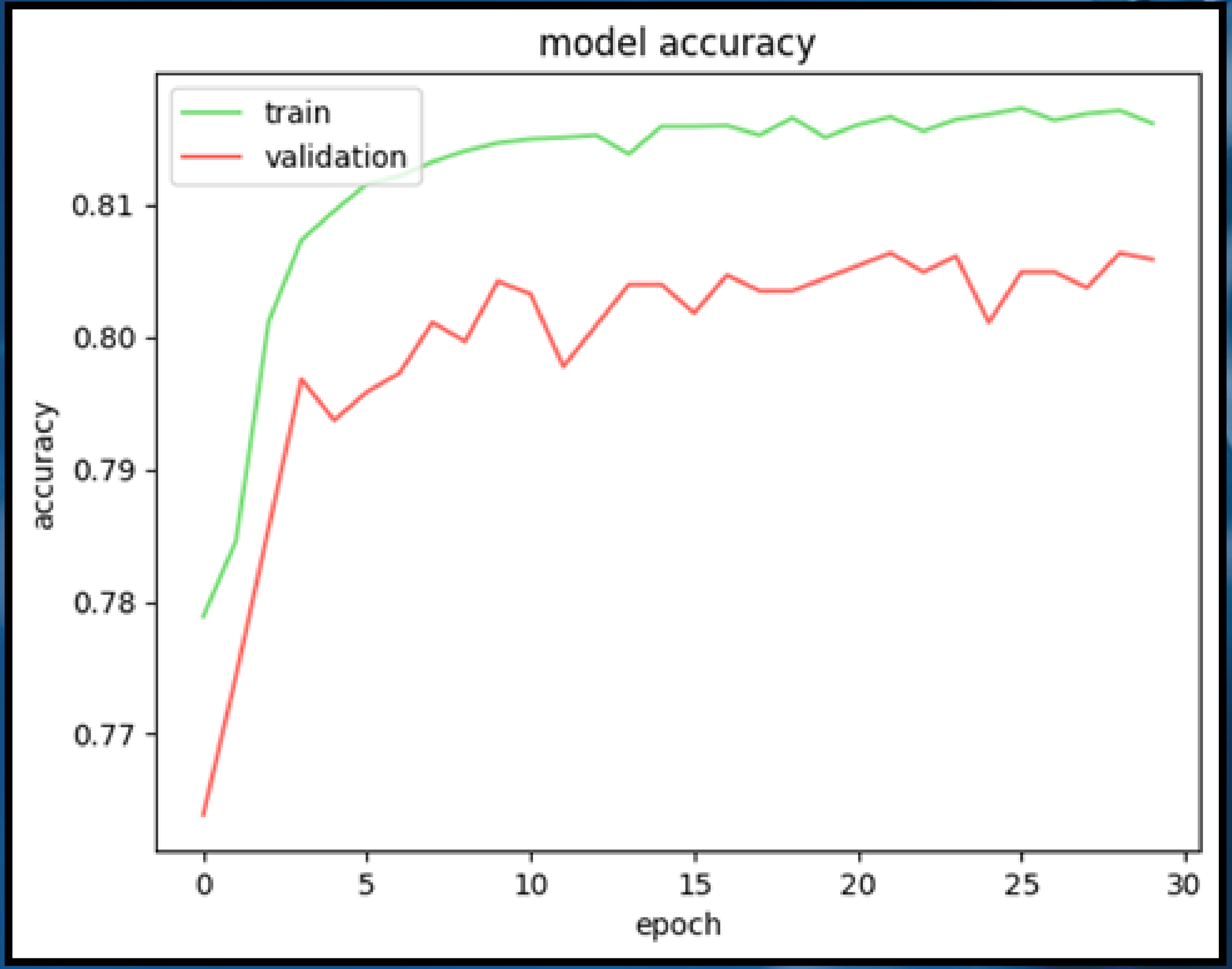
Out[94]:

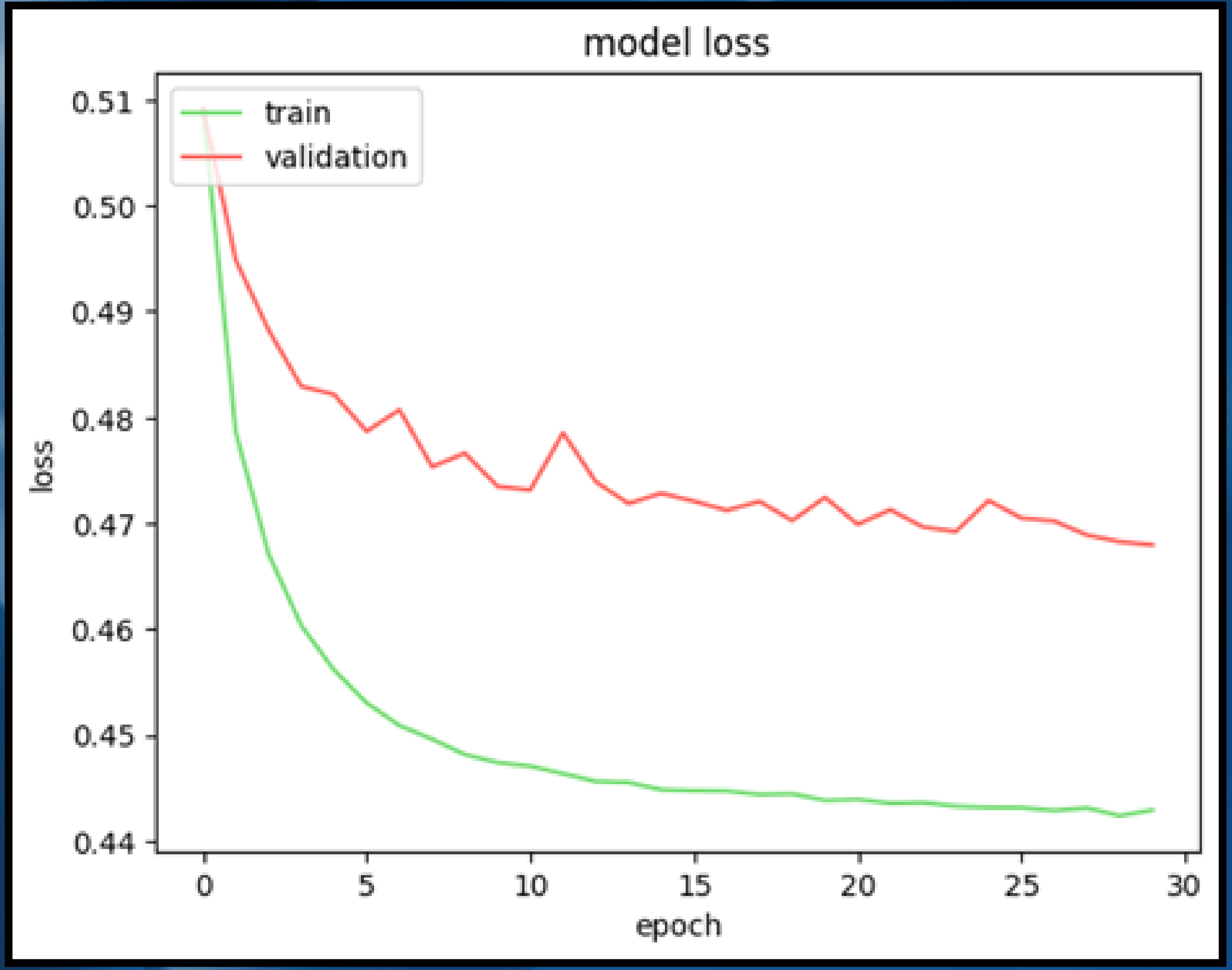
	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_batch_size	param_epochs	param_unit	params	split0_test_score	split1_test_score
0	32.689699	5.447167	0.936721	0.406353	16	15	8	{'batch_size': 16, 'epochs': 15, 'unit': 8}	0.803663	0.812250
1	31.819532	0.690208	0.596594	0.027643	16	15	16	{'batch_size': 16, 'epochs': 15, 'unit': 16}	0.801755	0.809578
2	65.353984	10.343005	0.759008	0.064106	16	30	8	{'batch_size': 16, 'epochs': 30, 'unit': 8}	0.808052	0.814348
3	65.739753	1.166646	0.693189	0.076400	16	30	16	{'batch_size': 16, 'epochs': 30, 'unit': 16}	0.806144	0.807480
4	17.419242	2.730712	0.411837	0.046060	32	15	8	{'batch_size': 32, 'epochs': 15, 'unit': 8}	0.802709	0.811486

