	PES UNIVERSITY (Established under Karnataka Act No. 16 of 2013) 100 Ft. Road, BSK III Stage, Bengaluru – 560 085 DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING SESSION: Aug-Dec 2021

Course Title: Ethical Algorithm Design	
Course code: UE18CS400SJ	
Semester: VII	Team Id: 17
SRN: PES1201801222	Name: Soundarya Ganesh
SRN: PES1201800149	Name: Araz Sharma

ASSIGNMENT REPORT

Problem Statement

A Comparative Study of Differential Privacy with Neural Networks, using the Bank Based Churn Modelling Dataset

Name and Signature of the Faculty

Description & Steps:

(Output Screenshots included)

1. Dataset Selection
2. EDA (Exploratory Data Analysis)
3. Feature Selection (Information + Privacy)
4. Pre-Processing
    a. Check for Missing and Anomalous data
    b. Encoding Categorical Columns
    c. Standard Scaling
5. Train-Validation-Test Split
6. Two Approaches for Training
    I. Training an ANN
        1. Without Age Modification
        2. With Age Modification
        3. Dropping Age Feature
    II. Training a Differentially Private ANN
7. Results & Performance Metrics
8. Interpretation of Efficiency & Learning Outcomes

1. Dataset Selection

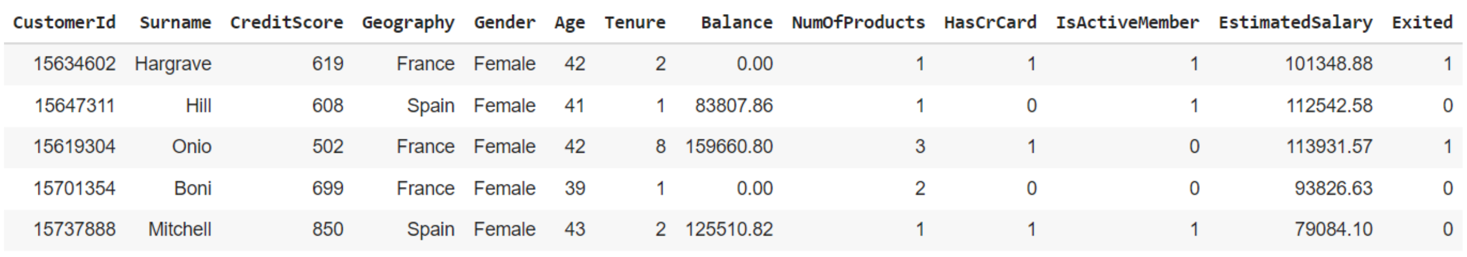
Customer churn is significant to any business because acquiring a new customer is more valuable than selling more to an existing customer. This proves to be an indicator that has the potential to make or break an organization. This restriction also applies to banks. We aim to use data of bank customers to determine which of them are more likely to quit the bank (i.e.close the bank account) or continue to use the bank's services.

For this dataset, Privacy is of the utmost concern, as banks often collect a lot of personal information of their customers. The <u>primary ethical issues</u> to be kept in mind, while using this dataset are:

1. Not to use any personal information that is not absolutely necessary

2. To make a model, from which, a third party cannot trace back individual customer records, from the model predictions

3. Since this dataset is comparatively small (~10,000), we need to maximise performance, therefore use all given records, keeping the above 2 points in mind. We don't have the luxury to choose a subset of the data.



CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSalary	Exited
15634602	Hargrave	619	France	Female	42	2	0.00	1	1	1	101348.88	1
15647311	Hill	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0
15619304	Onio	502	France	Female	42	8	159660.80	3	1	0	113931.57	1
15701354	Boni	699	France	Female	39	1	0.00	2	0	0	93826.63	0
15737888	Mitchell	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0

2. EDA (Exploratory Data Analysis)

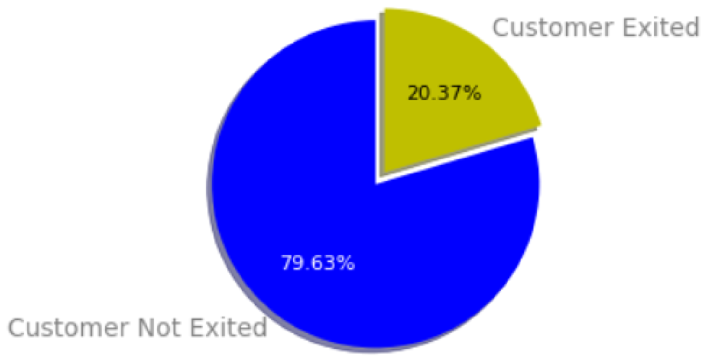


Fig 1: Imbalance in the target variable where 1 (yellow): exited/ churned customers and 0 (blue): retained/ not being churned

