Executive Summary

In the submitted model, a model was built to predict whether a customer would take a product or not. I included many data preparation steps across the pipeline, including handling typo, dropping missing values, encoding categorical features to become numerical, and applying StandardScaler to normalize numerical columns to prevent any single feature from dominating due to having a bigger number. I also addressed the imbalance of ProdTaken through manual oversampling and used a Wide and Deep neural network along with Dropout, L2, and BatchNormalization, trained with EarlyStopping and ReduceLROnPlateau callbacks.

A new feature such as log transformations is also created to handle outlier, and interaction terms like Income_Pitch_Score and Age_Income, polynomial squared terms, and age group binning to help the model capture non direct relationships. I also have explored different loss functions, but decided to use Binary Crossentrop after seeing that they give the best results. The model was also trained on an 85/15 stratified split, designed to generalize unseen data to prevent overfitting and class bias.

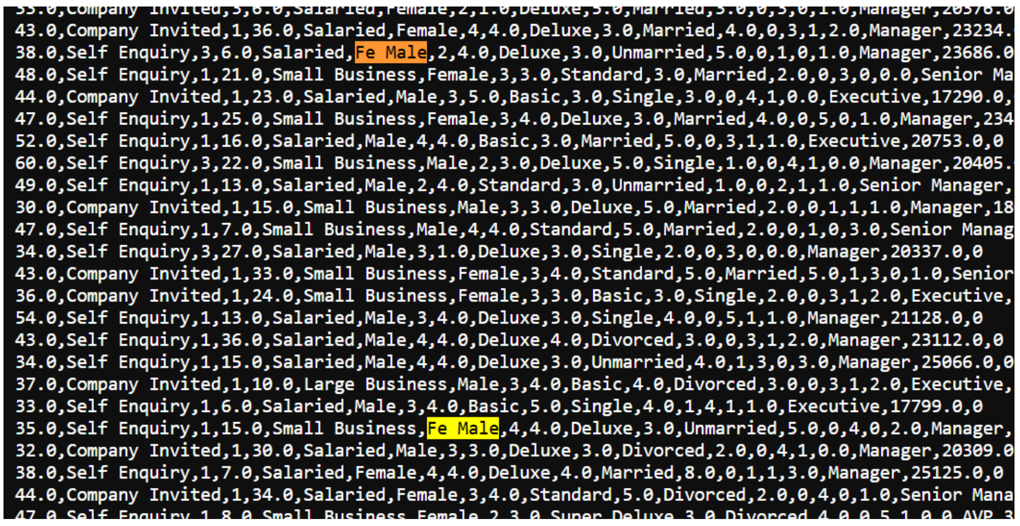
Preparing and Collecting Data

This was already done for us. So I just downloaded the data and put it in the code running platform.