Model Building with Identified Hyperparameters

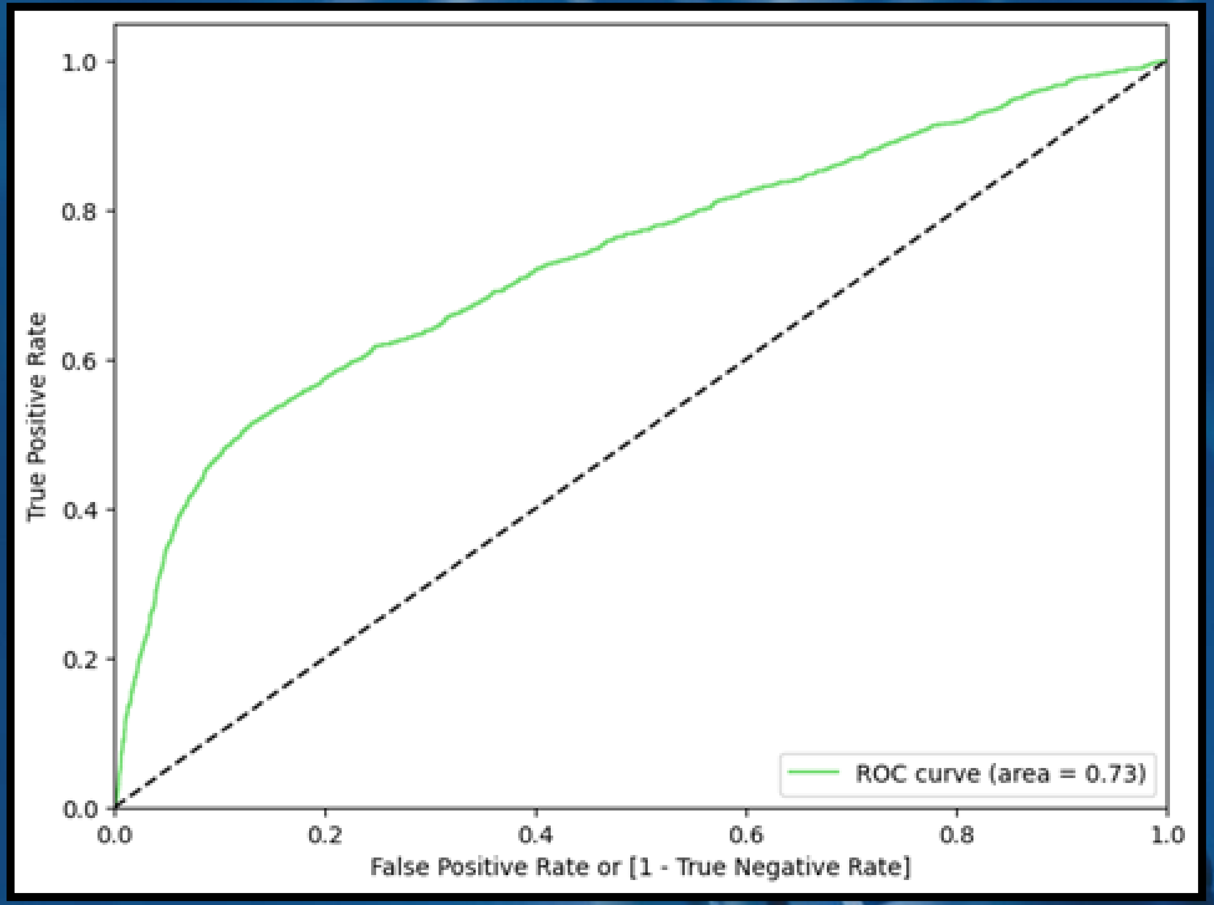
- Steady improvement in training and validation accuracy for initial 10-15 epochs.
- Plateaued performance beyond 15 epochs, indicating peak performance.
- Final training accuracy: 81.61%, validation accuracy peak: 80.63%, suggesting slight overfitting.
- Convergence observed with consistent decrease in training and validation loss.
- Potential for early stopping after around 20 epochs due to minor fluctuations in validation accuracy.

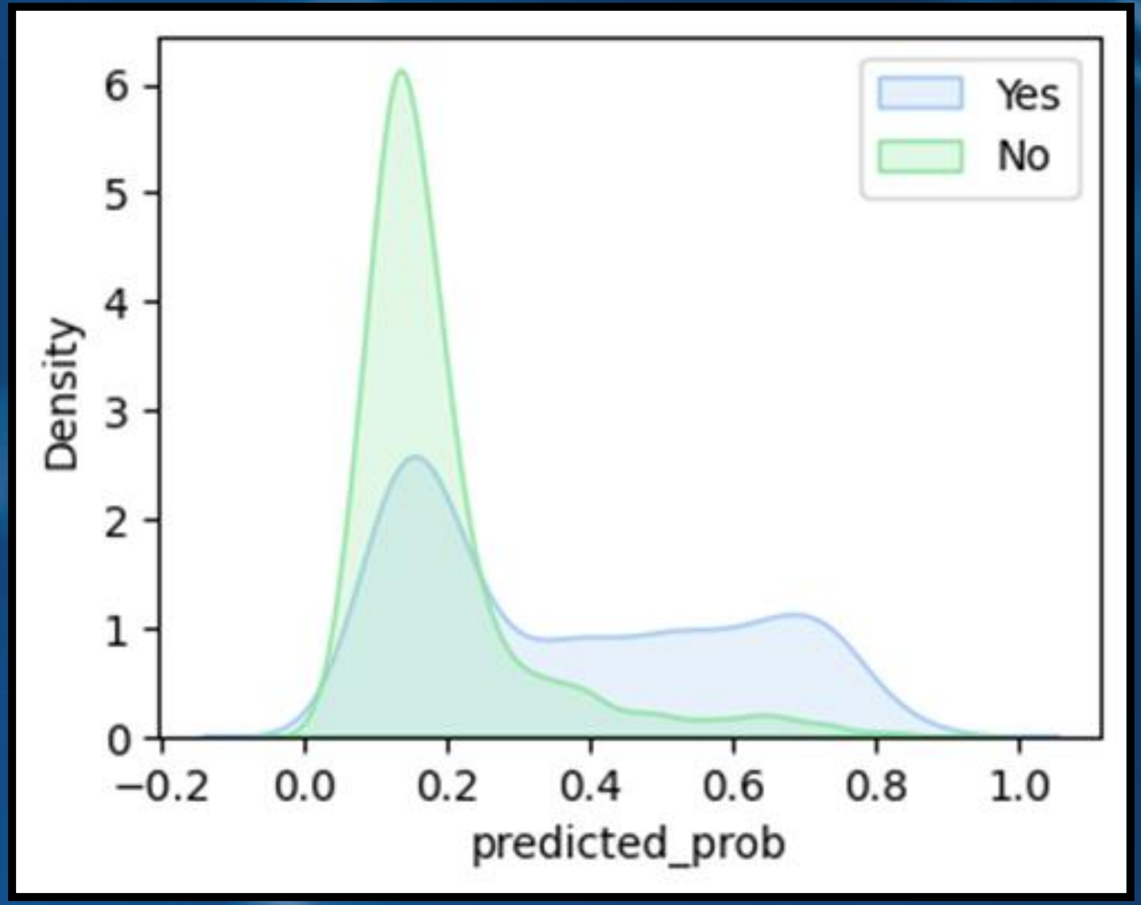
Overall assesment:The model exhibits a reasonable level of accuracy, suggesting effective learning; however, slight overfitting signals may be mitigated through regularization or early stopping, with potential performance enhancement through adjustments to model architecture and hyperparameter tuning.





Model Validation: The AUC score of 0.7333 indicates moderate performance

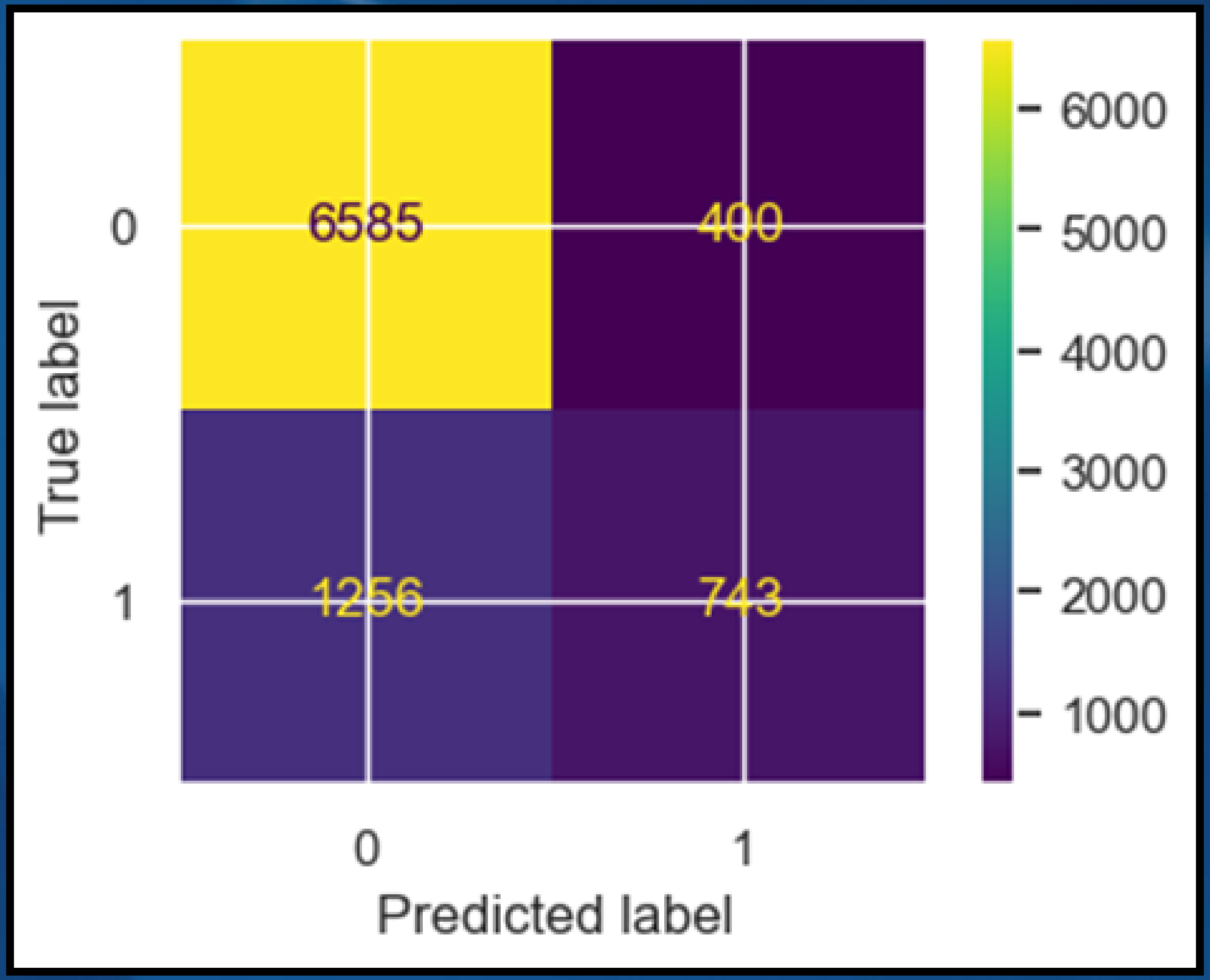




Confusion Matrix:

Kernel density plots depict predicted probability distributions for 'Yes' and 'No' classes, while the confusion matrix indicates:

- **High accuracy (84%) for class 0 with 84% recall and precision.**
- **Lower accuracy (82%) for class 1, marked by 37% recall and 65% precision.**
- **Overall accuracy is 82%, favoring the larger class (0).**



Hyperparameter Tuning - Random Search

Randomized Search CV, employing early stopping, explores various hyperparameter combinations using 4-fold cross-validation. The top-performing configuration, unit=12, epochs=45, batch_size=32, achieved the highest average test score of 0.8158, indicating sensitivity to parameter choices. The best overall score is 0.8191 with hyperparameters {'unit': 8, 'epochs': 30, 'batch_size': 8}. Iterating through combinations, the loop prints scores and corresponding hyperparameters.