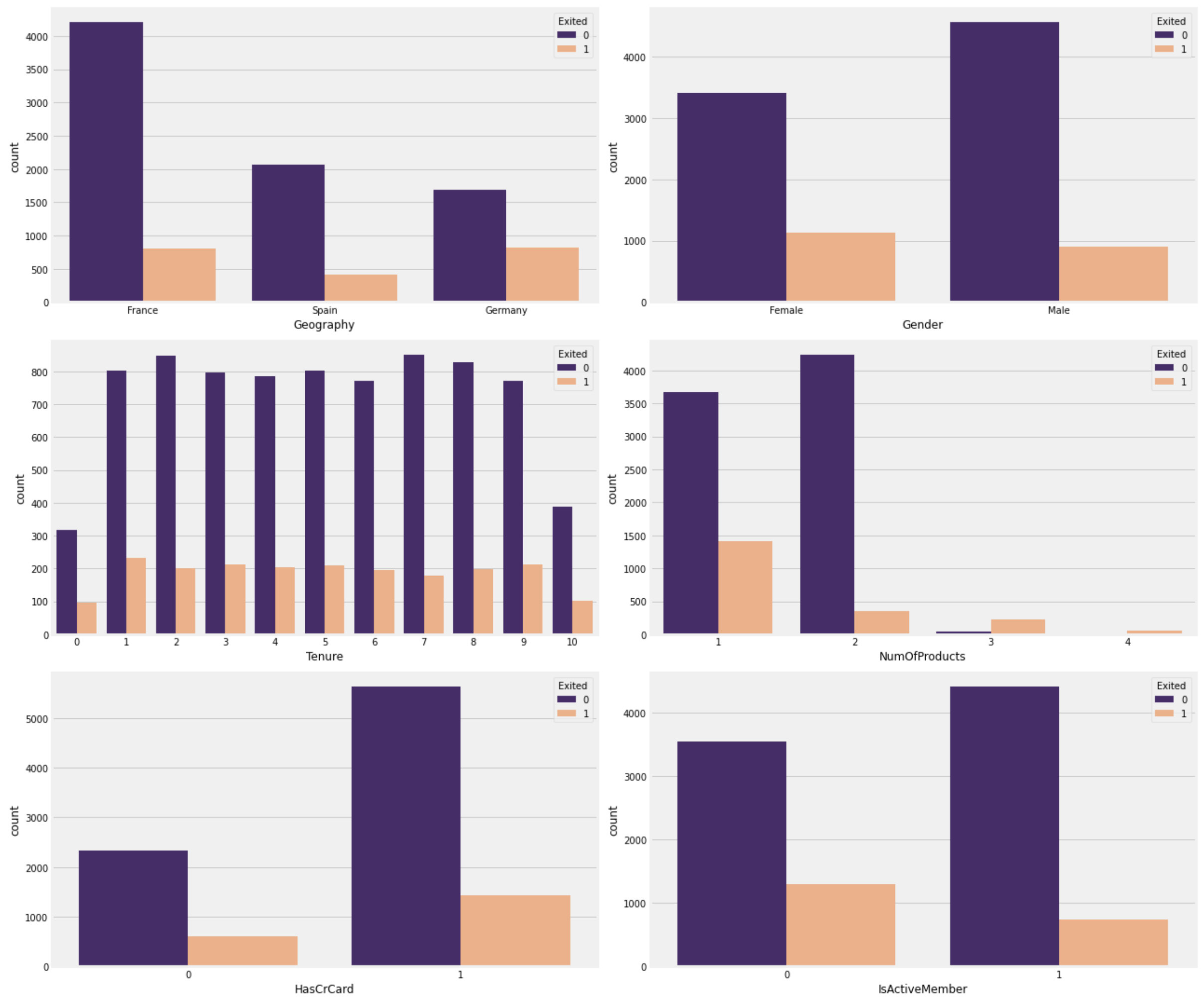


Fig 2: Bar plots to compare different features with the target variable (Exited); some useful insights include that Spain has a lower churn rate than France and Germany; customers with a card is more likely to stay/ retain according to this data