Data Analysis & Data Preparation

Analysis part

There are two types of data in the dataset which are "Numerical and Ordinal". And there are also some data that has the wrong format. Such as Female being written as Fe male. So I change those wrongly written Fe Male to become Female.



```
33.0,Company Invited,3,6.0,Salaried,Female,2,1.0,Deluxe,3.0,Married,3.0,0,3,0,1.0,Manager,20378.0
43.0,Company Invited,1,36.0,Salaried,Female,4,4.0,Deluxe,3.0,Married,4.0,0,3,1,2.0,Manager,23234.
38.0,Self Enquiry,3,6.0,Salaried,Fe Male,2,4.0,Deluxe,3.0,Unmarried,5.0,0,1,0,1.0,Manager,23686.0
48.0,Self Enquiry,1,21.0,Small Business,Female,3,3.0,Standard,3.0,Married,2.0,0,3,0,0.0,Senior Ma
44.0,Company Invited,1,23.0,Salaried,Male,3,5.0,Basic,3.0,Single,3.0,0,4,1,0.0,Executive,17290.0,
47.0,Self Enquiry,1,25.0,Small Business,Female,3,4.0,Deluxe,3.0,Married,4.0,0,5,0,1.0,Manager,234
52.0,Self Enquiry,1,16.0,Salaried,Male,4,4.0,Basic,3.0,Married,5.0,0,3,1,1.0,Executive,20753.0,0
60.0,Self Enquiry,3,22.0,Small Business,Male,2,3.0,Deluxe,5.0,Single,1.0,0,4,1,0.0,Manager,20405.
49.0,Self Enquiry,1,13.0,Salaried,Male,2,4.0,Standard,3.0,Unmarried,1.0,0,2,1,1.0,Senior Manager,
30.0,Company Invited,1,15.0,Small Business,Male,3,3.0,Deluxe,5.0,Married,2.0,0,1,1,1.0,Manager,18
47.0,Self Enquiry,1,7.0,Small Business,Male,4,4.0,Standard,5.0,Married,2.0,0,1,0,3.0,Senior Manag
34.0,Self Enquiry,3,27.0,Salaried,Male,3,1.0,Deluxe,3.0,Single,2.0,0,3,0,0.0,Manager,20337.0,0
43.0,Company Invited,1,33.0,Small Business,Female,3,4.0,Standard,5.0,Married,5.0,1,3,0,1.0,Senior
36.0,Company Invited,1,24.0,Small Business,Female,3,3.0,Basic,3.0,Single,2.0,0,3,1,2.0,Executive,
54.0,Self Enquiry,1,13.0,Salaried,Male,3,4.0,Deluxe,3.0,Single,4.0,0,5,1,1.0,Manager,21128.0,0
43.0,Self Enquiry,1,36.0,Salaried,Male,4,4.0,Deluxe,4.0,Divorced,3.0,0,3,1,2.0,Manager,23112.0,0
34.0,Self Enquiry,1,15.0,Salaried,Male,4,4.0,Deluxe,3.0,Unmarried,4.0,1,3,0,3.0,Manager,25066.0,0
37.0,Company Invited,1,10.0,Large Business,Male,3,4.0,Basic,4.0,Divorced,3.0,0,3,1,2.0,Executive,
33.0,Self Enquiry,1,6.0,Salaried,Male,3,4.0,Basic,5.0,Single,4.0,1,4,1,1.0,Executive,17799.0,0
35.0,Self Enquiry,1,15.0,Small Business,Fe Male,4,4.0,Deluxe,3.0,Unmarried,5.0,0,4,0,2.0,Manager,
32.0,Company Invited,1,30.0,Salaried,Male,3,3.0,Deluxe,3.0,Divorced,2.0,0,4,1,0.0,Manager,20309.0
38.0,Self Enquiry,1,7.0,Salaried,Female,4,4.0,Deluxe,4.0,Married,8.0,0,1,1,3.0,Manager,25125.0,0
44.0,Company Invited,1,34.0,Salaried,Female,3,4.0,Standard,5.0,Divorced,2.0,0,4,0,1.0,Senior Mana
47.0,Self Enquiry,1,8.0,Small Business,Female,2,3.0,Super Deluxe,3.0,Divorced,4.0,0,5,1,0.0,AVP,3
```

Class Imbalance: Initial analysis showed that successful product takes were minority cases. This led to the decision to use oversampling and threshold optimization later on in the training code.

Feature Skewness: Some numerical columns like MonthlyIncome and DurationOfPitch had wide ranges or skewed distributions compared to other categories. This can mean that using that raw data might be less effective than using an augmented one.

Column Name	Data Type	Data Type	Range or Potential Values
Age	Integer	Numerical	18-60
TypeofContact	String	Nominal	Self Enquiry, Company Invited
CityTier	Integer	Ordinal	1 - 4
DurationOfPitch	Integer	Numerical	5 - 36
Occupation	String	Nominal	Salaried, Free Lancer, Small Business, Large Business
Gender	String	Nominal	Male, Female
NumberOfPersonVisiting	Integer	Numerical	1 - 4
NumberOfFollowups	Integer	Numerical	1 - 6
ProductPitched	String	Ordinal	Basic, Standard, Deluxe, Super Deluxe, King
PreferredPropertyStar	Integer	Ordinal	3 - 5
MaritalStatus	String	Nominal	Single, Unmarried, Married, Divorced
NumberOfTrips	Integer	Numerical	1 - 20
Passport	Integer	Nominal	0 - 1
PitchSatisfactionScore	Integer	Numerical	1 - 5
OwnCar	Integer	Nominal	0 - 1
NumberOfChildrenVisiting	Integer	Numerical	0 - 3
Designation	String	Ordinal	Manager, Senior Manager, AVP, VP, Executive
MonthlyIncome	Integer	Numerical	1000 - 34246
ProdTaken	Integer	Nominal	0 - 1

Preparation part (The function that I use to prepare the data)

Manual data split into training&validation and testing(inference). I choose to divide it into 85% training&validation and 15% testing. This is done to know that the model is well generalized for other dataset sets, and isn't overfitted by just remembering the training&validation data.

Handling Missing Values: Used df.dropna() to remove any rows with null values. To ensure the model doesn't encounter undefined data during training to confuse it.