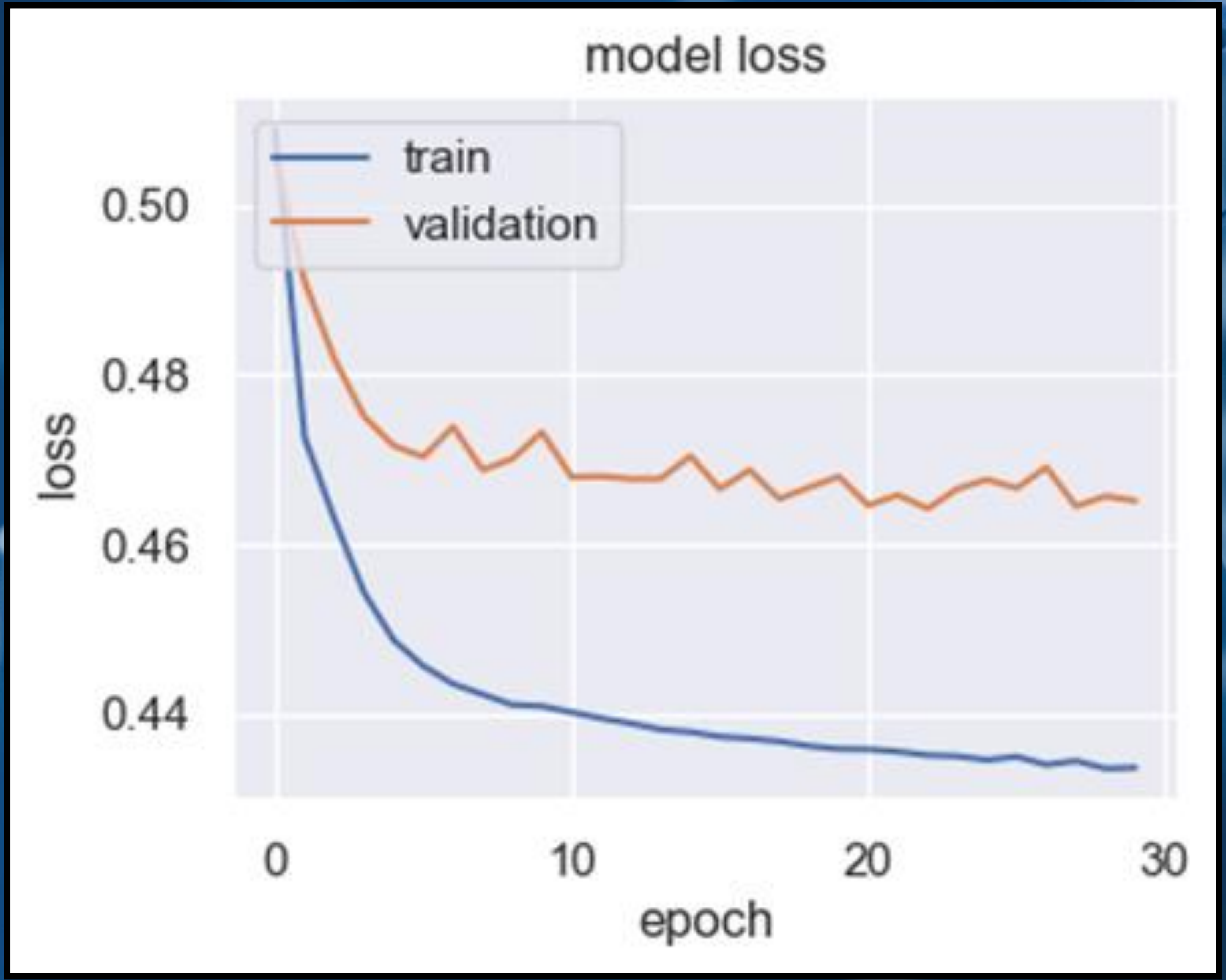
In [113]:
```
results_df = pd.DataFrame(rand_result.cv_results_)
results_df.head()
```

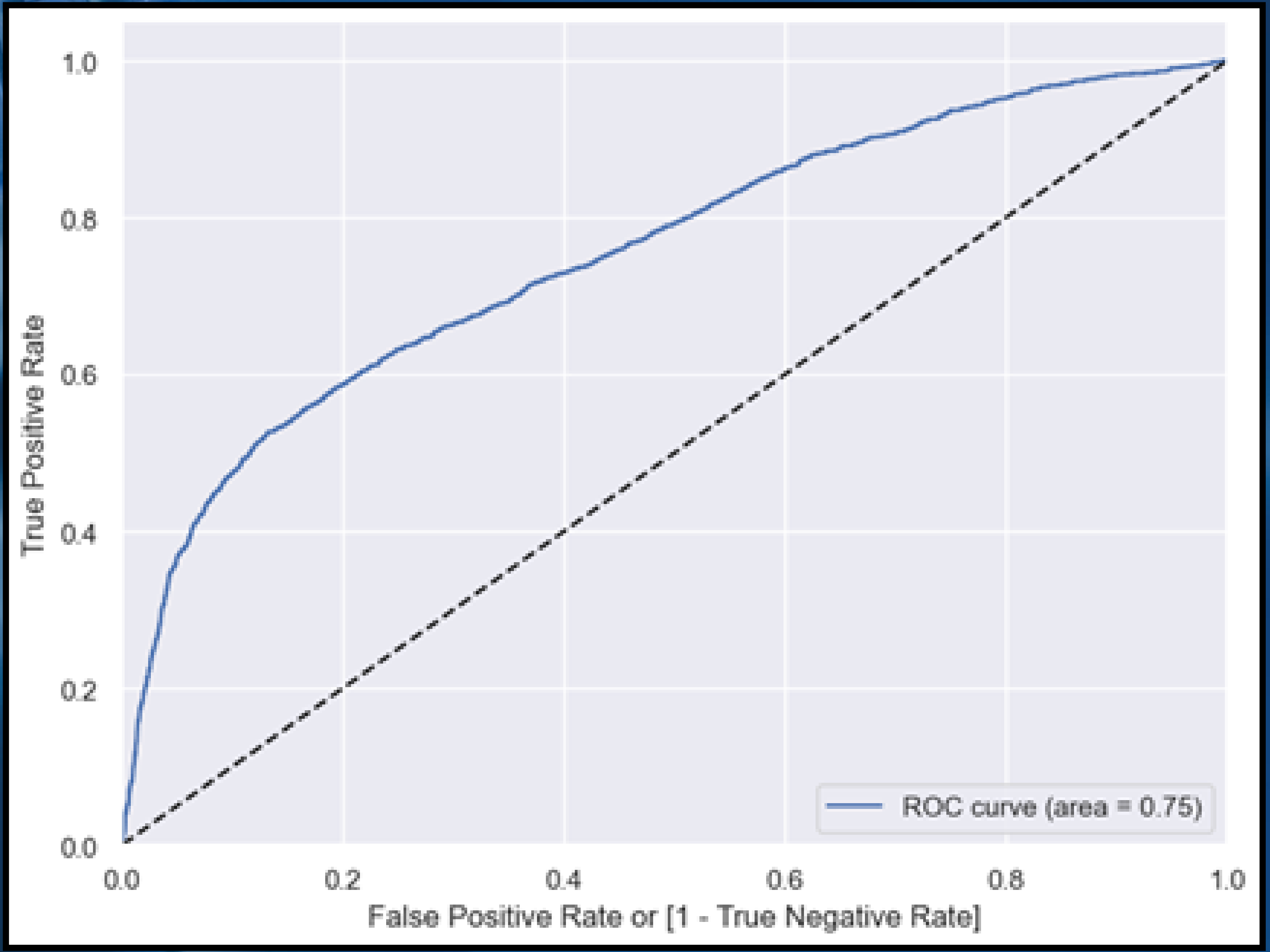
Out[113]:

	mean_fit_time	std_fit_time	mean_score_time	std_score_time	param_unit	param_epochs	param_batch_size	params	split0_test_score	split1_test_score
0	57.224805	14.724238	0.455118	0.064720	12	45	32	{'unit': 12, 'epochs': 45, 'batch_size': 32}	0.807861	0.812822
1	17.479127	0.263221	0.486095	0.045419	16	15	32	{'unit': 16, 'epochs': 15, 'batch_size': 32}	0.801946	0.809006
2	59.203795	0.396668	0.678154	0.045849	12	30	16	{'unit': 12, 'epochs': 30, 'batch_size': 16}	0.807670	0.816256
3	175.564897	17.797046	1.498867	0.408284	8	45	8	{'unit': 8, 'epochs': 45, 'batch_size': 8}	0.811296	0.817401
4	124.107967	2.894311	1.339456	0.132959	16	30	8	{'unit': 16, 'epochs': 30, 'batch_size': 8}	0.805190	0.821217

