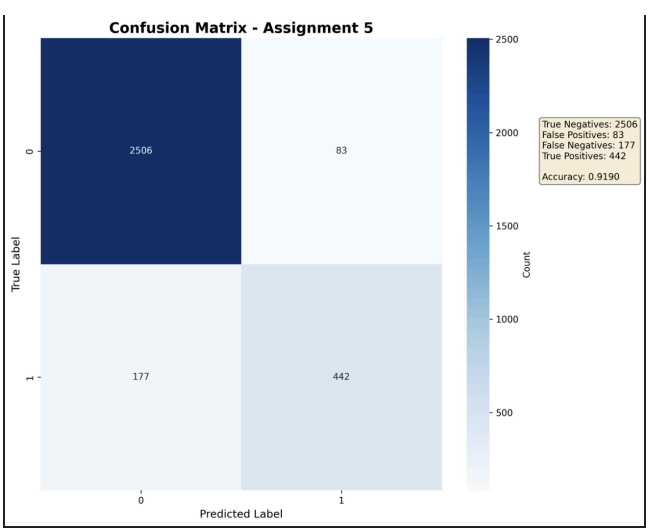
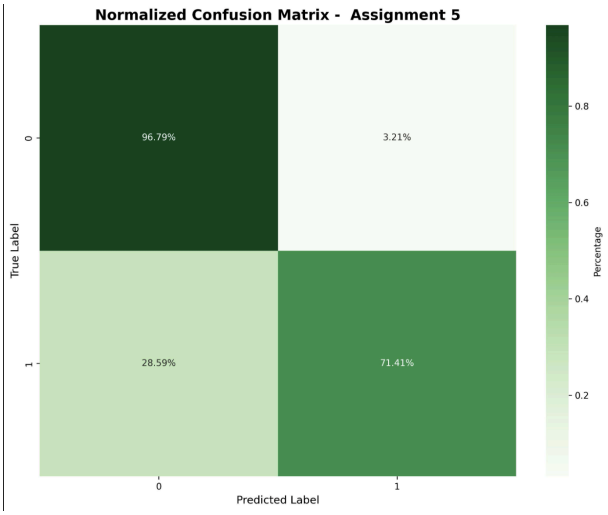
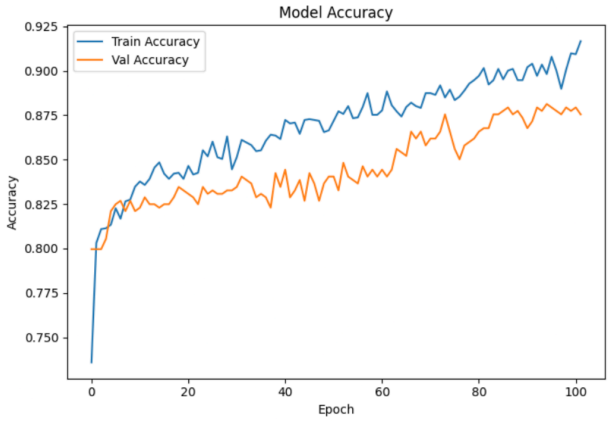
Class 5 Assignment Report

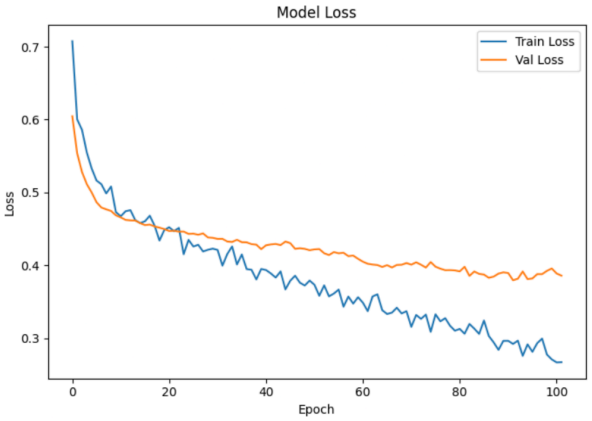
Natchanon Limswatwong 6758029056

Final Results from the submitted model:










```
Accuracy: 0.9190
AUC Score: 0.9490

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      2589
           1       0.84      0.71      0.77       619

    accuracy                           0.92      3208
   macro avg       0.89      0.84      0.86      3208
weighted avg       0.92      0.92      0.92      3208
```

Here's the summary of what I did:

Executive summary

Disclaimer that my goal for this assignment is to do it by my understanding so it might have some part that I understand incorrectly.