Separation: Features (X) and the target variable (ProdTaken, indicating if a product was taken) are separated. This is to prevent Data Leakage. Because if we left ProdTaken inside the features, the model would simply see remember by looking at the ProdTaken result. It would result in a really high accuracy (like 98 or 99 percent) during training but would be completely useless when it got tested with data it doesn't know.

Categorical Encoding: Used (pd.get_dummies) to convert categorical text like Gender or Occupation into numerical values so the neural network can process more effectively.

Data Splitting for validation: Split into 90% Training and 10% Validation. I used a stratified split to make sure that both sets have the same ratio of 'Product Taken' vs 'Not Taken' in the splitted dataset. Because if a random split was used then it might accidentally put all the "Yes" cases in the training set and none in the validation set, which will worsen the result.

Scaling: Applied StandardScaler to normalize numerical features. The model uses weights to decide importance. So if MonthlyIncome is 20,000 and Age is 30, the model might think Income is much more important just because the number is bigger. This helps with that by making all features around 0 with a standard deviation of 1, so the model treats every feature equally.

Oversampling: I notice that the dataset was imbalanced (many more 'Not Taken' than 'Taken' results). We manually oversampled the positive class (ProdTaken=1) to match the negative class exactly. This prevents the model from being biased toward the majority class.

Feature Extraction

Log Transformations: DurationOfPitch_log and MonthlyIncome_log were created using the np.log1p code. This compresses high value ranges and helps the model handle outliers or skewed financial data more effectively. These are used to "squish" columns with extreme outliers so the model isn't confused by one person whose data is way off the average.

I also created new data and information based on the existing data. I believe that it can help the model see non-linear relationships and correlation.

Some of those new data and information:

Income_Pitch_Score: Combines income with satisfaction to see if high earners react differently to pitch quality.

Age_Income: Captures the combined effect of maturity and purchasing power.

Income_per_Person: Derived by dividing income by the number of visitors, representing disposable income.

Polynomial Features: I also squared some of the values such as Age, MonthlyIncome, and DurationOfPitch, and named it as Age_2, MonthlyIncome_2, and Duration_2. This allows the model to capture some trends where both very low and very high values might lead to the same outcome.

Age Binning: This categorizes the Age into groups (Young, Middle, Senior, Old) and allows the model to treat age phases as distinct behavioral segments.

Building the Model

Created the layer using Wide and Deep path.
I found it by searching that the Wide model is good at remembering simple patterns such as (people with high income like this product). While a deep model is good at finding complex patterns.
Wide Path: A simple dense layer that memorizes specific, direct relationships in the data.
Deep Path: A multi-layered structure (256 -> 128 -> 64 neurons) that generalizes and discovers complex patterns.

Activation (Swish): I changed some to this instead of ReLu and the result is better.
The Swish one used a smooth activation function that sometimes performs better than standard ReLU in deep networks.