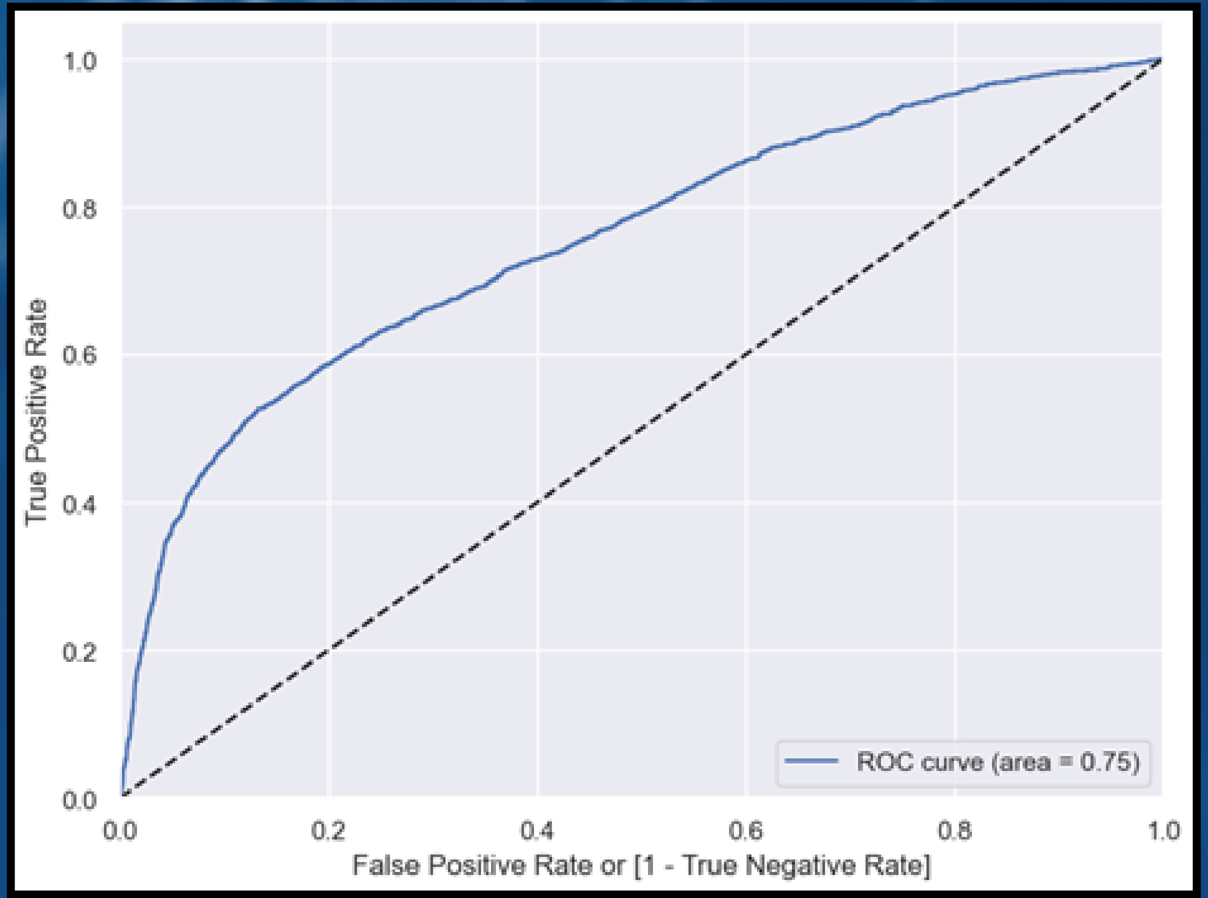
Model Building with Identified Hyperparameters

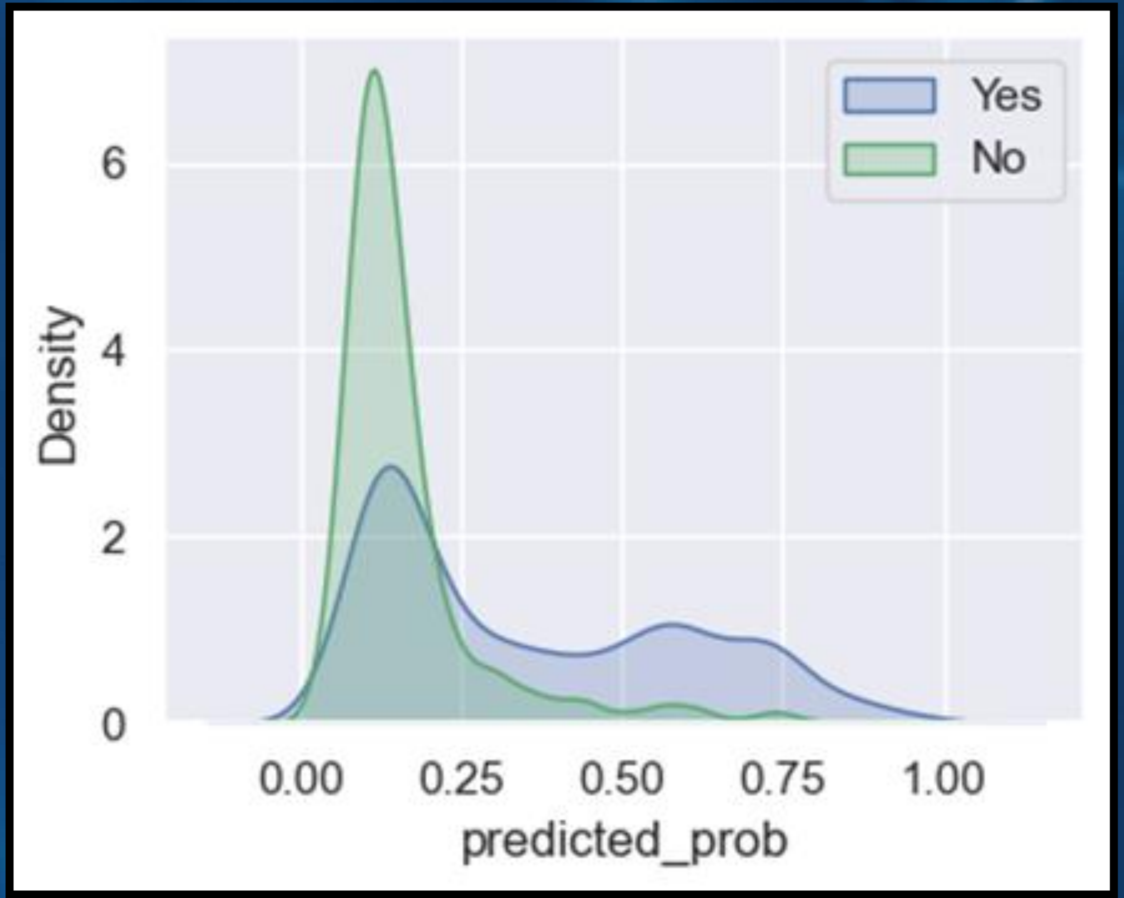
- Initial effective learning plateaued after 15-20 epochs.
- Final accuracies: Training 82.23%, validation 80.80%.
- Slight overfitting indicated by consistently higher training accuracy.
- Convergence observed in decreasing training and validation loss.
- Potential for early stopping around 20 epochs to prevent overfitting and save time.

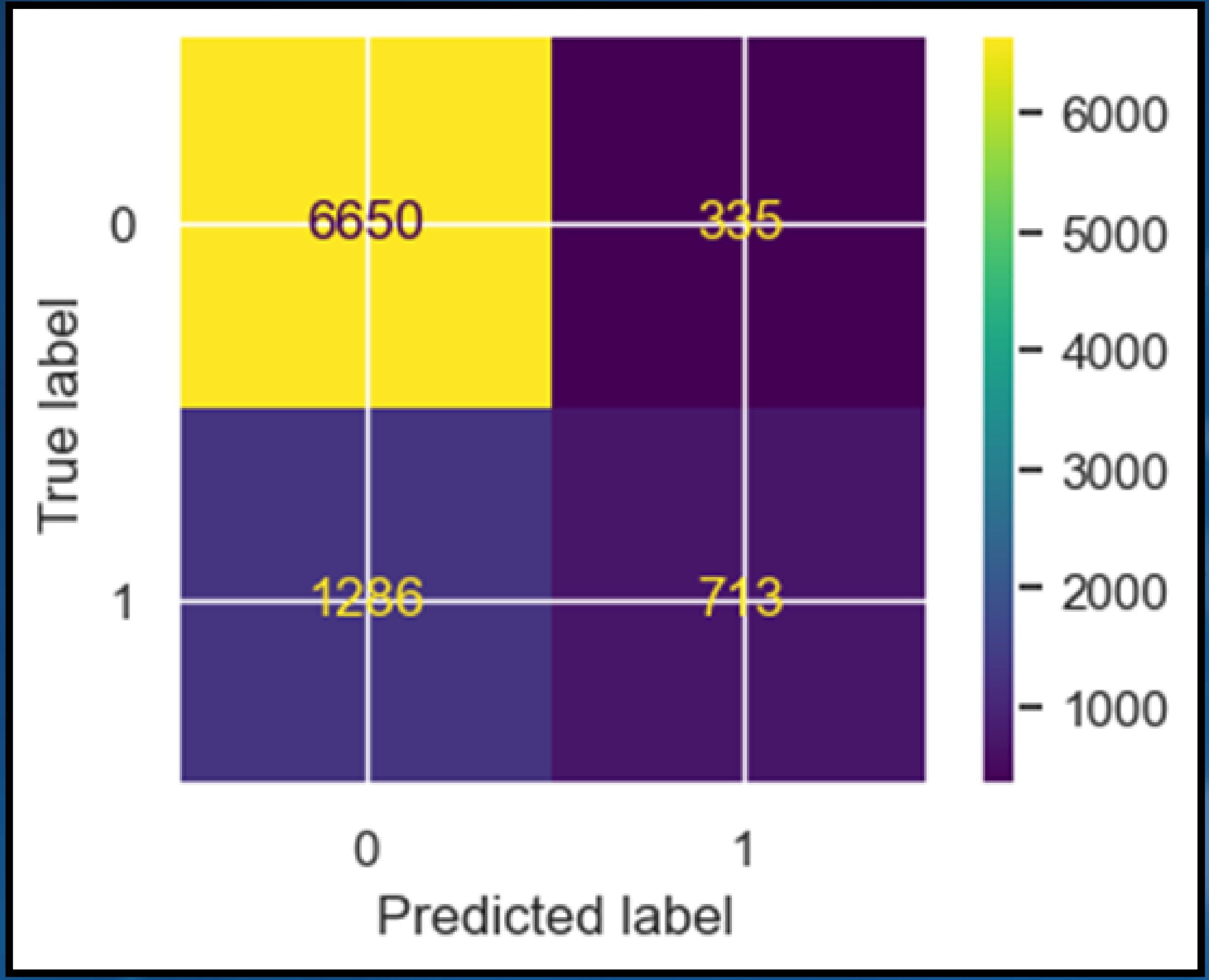




AUC score of 0.7547 indicates moderate performance, signifying the model's reasonable ability to distinguish between positive and negative cases.