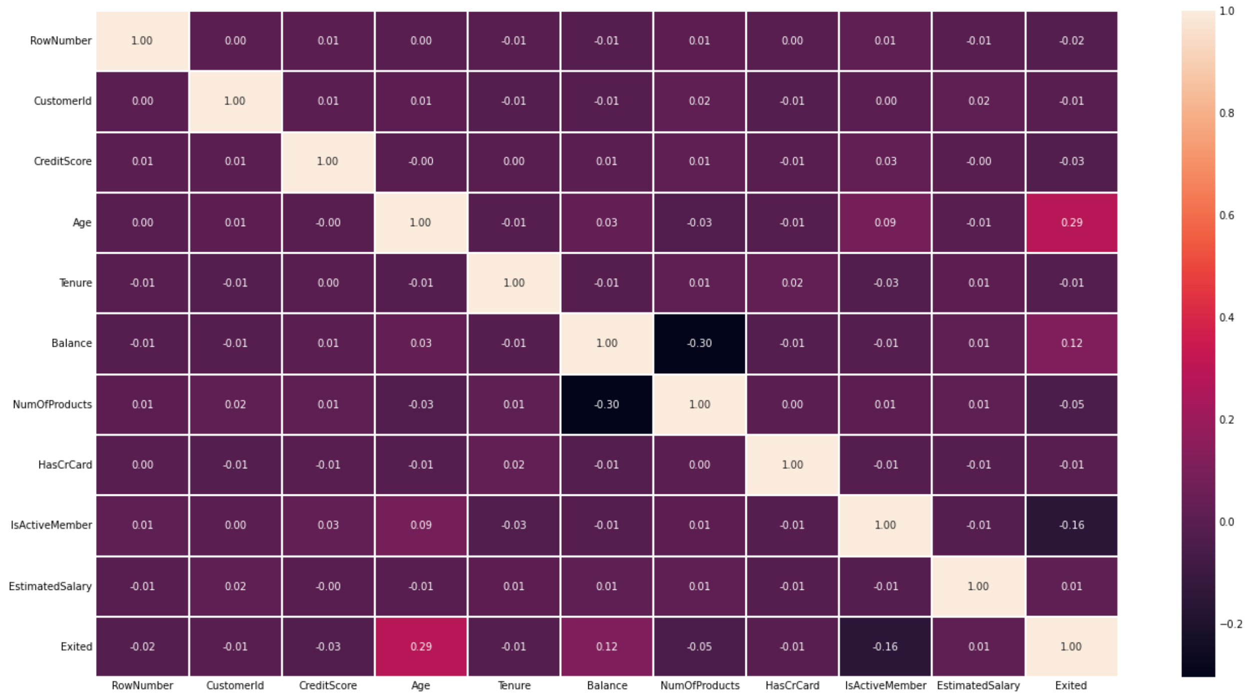


Fig 3: Correlation matrix- correlation between features

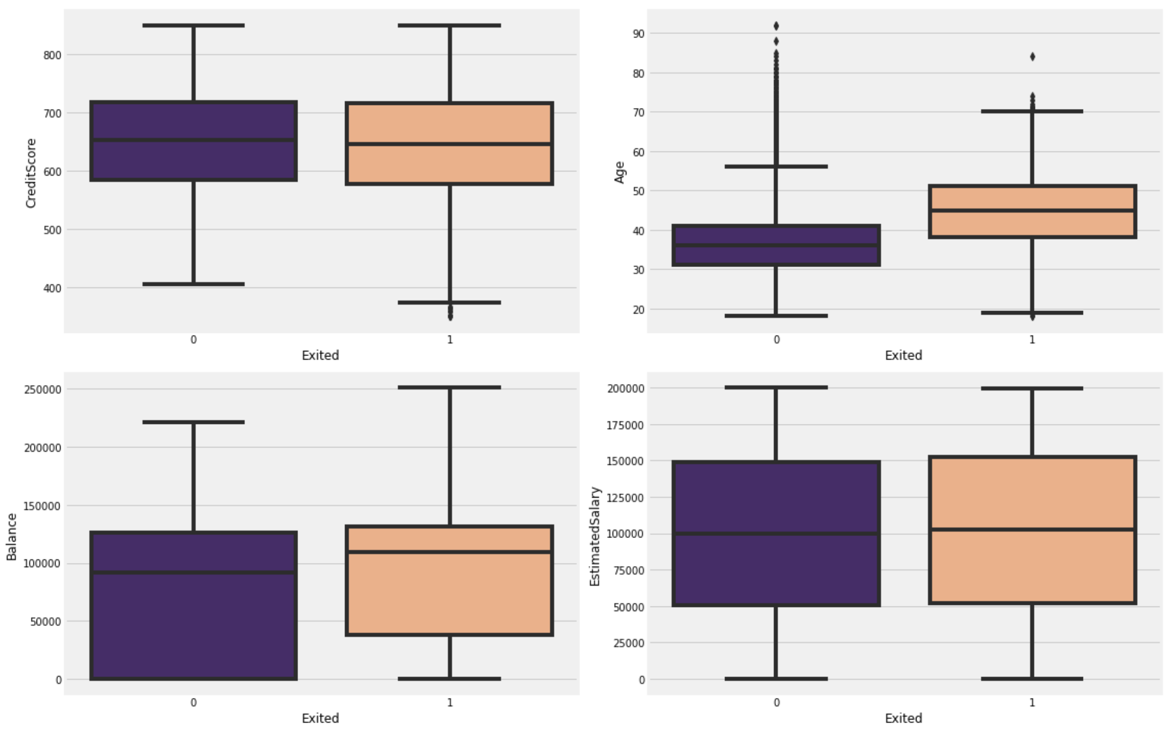
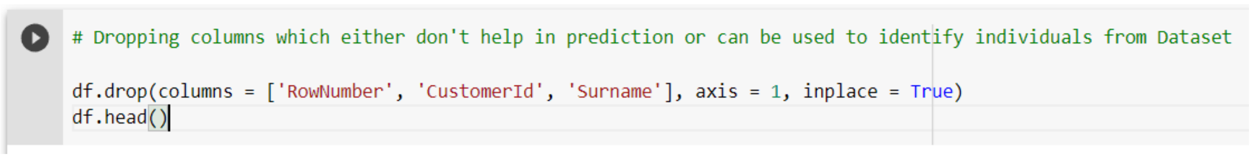