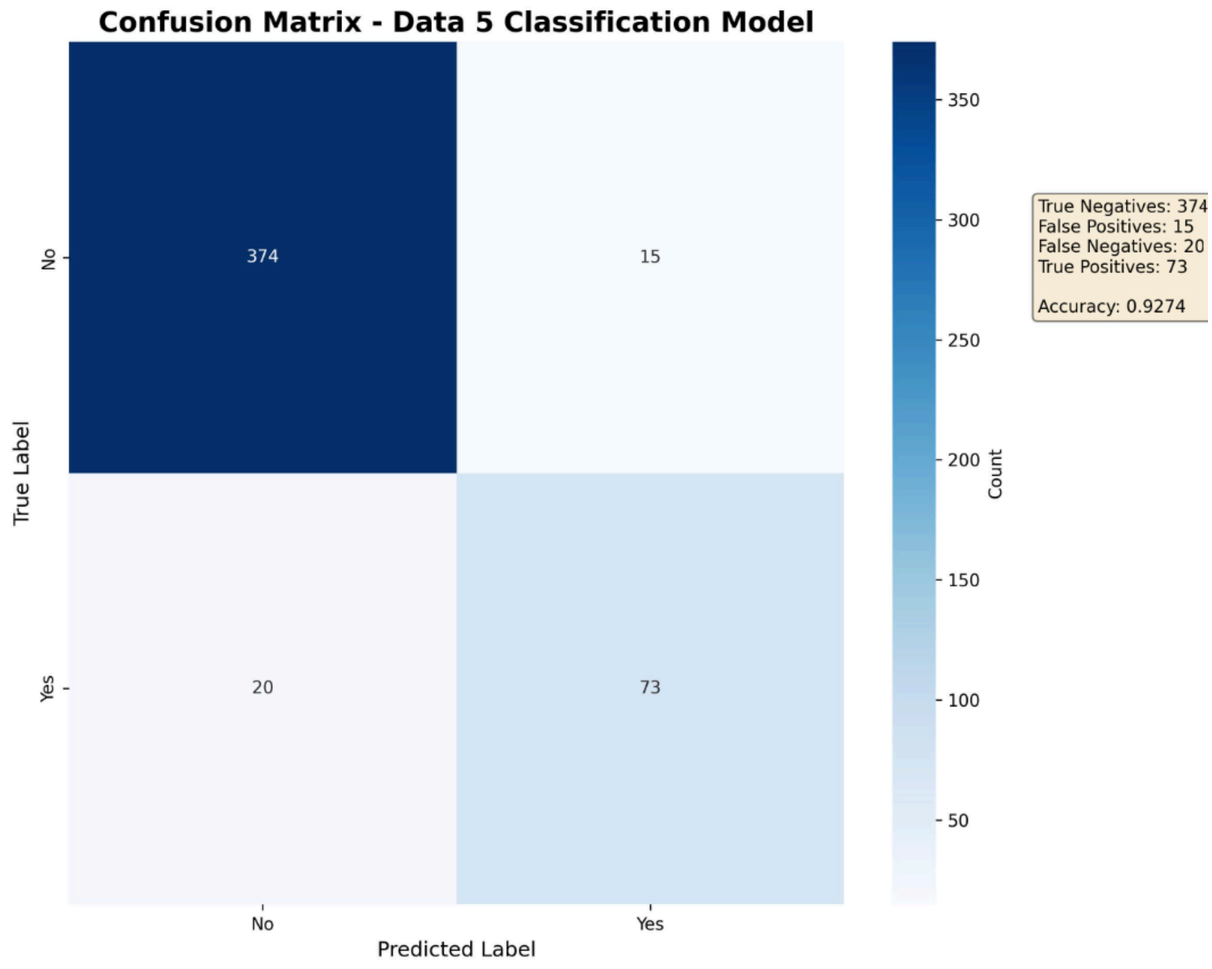
BatchNormalization: Stabilizes the training process by re-centering inputs to each layer.
Dropout (0.4/0.3): Randomly "turns off" neurons during training to prevent the model from memorizing the training data (overfitting).

L2 Regularization: This one adds a small penalty for large weights to keep the model weights small and stable.

Optimizer (Adam): An adaptive learning rate optimizer that efficiently finds the best weights.

Loss Function (Binary Crossentropy): I've tried to use many loss functions and this one gives the best result. Likely because it is specially designed for binary classification yes/no tasks.

Iteration
This is the result of me using the other loss function which is the focal one. And the accuracy significantly reduced from around 94%-96% to just 92%-93%.