Performance Measure: High accuracy for Class 0 (precision 0.84, recall 0.95), lower accuracy for Class 1 (precision 0.68, recall 0.36). Overall accuracy is 0.82, and macro/weighted averages suggest moderate performance across both classes

Interpretation: Model excels with majority class (0) but struggles with class 1, as shown by lower recall and precision. AUC score supports moderate class discrimination. Overall accuracy may be acceptable, but class performance disparity is crucial to consider in context.

Hyperparameter Tuning - Bayesian Optimization

Bayesian optimization tunes model architecture with units (8-64) and activation (sigmoid or relu) for highest validation accuracy. Executed for 5 trials, the best accuracy achieved is 0.8247 in 6 minutes 58 seconds, with the 5th trial scoring 0.8163.

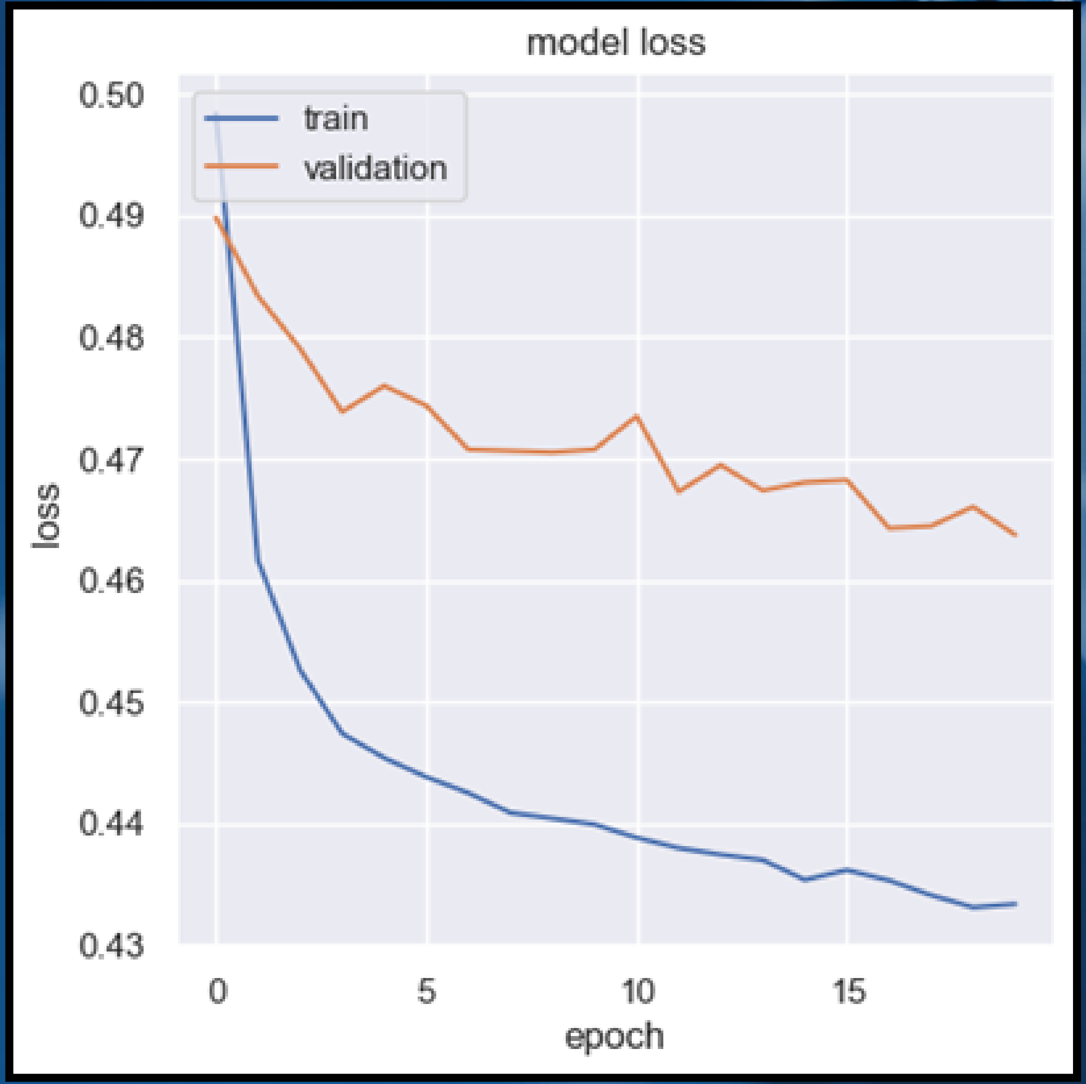
Interpretation: Bayesian optimization, in 5 trials, found the best hyperparameter configuration with an accuracy of 0.8247 on the validation set, indicating a successful identification of a promising combination for the model.

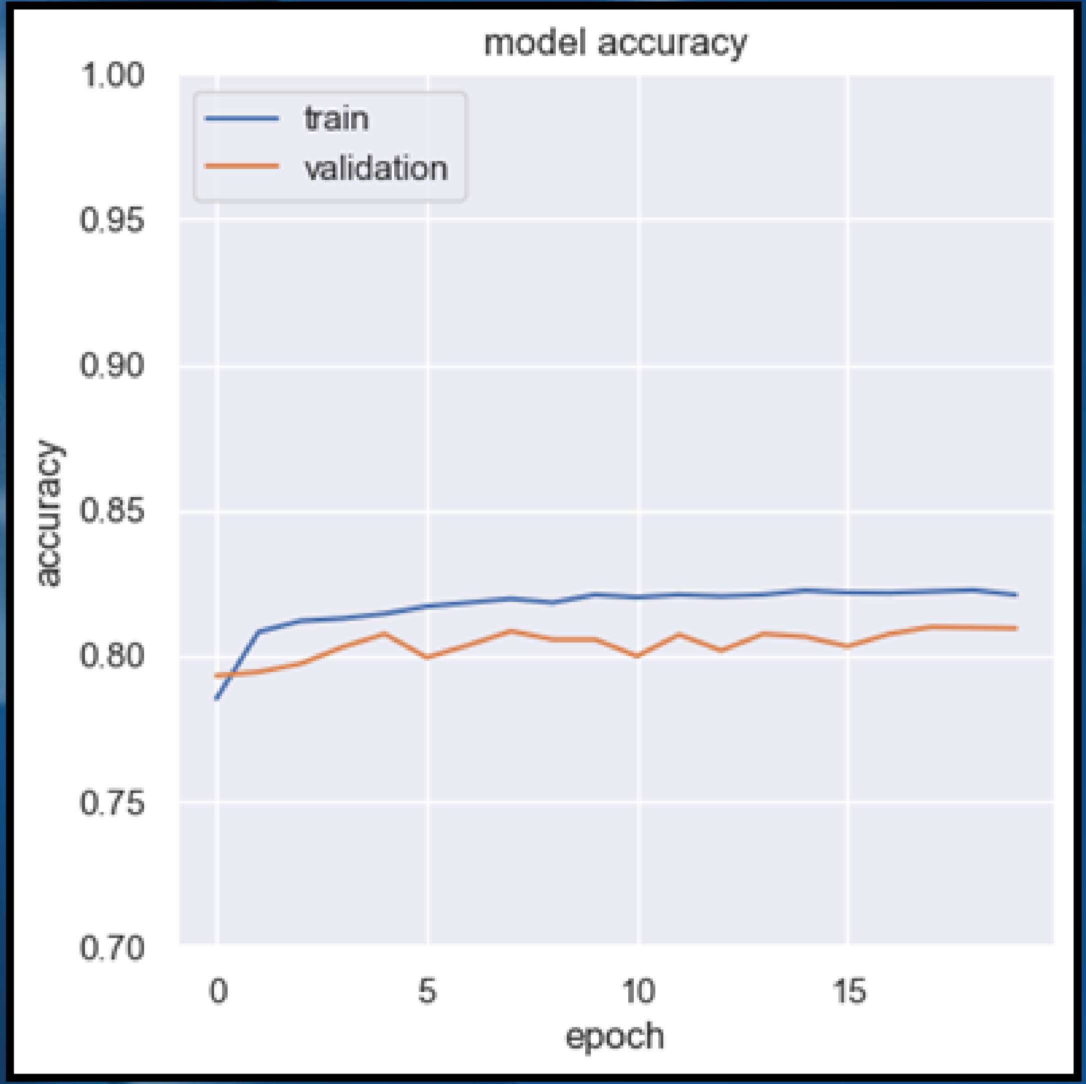
STEP 1.2.4: Obtaining Best HyperParameters

```
In [136]: best_hp = bayesian_tuner.get_best_hyperparameters(num_trials = 1)[0]

In [137]: best_hp.get('units')

Out[137]: 56

In [138]: best_hp.get('activation')

Out[138]: 'relu'
```