Fig 4: Box plot; some insights from these plots include: Customers with lower bank balance are more likely to stay with the bank; customers in their late 30s to early 50s are more likely to churn

3. Feature Selection:

1. Columns like Customer ID and Surname are a threat to customers' privacy. Such columns can be used to uniquely identify individuals so we considered dropping them. Row number is just an index.
2. All these columns don't provide information thereby not being used for prediction.

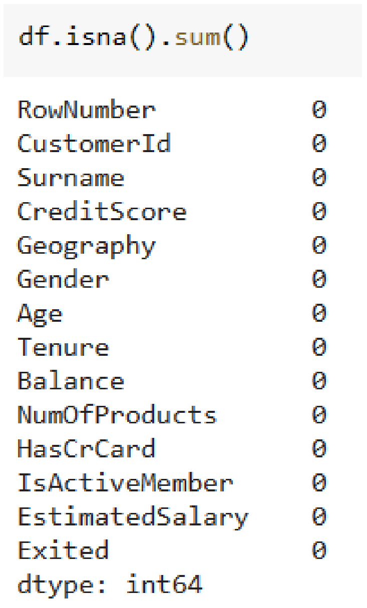


```
# Dropping columns which either don't help in prediction or can be used to identify individuals from Dataset

df.drop(columns = ['RowNumber', 'CustomerId', 'Surname'], axis = 1, inplace = True)
df.head()
```

4. Pre-Processing

a. **Check for Missing and Anomalous data**
   The dataset consisting of European Based Bank Customers does not have missing values. There aren't records that deviate from the dataset behaviour (like minimum age for a bank account is 18, no negative ages, tenure not greater than age)

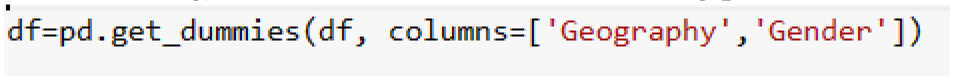


```
df.isna().sum()

RowNumber          0
CustomerId         0
Surname            0
CreditScore        0
Geography          0
Gender             0
Age                0
Tenure             0
Balance            0
NumOfProducts      0
HasCrCard          0
IsActiveMember     0
EstimatedSalary    0
Exited             0
dtype: int64
```

b. Encoding Categorical Columns

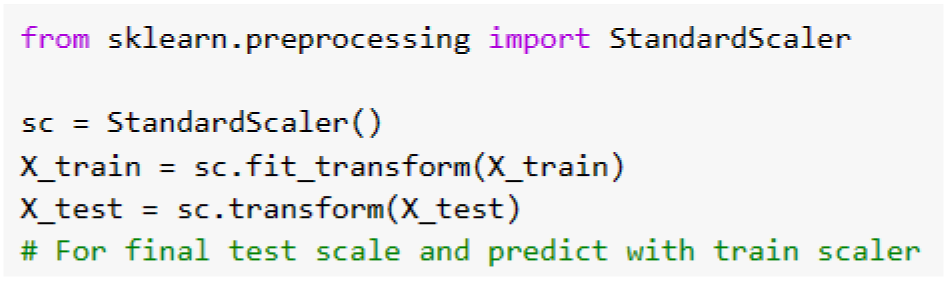
Categorical features like Geography and Gender had to be encoded (One-Hot Encoding) into numerical data before being passed into the ANN.



```
df=pd.get_dummies(df, columns=['Geography','Gender'])
```

c. Standard Scaling using StandardScaler

Since the range of values across columns like Credit Score, Balance, EstimatedSalary vary widely, scaling is a necessary step.