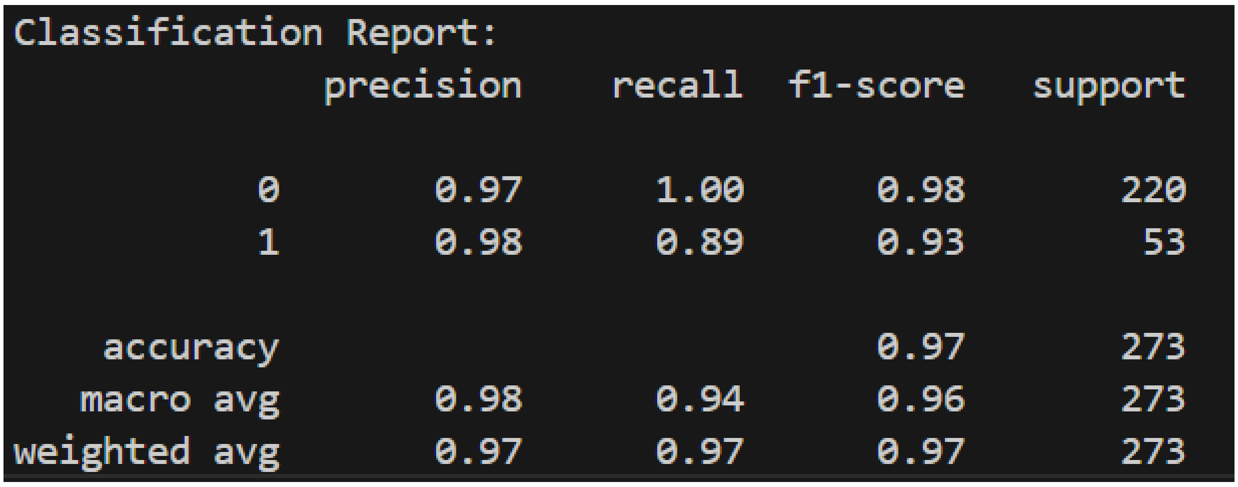


Callbacks:
EarlyStopping: Added this to stop the training if the validation performance stops improving, preventing waste of time and overfitting.
ReduceLROnPlateau: This helps lower the learning rate if the model gets stuck.

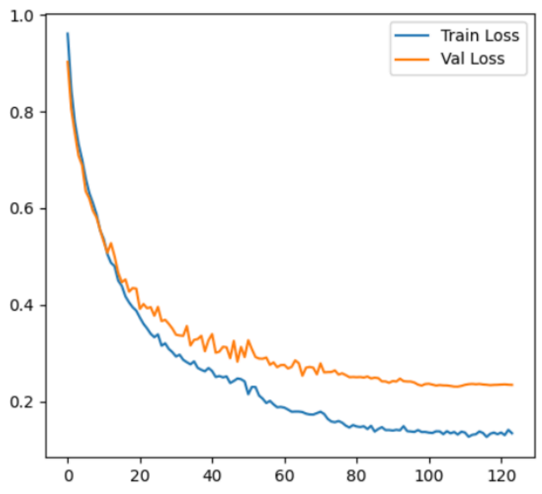
Evaluation Results

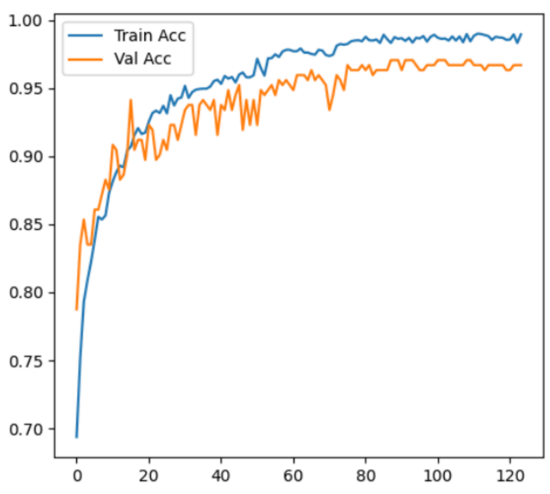
Training result

```
Classification Report:
                 precision   recall   f1-score   support

       0           0.97        1.00     0.98      220
       1           0.98         0.89    0.93      53

    accuracy                    0.97    273
   macro avg       0.98        0.94     0.96      273
weighted avg       0.97        0.97     0.97      273
```