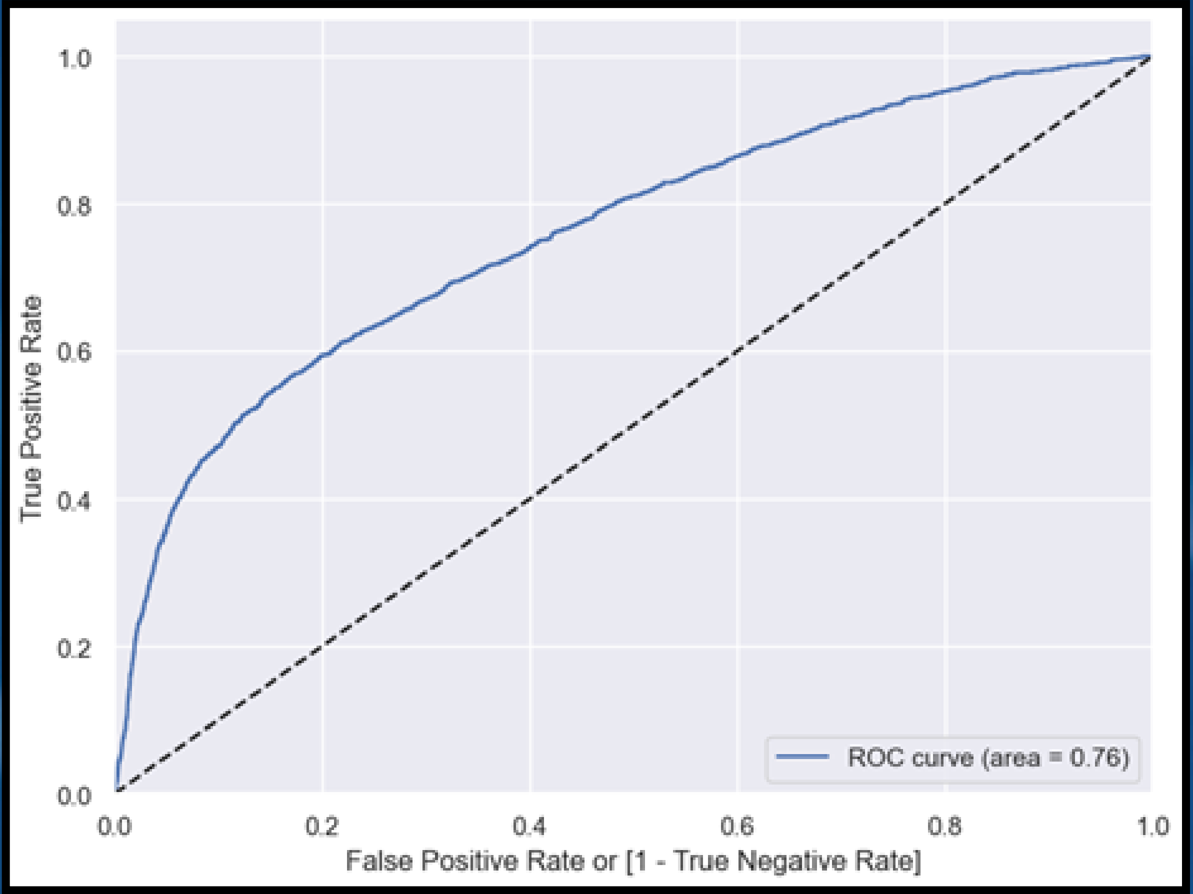
Model Building with Identified Hyperparameters

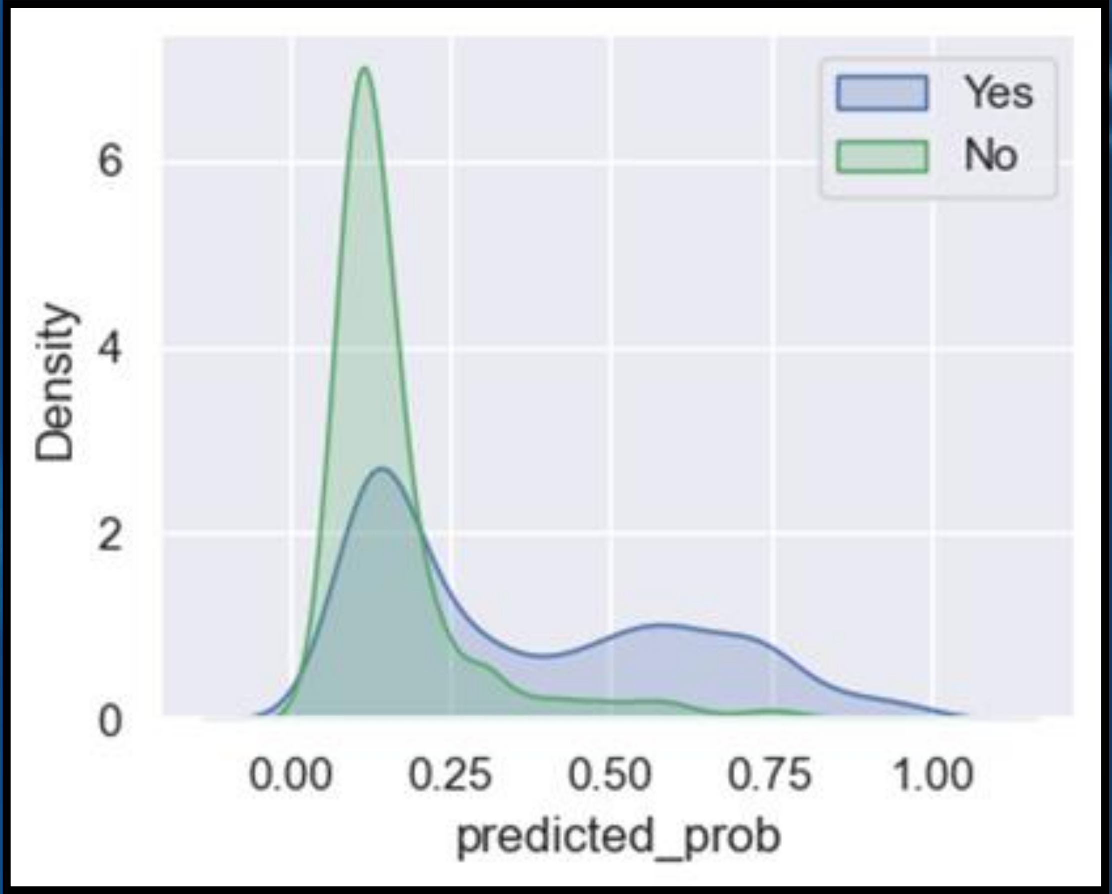
- Learning and Convergence: Clear patterns observed in both training and validation accuracy and loss over 20 epochs.
- Final Accuracies: Training accuracy of 82.10% and validation accuracy of 80.94% indicate reasonable generalization.
- Minimal Overfitting: Slight gap between training and validation accuracy suggests minimal overfitting.
- Dropout Effectiveness: Inclusion of a 5% dropout layer appears effective in mitigating overfitting, evident in the close alignment of training and validation accuracy curves.

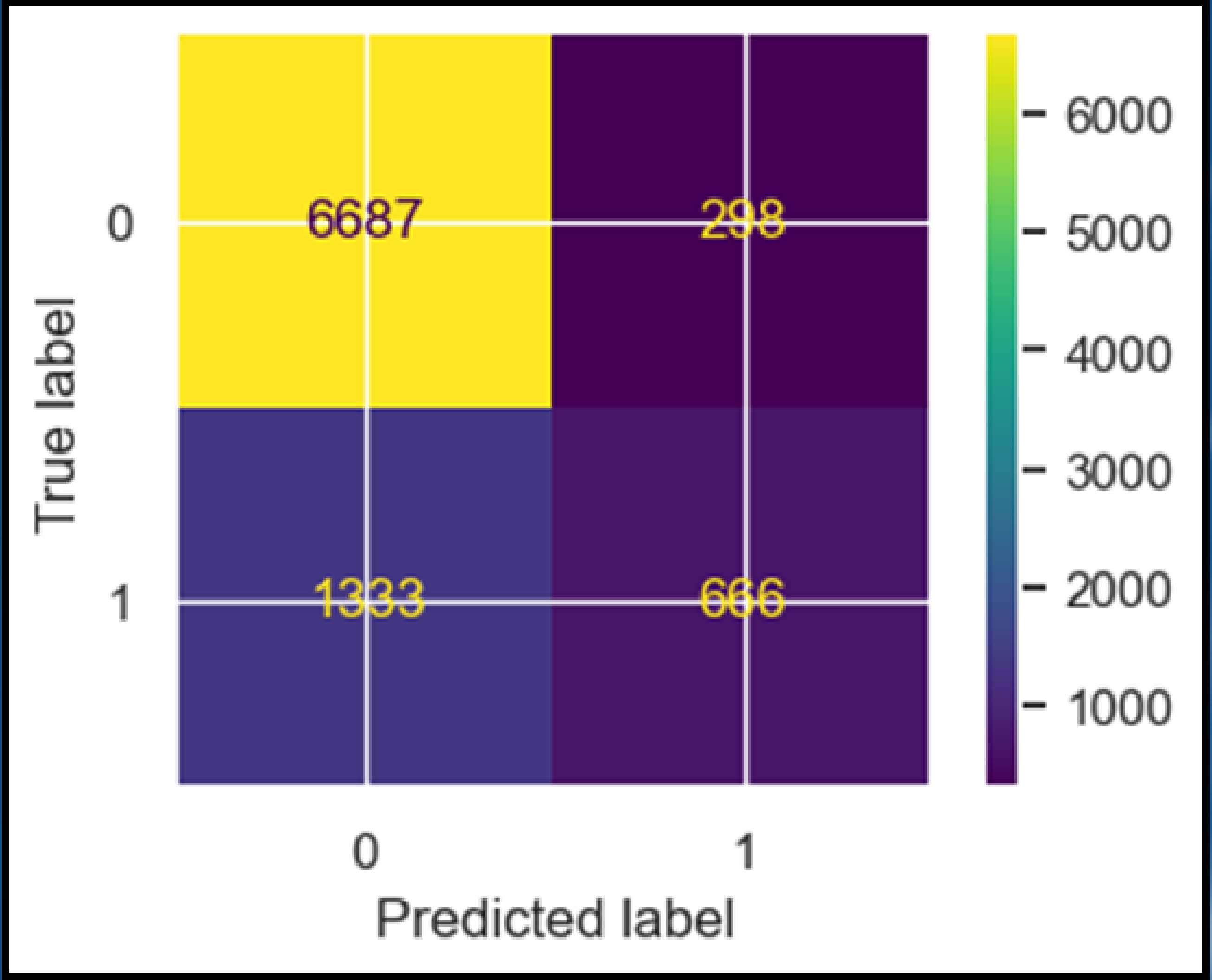




AUC score improved slightly from 0.7547 to 0.7595, indicating a marginal increase in the model's discrimination ability between positive and negative cases.