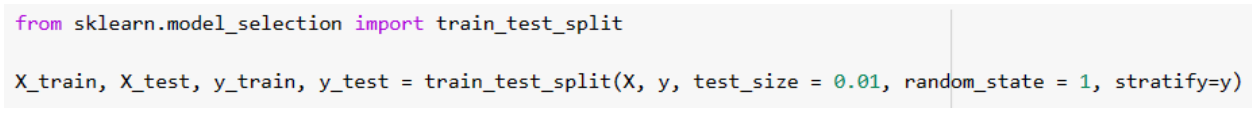


```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
# For final test scale and predict with train scaler
```

5. Train-Validation-Test Split

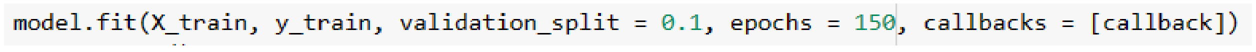
The original dataset is randomly sampled to create a test and train subset that represents the original distribution (using stratify), where the ratio of customers exited (label 1) to retained (label 0) is 20:80. The test and the train size is 100 and 9900 respectively



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.01, random_state = 1, stratify=y)
```

Similarly, there was a validation split created where 10% of the train data was taken for validation (990 records)



```
model.fit(X_train, y_train, validation_split = 0.1, epochs = 150, callbacks = [callback])
```

6. Approaches for Training

- The aim of this Project was to train 2 similar Neural Networks (NN), one trained with a standard optimiser, and the other with a Differentially Private Optimiser (from TF Privacy Library)
- To find Performance vs. Privacy Tradeoff, using these 2 approaches
- The Common NN Model Architecture used was:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_65 (Dense)            (None, 18)                252

 dropout_51 (Dropout)        (None, 18)                0

 batch_normalization_51 (Bat  (None, 18)               72
 chNormalization)

 dense_66 (Dense)            (None, 14)                266

 dropout_52 (Dropout)        (None, 14)                0

 batch_normalization_52 (Bat  (None, 14)               56
 chNormalization)

 dense_67 (Dense)            (None, 10)                150

 dropout_53 (Dropout)        (None, 10)                0

 batch_normalization_53 (Bat  (None, 10)               40
 chNormalization)

 dense_68 (Dense)            (None, 6)                 66

 dropout_54 (Dropout)        (None, 6)                 0

 batch_normalization_54 (Bat  (None, 6)                24
 chNormalization)

 dense_69 (Dense)            (None, 1)                 7

=================================================================
Total params: 933
Trainable params: 837
Non-trainable params: 96
_________________________________________________________________
```

I. Approach 1: Training an Artificial Neural Network

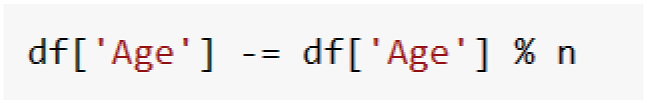
Considering the correlation and the importance of the feature 'Age' from the Churn Data we decided to explore the tradeoff between the privacy of the customer and the performance metric (Refer to the table given in Results)

a. Without Age Modification

- The feature Age consists of numeric continuous integer values which depict the age of the customer.
- We passed the original feature column to the ANN architecture

b. With Age Modification

The 'Age' feature was imputed using a generalization technique where a continuous range of values is coarsened by rounding it to the nearest n.



```
df['Age'] -= df['Age'] % n
```

Considering n to be,

1. Nearest 5: Abstraction of data is introduced by rounding the age of the customer to the floor of nearest 5. Eg: 32 is coarsened to 30 and 49 to 45.
2. Nearest 10: A higher level of abstraction is introduced by rounding the age of the customer to the floor of nearest 10. This leads to more loss of information which causes a slight drop in performance metrics.

c. Dropping Age Feature

Considering the contribution of feature 'Age' in the prediction of the target column, our understanding leads to the conclusion that dropping a significant column leads to a decrease in performance of ANN, and we verified that by dropping the feature. Increase in Privacy led to decrease in Performance.

Training Parameters:

- Using Adam Optimiser with Early Stopping for Patience = 40
- Epochs (Max) = 100 (Actual after Early Stopping ~ 60)
- Loss is 'Binary Cross Entropy'

II. Approach 2: Training a Differentially Private Neural Network

In a Differentially Private Model, we should see that it doesn’t get affected by any single training example or a smaller set of examples from our dataset.