I started by cleaning two data quality issues: 115 instances of 'Fe Male' typo in Gender and 510 instances of 'Single' that should be merged with 'Unmarried'. After cleaning, I ran correlation analysis and dropped four features with near-zero correlation (less than 0.01): NumberOfTrips, NumberOfChildrenVisiting, NumberOfPersonVisiting, and OwnCar. I then applied one-hot encoding for categorical variables and StandardScaler for normalization.

I built a 4-layer neural network (128-64-32-16-1) using weighted binary crossentropy with pos_weight=4.0 to handle class imbalance. The initial model achieved 94.51% accuracy but showed signs of overfitting, with training and validation accuracy diverging. To fix this, I added Dropout layers (0.4, 0.4, 0.3, 0.3) across all hidden layers. This significantly reduced overfitting, bringing the training and validation curves much closer together while maintaining strong predictive performance.

The final model successfully balances accuracy and generalization, making it suitable for predicting customer purchase behavior on unseen data.

Pipeline Step 1 - 2 : Data Preparation & Analysis

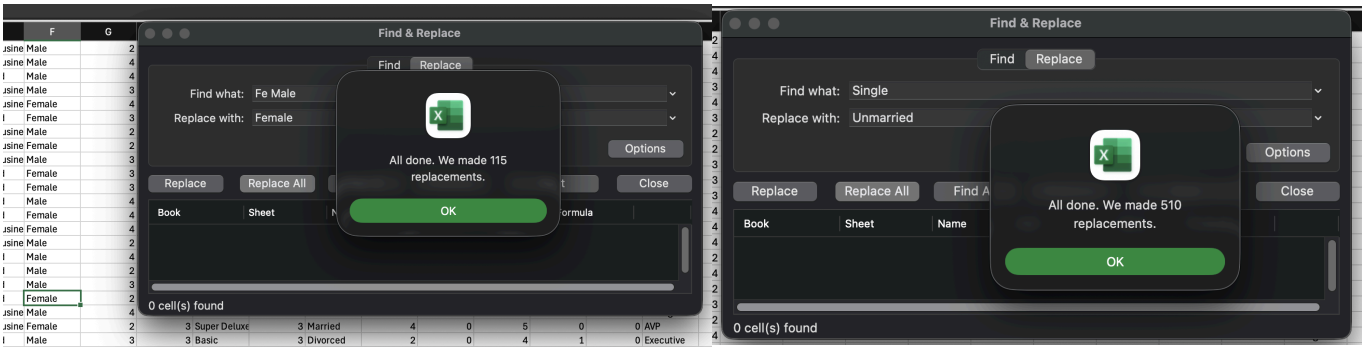
```
Loading data...
Dataset shape: (3208, 19)

Column names: ['Age', 'TypeofContact', 'CityTier', 'DurationOfPitch', 'Occupation', 'Gender', 'NumberOfPersonVisiting', 'NumberOfFollowups', 'ProductPitched', 'PreferredPrope
rtyStar', 'MaritalStatus', 'NumberOfTrips', 'Passport', 'PitchSatisfactionScore', 'OwnCar', 'NumberOfChildrenVisiting', 'Designation', 'MonthlyIncome', 'ProdTaken']

Target distribution:
ProdTaken
0    2589
1     619
Name: count, dtype: int64
```

Firstly, I look into the dataset to see the overview of the data first. The dataset came with 3,208 rows and 19 columns with a very imbalance ratio of yes / no : 2,589 rows = No (81%), 619 rows = Yes (19%) roughly 4:1 so my first action after that is to customize the data to make it equally balance in the number reducing training bias.

After that I found and cleaned two data quality issues.

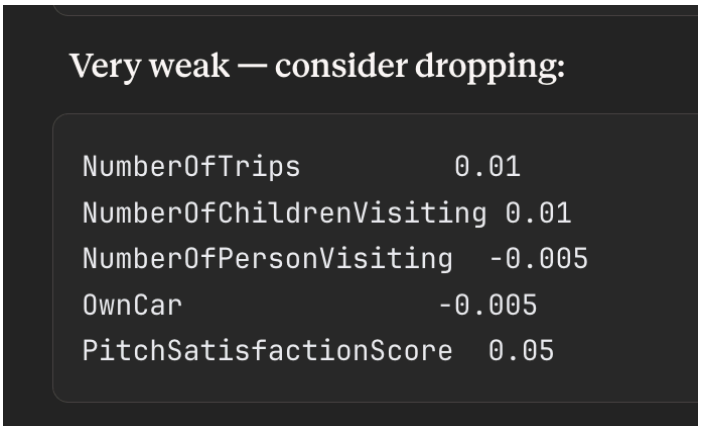


First, there were around 115 instances of 'Fe Male' typo in the Gender column.