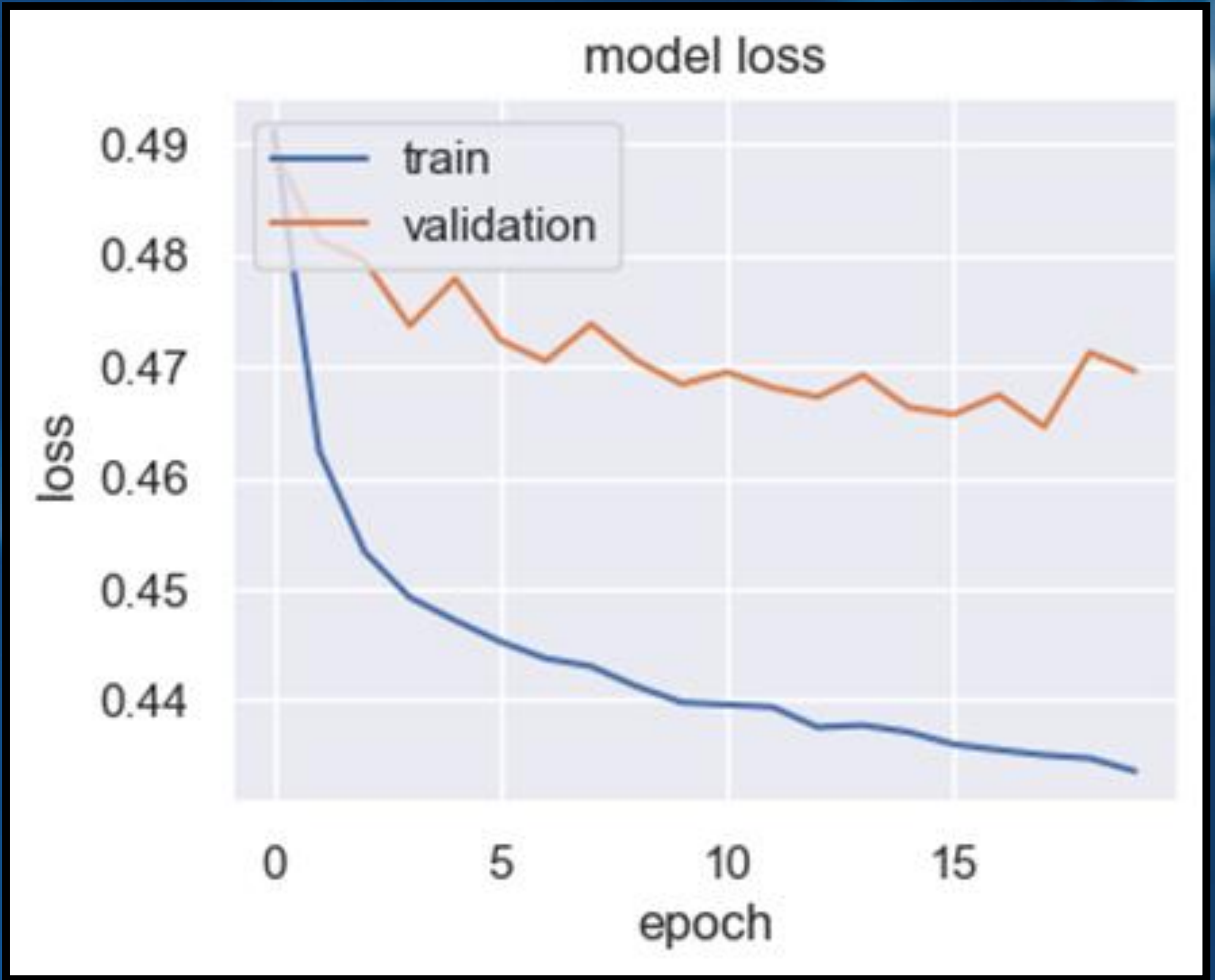
- Class 0 performance remains high, while Class 1 recall is still lower at 33%.
- Overall accuracy stays around 82%, similar to the previous analysis.
- Precision and F1-score for Class 1 slightly lower, and the recall gap between classes widened.

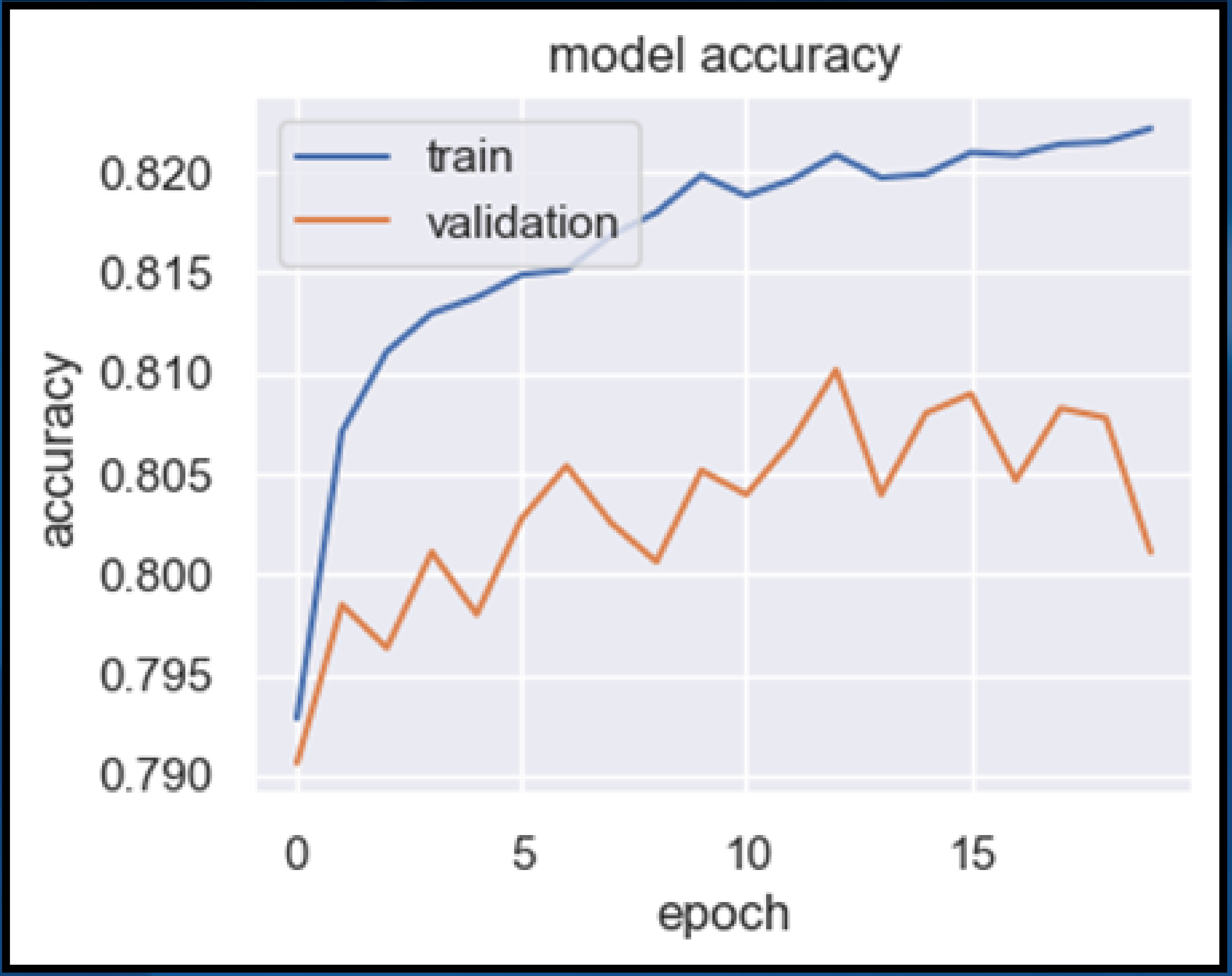
Interpretation: Bayesian optimization led to a minor AUC improvement, but Class 1 performance regressed, indicating suboptimal hyperparameters for addressing class imbalance or specific characteristics of Class 1 samples.

Hyperparameter Tuning - Hyperband Optimisation

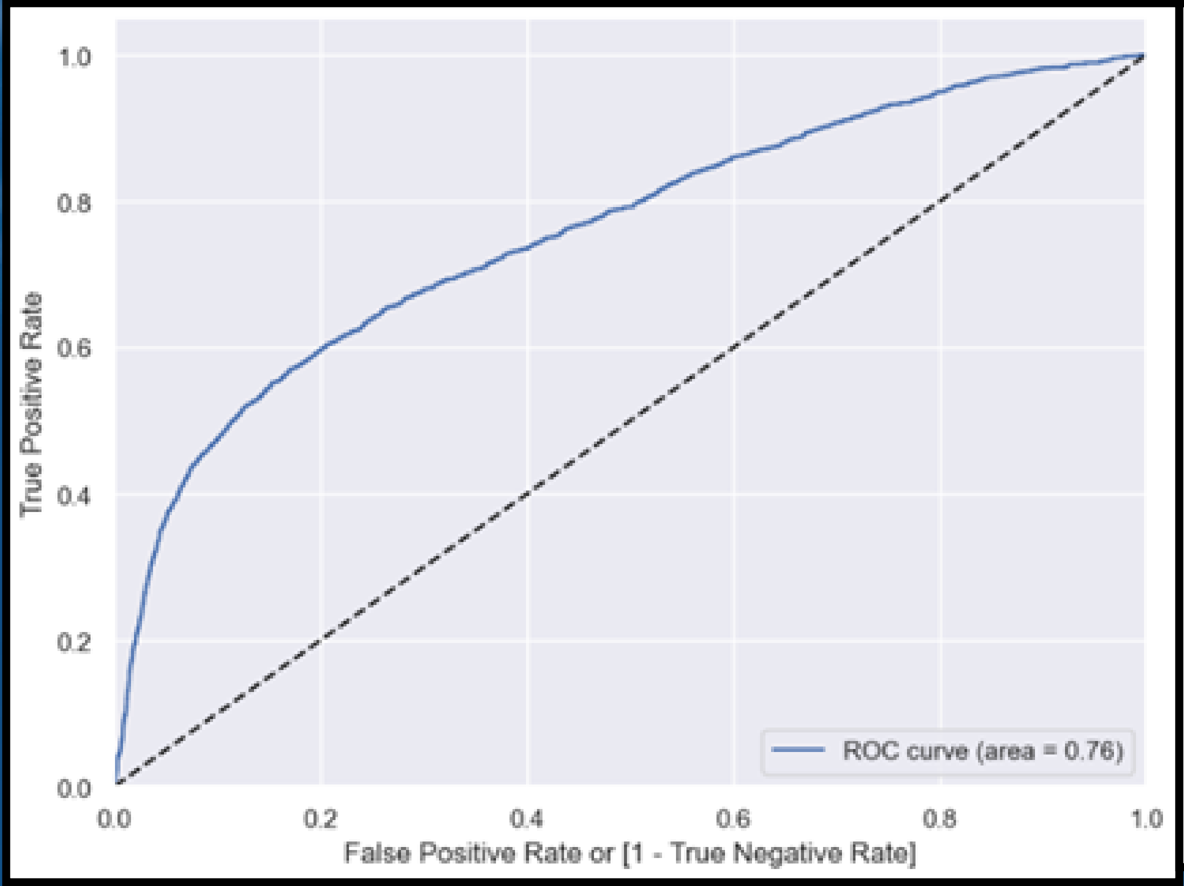
Hyperband tuner maximizes accuracy with parameters like max_epochs (300), hyperband_iterations, and a reduction factor (10x per round). Early stopping prevents overfitting, with the 16th trial achieving 0.7857 accuracy, and the current best is 0.8125. Top hyperparameters are 56 neurons in the first layer with relu activation.