The method behind making it Differentially Private is called differentially private stochastic gradient descent (DP-SGD). Here, you modify the gradients used in stochastic gradient descent.

Models trained with DP-SGD provide provable differential privacy guarantees for the training data. There are 2 changes made to the standard or the vanilla SGD algorithm:

1. You bound the sensitivity of each gradient, by **clipping** the gradient computed for each training point used. Thus, this will limit the effect of each training point on the model parameters.
2. You sample **Random noise** & add that to the clipped gradients, which will make it statistically impossible to know whether or not a particular data point was included in the training dataset you had, by comparing the updates gradient descent applies when it trains with or without this particular data point in the dataset.

For this approach, we use the `DPKerasSGDOptimizer` from the Tensorflow Privacy Library. We use the exact same data, as we had used in approach I, and use a similar train-validation-test split.

For the DP-SGD Optimiser, there are additional privacy related hyperparameters, which we must use:

1. **l2_norm_clip (float)** - This is the maximum Euclidean (L2) norm, for each individual gradient, which is computed on an individual training example from that minibatch. This will be used for sensitivity to individual training points.
2. **noise_multiplier (float)** - This is the amount of noise sampled and added to gradients during training. Generally, more noise results in better privacy, but can result in a lower performance.
3. **microbatches (int)** - Each batch is split into this number of microbatches. Generally, increasing this will improve performance, but also increase your overall training time. The total number of examples consumed in one global step remains the same. We have to ensure that the input batch size should be an integer multiple of the number of microbatches.

For our training, the hyper-parameters we used were:

```
l2_norm_clip = 1.5

noise_multiplier = 1.3

num_microbatches = Same as batch_size

learning_rate = 0.25

batch_size = 128

epochs = 100
```

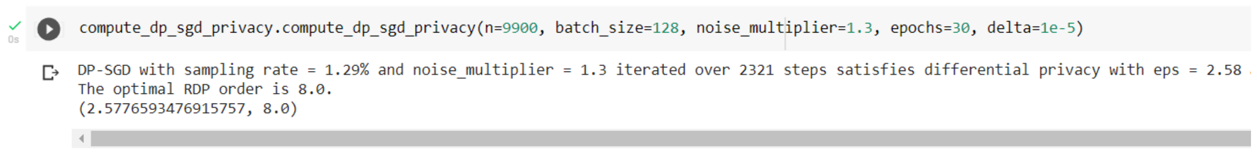
Early Stopping with a Patience of 40 was used, while training the model.

7. Results & Performance Metrics

Performance Metric	Original ANN	Age Generalization		Dropping column Age	Differentially Private ANN
		5	10		
Accuracy	0.8999 (~0.9)	0.86	0.889	0.85	0.899 (~0.9)
Precision	0.8571	0.66	0.846	0.77	0.8571
Recall	0.6	0.6	0.55	0.35	0.6
F1 Score	0.7058	0.6315	0.66	0.482	0.7058

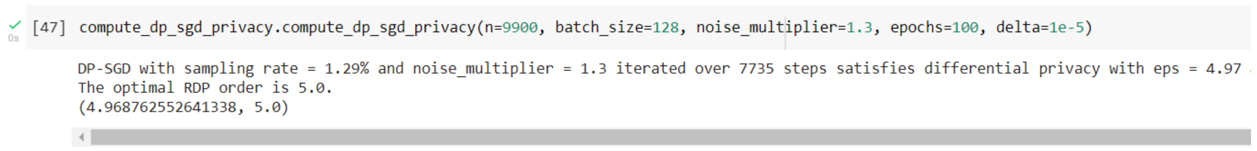
```
Overall Training Time:  0:01:12.714625 (Original ANN)
Overall Training Time:  0:05:27.995714 (Differentially Private ANN)*
```
(* DP ANN takes approximately 5 times longer to train)

Snippets to show Model Evaluation:



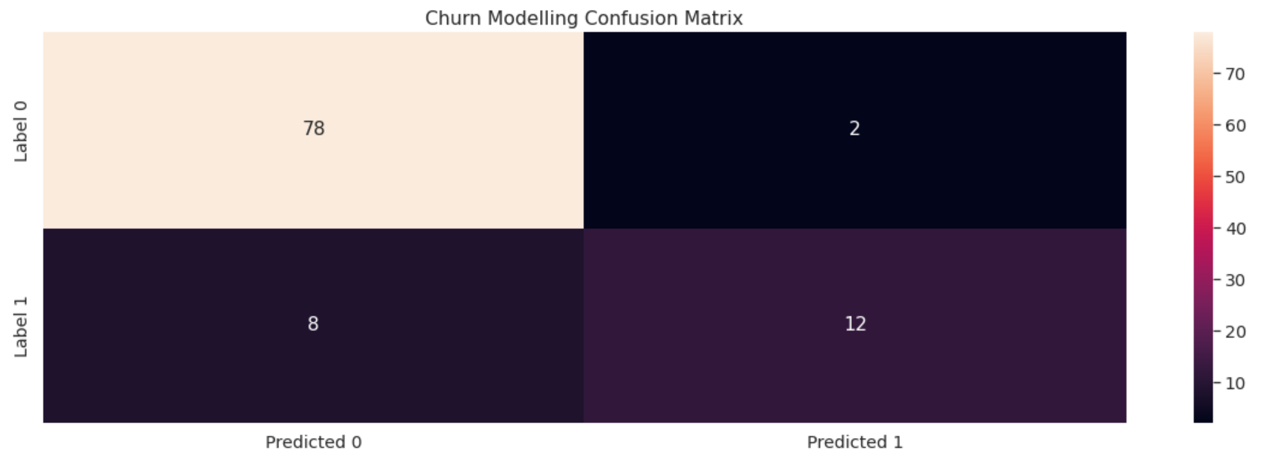
```
compute_dp_sgd_privacy.compute_dp_sgd_privacy(n=9900, batch_size=128, noise_multiplier=1.3, epochs=30, delta=1e-5)

DP-SGD with sampling rate = 1.29% and noise_multiplier = 1.3 iterated over 2321 steps satisfies differential privacy with eps = 2.58
The optimal RDP order is 8.0.
(2.5776593476915757, 8.0)
```



```
[47] compute_dp_sgd_privacy.compute_dp_sgd_privacy(n=9900, batch_size=128, noise_multiplier=1.3, epochs=100, delta=1e-5)

DP-SGD with sampling rate = 1.29% and noise_multiplier = 1.3 iterated over 7735 steps satisfies differential privacy with eps = 4.97
The optimal RDP order is 5.0.
(4.968762552641338, 5.0)
```



```
# Precision: Whenever Model Predicts True, how many times it is actually true
# Recall: Out of all True, how many did the Model find out

# Precision:   True Positives/(True Positives + False Positives)
# Recall:   True Positives/(True Positives + False Negatives)

TP = conf_Log[1][1]
FP = conf_Log[0][1]
TN = conf_Log[0][0]
FN = conf_Log[1][0]
```

```
[ ]  Precision_Metric = TP/(TP + FP)
     print("Precision is:", Precision_Metric)
```

```
Precision is: 0.8571428571428571
```

```
[ ]  Recall_Metric = TP/(TP + FN)
     print("Recall is:", Recall_Metric)
```

```
Recall is: 0.6
```

```
[ ]  F1_Metric = (2*Precision_Metric*Recall_Metric)/(Precision_Metric + Recall_Metric)
     print("F1 Score is:", F1_Metric)
```

```
F1 Score is: 0.7058823529411764
```

8. Interpretation of Efficiency & Learning Outcomes

ε: Epsilon is a metric to measure the level of privacy protection provided on change (addition or removal) of a record from the data. Adding noise makes it difficult to identify the customer and leads to drop in performance metrics.
As that epsilon value increases, the risk of revealing the individual user's information increases as the finer details are learnt by the architecture which might lead to overfitting. Ideally for higher privacy protection the value of epsilon is expected to be smaller.

We see that for 30 Epochs of training, $\varepsilon = 2.57$
We see that for 100 Epochs of training, $\varepsilon = 4.96$