Second, there were 510 instances of 'Single' in the MaritalStatus column that should be merged with 'Unmarried' since they mean the same thing. I fixed both issues manually using Excel before uploading this data to the VS code.

Pipeline Step 3 : Feature Extraction:

After that I ran a correlation analysis against ProdTaken to see which features actually matter and have influence over the Y . I found four columns with near-zero correlation, less than 0.01. These were NumberOfTrips, NumberOfChildrenVisiting, NumberOfPersonVisiting, and OwnCar.



I dropped them because they do not help predict the target. After that, I applied one-hot encoding on all categorical columns and used StandardScaler for normalization.

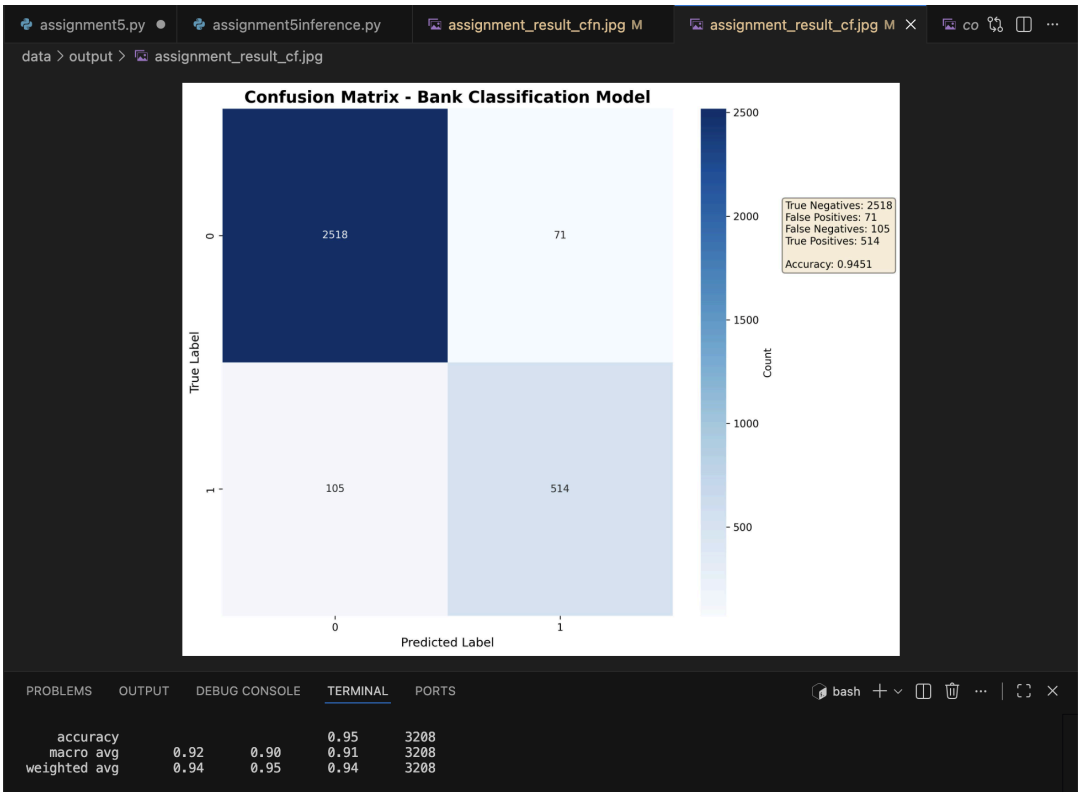
Pipeline Step 4 Model Building:

Then I chose to split data 80/20 before training

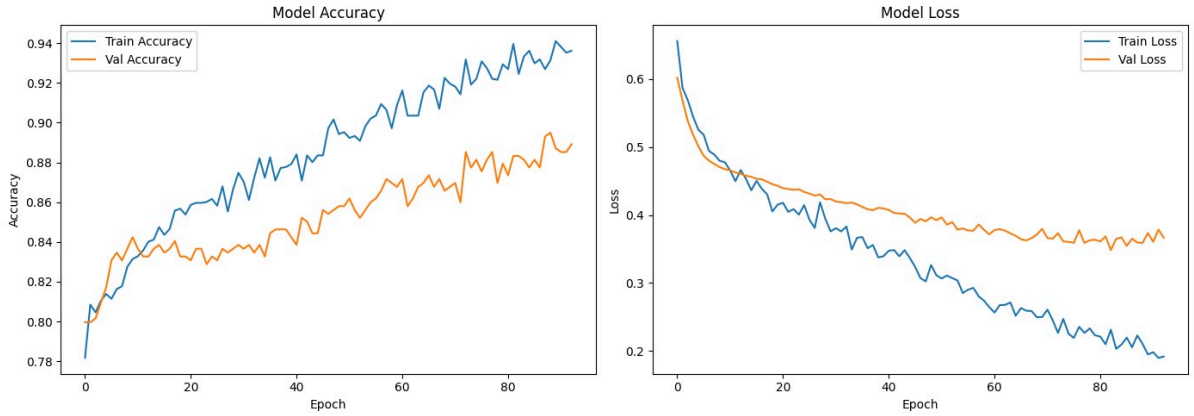
After a few tries to build the model and find the most optimal number of accuracy I ended up using this following number :

- Architecture: 128 (relu) → 64 (relu) → 32 (relu) → 16 → 1 (sigmoid)
- Loss: weighted_binary_crossentropy with pos_weight=4.0
- Optimizer: Adam with learning rate 5e-4
- Epochs: 300 max : Last time I test with p'ming in class he told me to set this high because there would be EarlyStopping kicks in before that (patience=10)

Note that I also have tried several loss functions including Focal loss and binary_crossentropy. But it doesn't have a good performance so I switched back to weighted_binary_crossentropy. With these setup I got the accuracy around 90%+ ; maximum I got was 94.51% (It's not for bank classification model I just forgot to change the header of inference to "assignment 5" T-T) as can be seen in the following pictures



However after plotting the training graph, I noticed the train accuracy and validation accuracy were diverging. Training accuracy kept improving, but validation accuracy was not following. This is a sign of overfitting so I take an action



To fix this, I added Dropout to the hidden layer to reduce the overfitting issue. And here's the Updated architecture

Dense(128) → Dropout(0.4) → Dense(64) → Dropout(0.4) → Dense(32) → Dropout(0.3) → Dense(16) → Dropout(0.3) → Dense(1, sigmoid)

After adding Dropout, the train and val accuracy curves became much closer as shown in the result of the final code and we ended up finalized this model as submitted version

Pipeline Step 5 : Evaluation

To see how the model actually performed, I looked at a few different angles: overall accuracy, the weighted F1-score, and a full breakdown of precision and recall.

Since our data is leaning heavily toward one side (before cleaning), I focused mostly on the weighted F1-score to get a fair picture. I also paid close attention to the recall for potential buyers (Class 1), because from a real world setting , overlooking a customer is much more expensive than reaching out to someone who isn't interested.

Conclusion

Overall, my model follows the ML pipeline from class: prepare and clean the data, analyze features, encode and scale, train a neural network, and evaluate using the right metrics for imbalanced data.

The biggest things that actually moved the needle:

- **weighted_binary_crossentropy** — without this the model ignored the minority class entirely
- **Dropout layers (0.4, 0.4, 0.3, 0.3)** — spotted overfitting from the training graph where train and val accuracy were diverging, added dropout which closed that gap significantly
- **One-hot encoding + StandardScaler** — proper preprocessing so the model could actually learn from text columns and wasn't dominated by high-value columns like MonthlyIncome
- **Feature selection via correlation** — dropped 4 near-zero features to reduce noise

Weighted Average F-1 Score: 91.9%

data > output > tourism_inference_report.txt

```
======================================================================
Inference Results (Tourism)
======================================================================
Dataset:    data/tourism.csv
Model:      examples/assignment.h5
Threshold:  0.500

Accuracy:   0.9190
AUC:        0.9490
Weighted Precision: 0.9190
Weighted Recall:    0.9190
Weighted F1-Score:  0.9190

Confusion Matrix:
[[2506   83]
 [ 177  442]]

Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.97      0.95      2589
           1       0.84      0.71      0.77       619

    accuracy                           0.92      3208
   macro avg       0.89      0.84      0.86      3208
weighted avg       0.92      0.92      0.92      3208

```