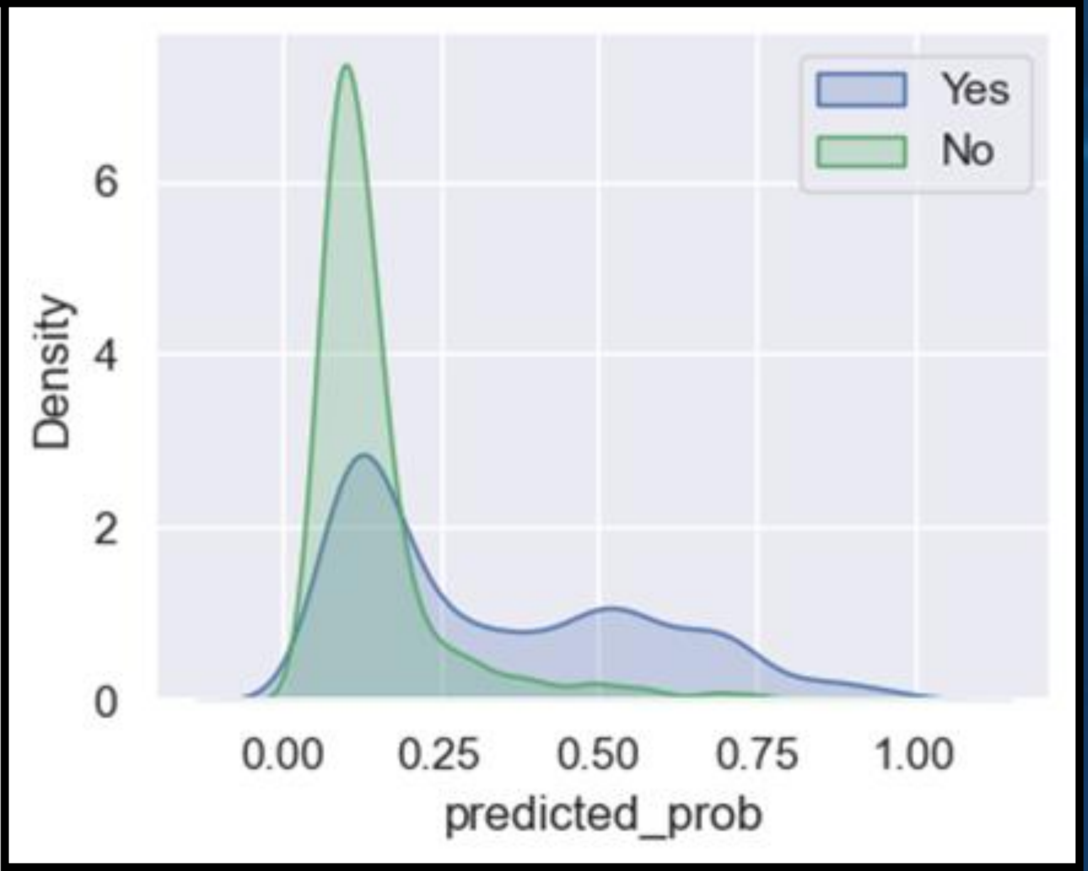
- Learning and Convergence: Clear patterns observed in both training and validation accuracy and loss over 20 epochs.
- Final Accuracies: Training accuracy 82.22%, validation accuracy 80.11%, indicating moderate generalization.
- Overfitting: Slightly larger gap between training and validation accuracy suggests potential overfitting.
- Dropout Effectiveness: 5% dropout mitigated overfitting, but exploring higher dropout rates or additional regularization techniques is worth considering.

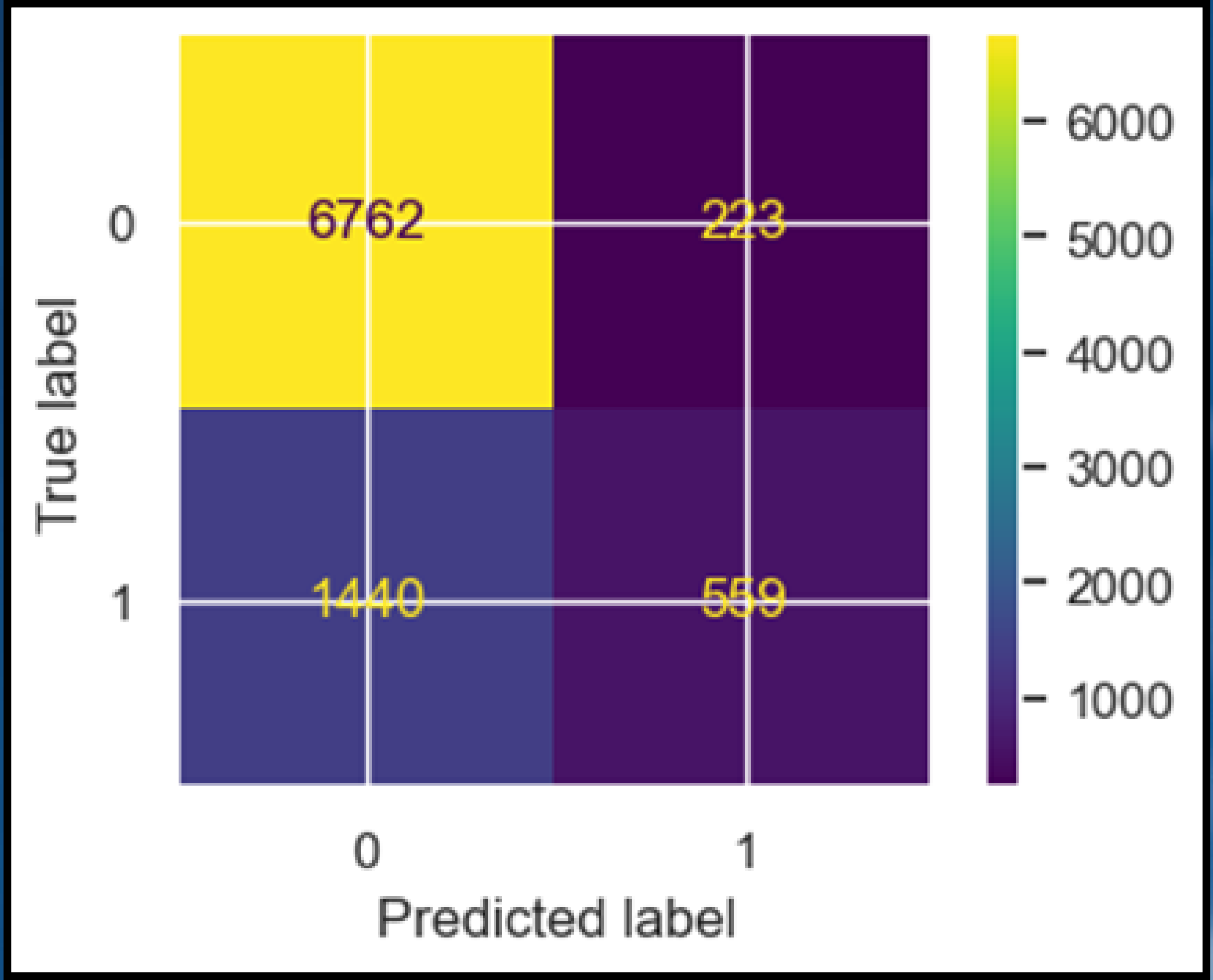




- Overall Accuracy: 81%, consistent with previous analyses.
- Class 0: High precision (0.82) and recall (0.97).
- Class 1: Lower precision (0.71) and recall (0.28).
- AUC Score: Slightly decreased to 0.7572, suggesting a marginal drop in class discrimination ability







Confusion matrix confirms class imbalance and difficulty in identifying true positives; kernel density plots reveal overlap in predicted probability distributions, emphasizing the challenge of distinguishing both classes.

Interpretation:

Hyperband identified a configuration with similar overall accuracy, but failed to improve performance on the minority class (1), evident in low recall. The overlap in predicted probability distributions and the slight decrease in AUC and F1-scores suggest suboptimal hyperparameters for addressing class imbalance or specific characteristics of class 1 samples.

Conclusion

Predictive Modeling for Credit Card Defaults:

- Problem: Predicting client defaults to improve risk management and customer satisfaction.
- Models: Promising results with Logistic Regression, Decision Trees, and DNNs.
- Key Metrics: Prioritize recall (avoiding missed defaults) along with accuracy.

Key Variables:

- Primary: Income, credit history, debt, utilization rate, and financial indicators.
- Boosting Power: Credit scores, work history, and demographic data.

Model Selection:

- Choice: Depends on goals, resources, and interpretability.
- Strong Candidates: Logistic Regression (AUC, threshold adjustment) and Decision Trees (good performance, interpretability).
- Exploration: Deep Neural Networks for potential improvements.

Neural Network Optimization:

- Challenges: DNN sensitivity to hyperparameters, imbalanced data.
- Tuning Methods: Grid Search, Random Search, Bayesian Optimization, and Hyperband Optimization.
- Trade-offs: Optimizing for overall accuracy vs. identifying true defaults (recall).

Business Implications:

- Reduced Losses: Informed decisions on approvals, credit limits, and risk assessments.
- Personalized Offerings: Tailored credit options for specific customers.
- Model Maintenance: Regular updates with fresh data due to changing behavior and economic conditions.

THANK YOU!