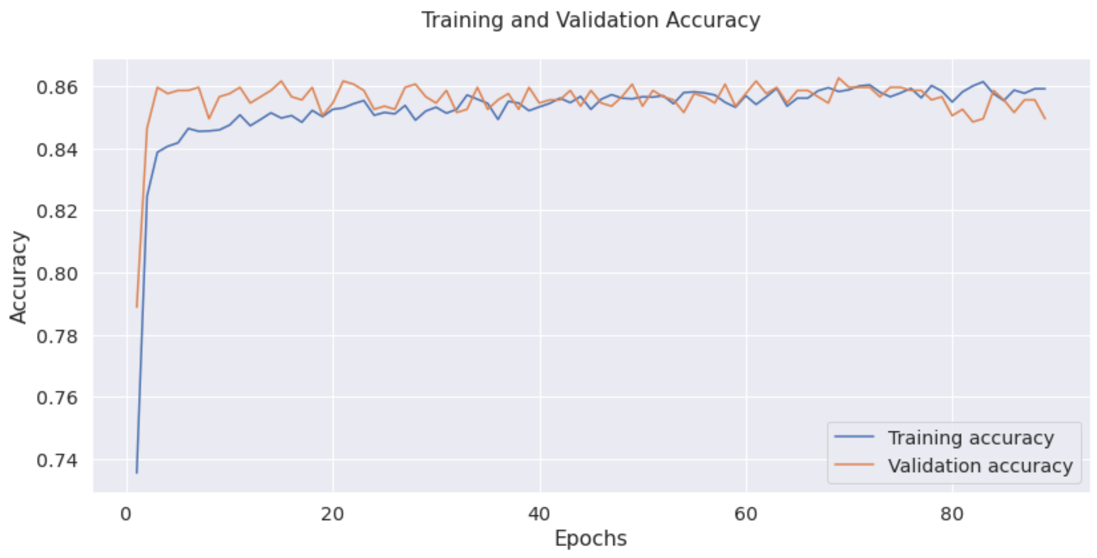
Therefore increase in Performance is at a cost of decrease in Privacy

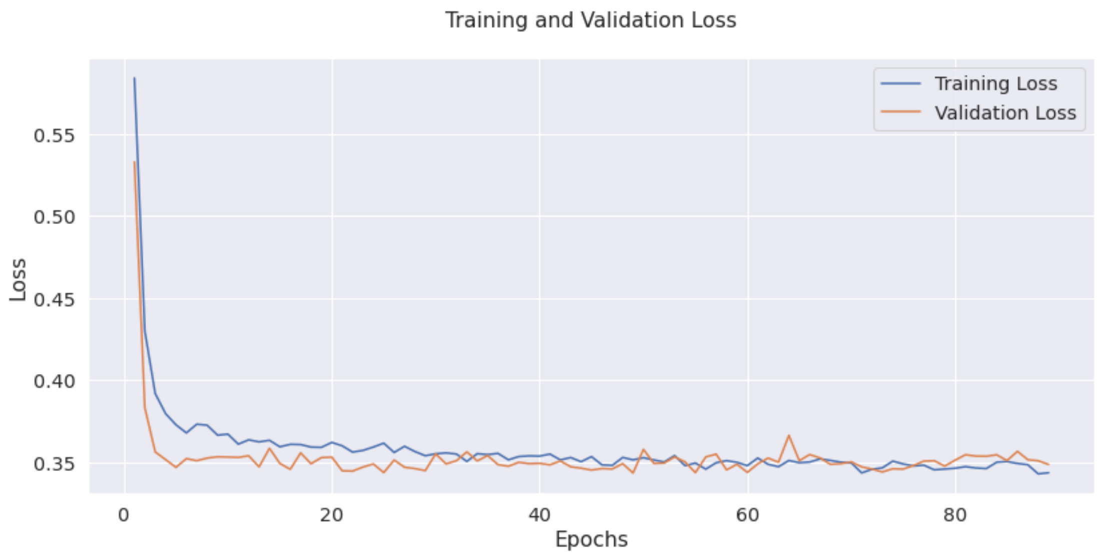
<u>Observations:</u>

- We see that in a standard trained ANN, we can get a good accuracy and F1 score, with a moderate complexity network, and < 100 epochs of training
- Since the data is skewed, with 80% of ‘Exited’ label 0 and 20% of ‘Exited’ label 1, we see that the model is easily able to learn label 0, compared to label 1. So the recall is tougher to improve, and there is a stark difference in accuracy and F1 metric.
- For the first approach, with Age Generalisation, we see more generalisation will result in better privacy, but lesser performance. If we drop age, we see that the F1 score drops significantly, yet accuracy isn’t affected a lot.
- For the Differentially Private ANN, we observe that we can get a similar performance as the standard ANN, at the cost of more training time and greater number of epochs.
- For the standard ANN, we used 50-60 epochs vs for DP ANN 100 epochs
- This makes sense, as the DP process clips gradients and adds noise, which slows down & decreases learning at each epoch

- A nice & surprising effect of the DP process, was that the gradient clipping and noise addition, also acts as a regularising effect, which shows good generalisation of the model on the test set!

We deem our comparative study to be successful, as we were able to come up with a Differentially Private Model, which was able to provide a similar performance as our standard ANN Model. We also realised the Performance vs Privacy Tradeoff, with 2 experiments, seeing that increase in Privacy leads to lesser performance. So to reach a similar performance level, you need more resources.


Training and Validation Accuracy
Accuracy
Epochs
Training accuracy
Validation accuracy


Training and Validation Loss
Loss
Epochs
Training Loss
Validation Loss