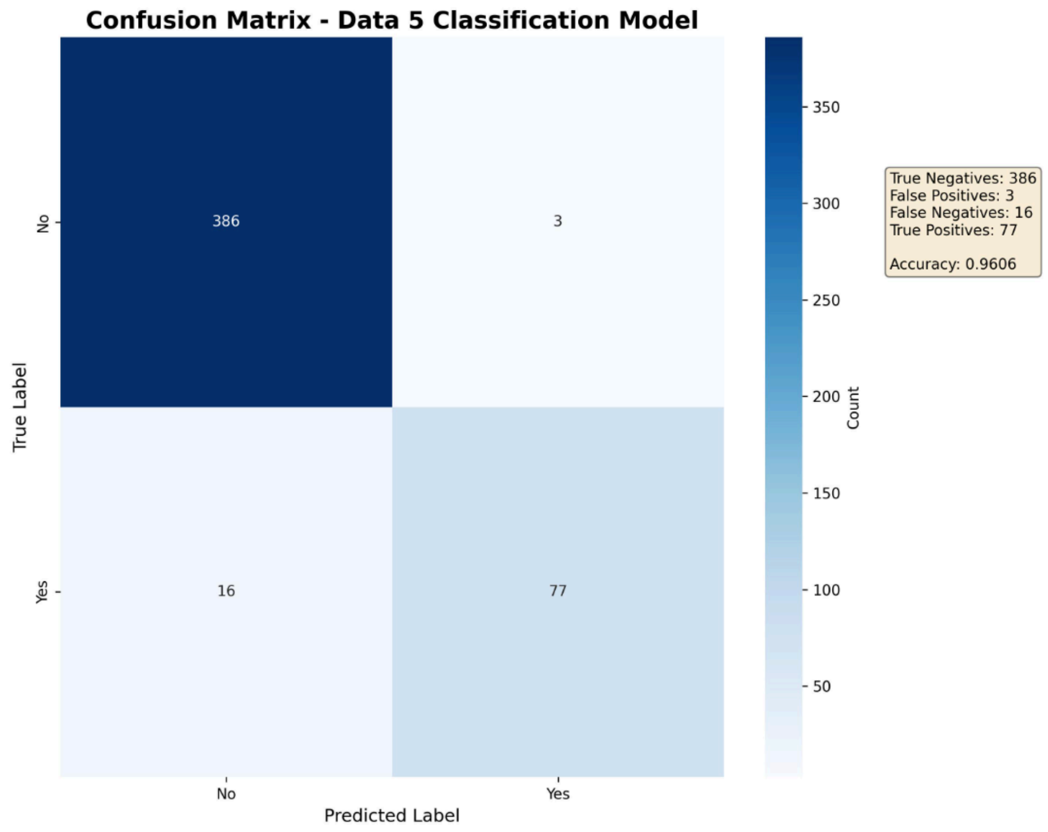


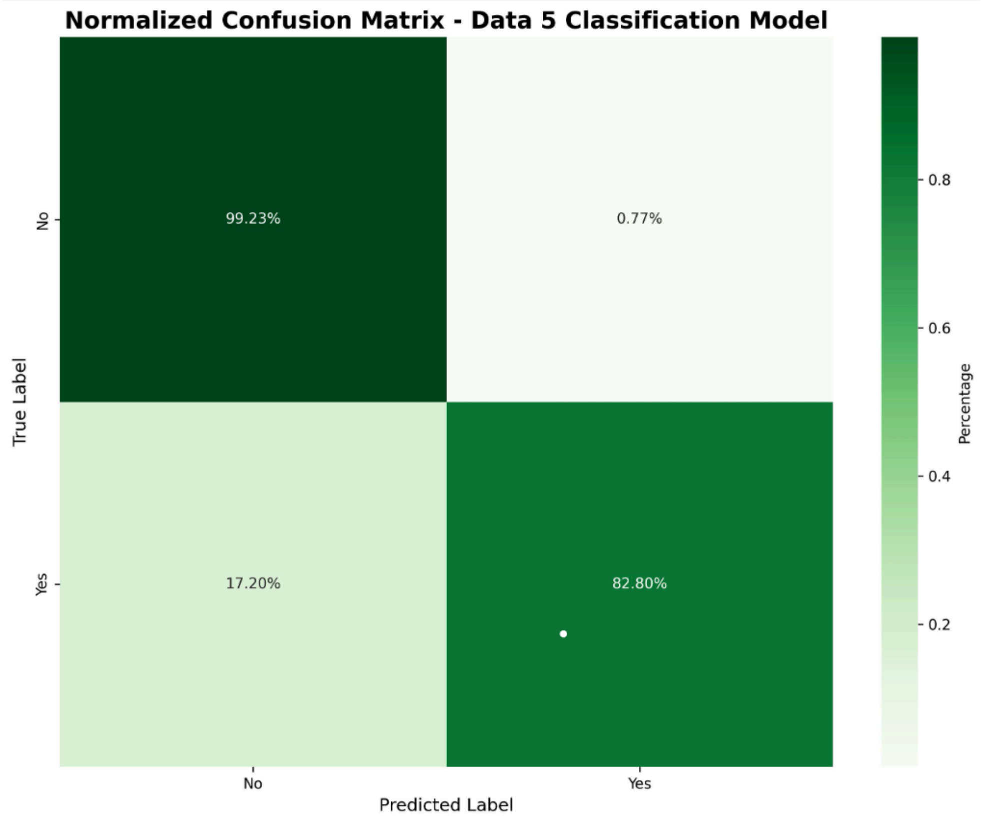
```
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       220
           1       0.98      0.89      0.93        53

    accuracy                           0.97       273
   macro avg       0.98      0.94      0.96       273
weighted avg       0.97      0.97      0.97       273
```





Inference result


Confusion Matrix - Data 5 Classification Model
True Label
Predicted Label
No
Yes
Count
True Negatives: 386
False Positives: 3
False Negatives: 16
True Positives: 77
Accuracy: 0.9606


Normalized Confusion Matrix - Data 5 Classification Model
True Label
Predicted Label
No
Yes
Percentage