Loading Libraries

In [1]:

```
# To help with reading and manipulating data
import pandas as pd
import numpy as np

# To help with data visualization
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns

# To help with model building
from sklearn.model_selection import train_test_split
import tensorflow as tf
from sklearn import preprocessing
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
import keras
from keras import backend as K
from keras.models import Sequential
from keras.layers import Dense, Dropout
from tensorflow.keras import optimizers
from tensorflow.keras.optimizers import Adam

# To get different metric scores
import sklearn.metrics as metrics
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score, recall_score, f1_sc

# To define maximum number of columns to be displayed in a dataframe
pd.set_option("display.max_columns", None)

# To supress scientific notations for a dataframe
pd.set_option("display.float_format", lambda x: "%.3f" % x)

# To supress warnings
import warnings

warnings.filterwarnings("ignore")
```

Load data

In [3]:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
/kaggle/input/bank-churn-prediction/bank.csv
```

In [4]:

```
#Defining the path of the dataset
dataset_file = '/kaggle/input/bank-churn-prediction/bank.csv'
```

In [5]:

```
#reading dataset
data = pd.read_csv(dataset_file)
```

Checking for Duplicates in CustomerID

In [10]:

```python
dupe = data["CustomerId"].duplicated()
dupe[dupe == True].count()
```

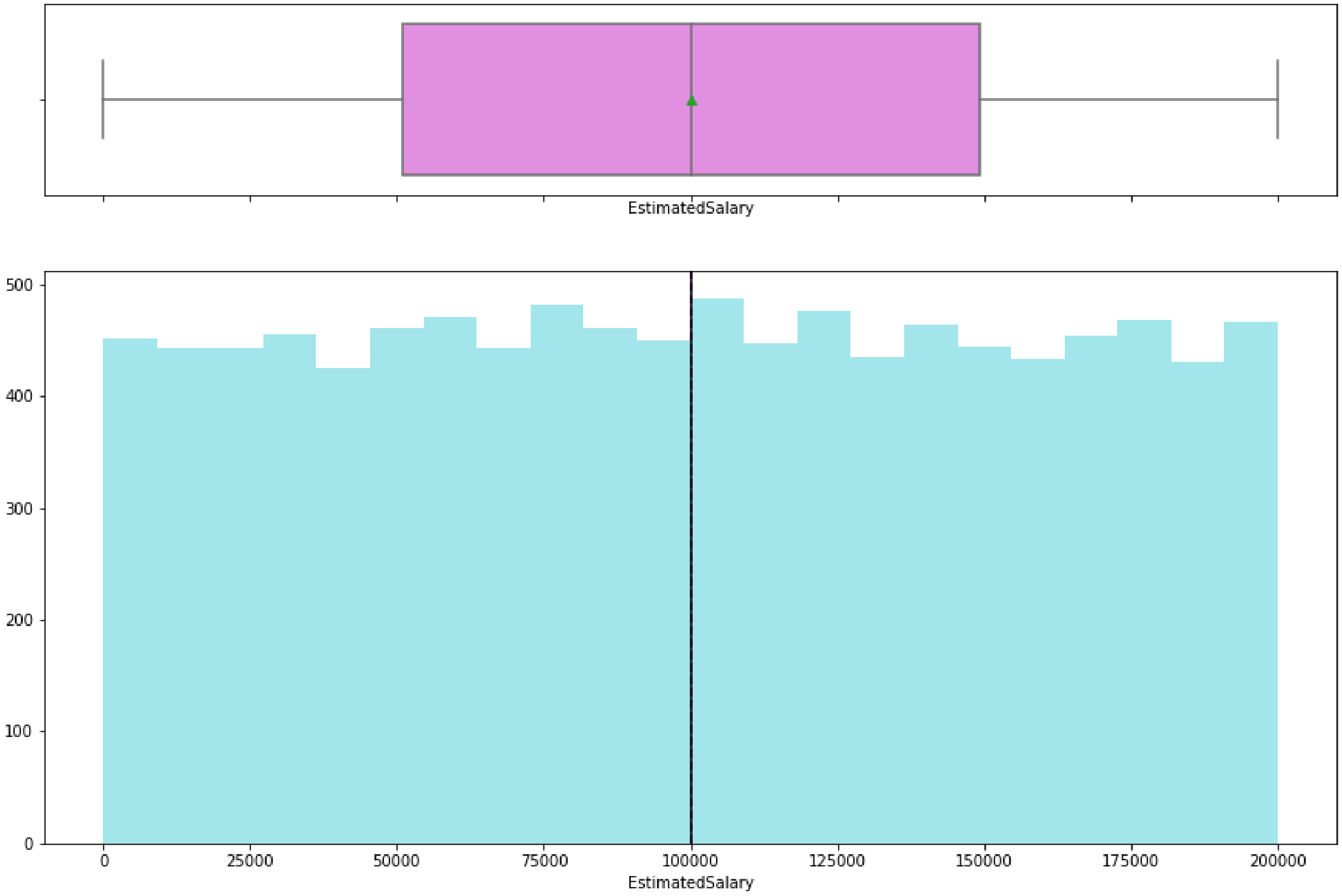
Out[10]:

```
0
```

EstimatedSalary

In [17]:

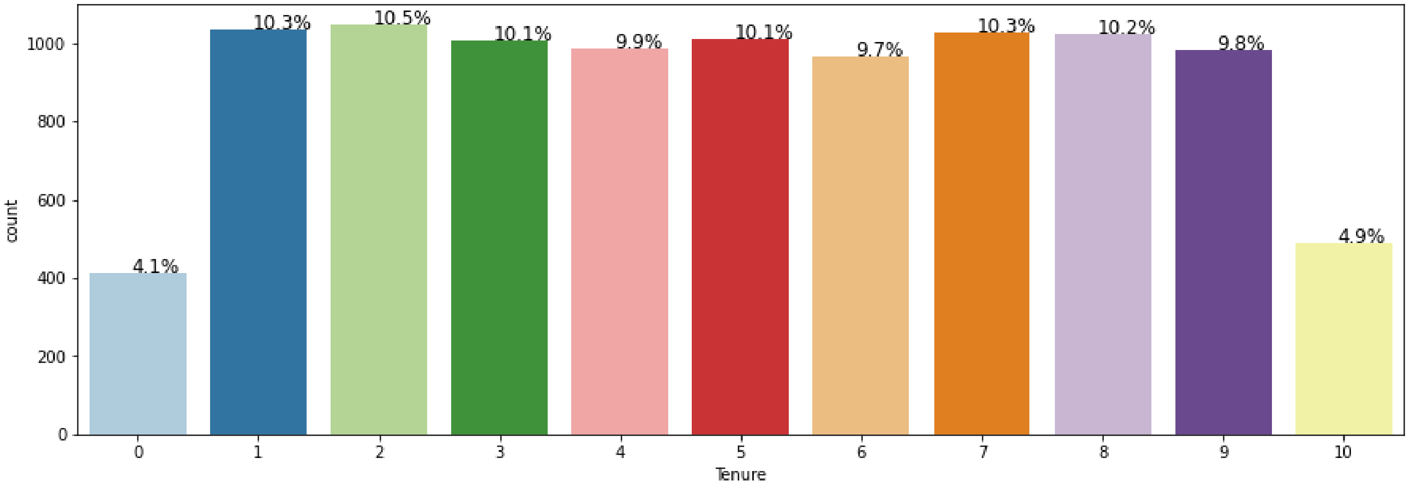
```python
histogram_boxplot(data.EstimatedSalary)
```



Tenure

In [20]:

```python
perc_on_bar("Tenure")
```



Splitting Data into Training, Validation and Test Set

In [39]:

```
data1 = data.copy()

# Separating target variable and other variables
X_data = data1.drop(columns=["CustomerId","Exited", "Surname", "RowNumber"])
Y_data = data1["Exited"]

X_data.head()
```

Out[39]:

	CreditScore	Balance	HasCrCard	IsActiveMember	EstimatedSalary	Age_log	Geography_Germany	Geography_S
0	619	0.000	1	1	101348.880	3.761	0	
1	608	83807.860	0	1	112542.580	3.738	0	
2	502	159660.800	1	0	113931.570	3.761	0	
3	699	0.000	0	0	93826.630	3.689	0	
4	850	125510.820	1	1	79084.100	3.784	0	

In [40]:

```
from sklearn.preprocessing import StandardScaler

# Normalize in [-1,+1] range

Scale_cols = ["CreditScore","Age_log","Balance","EstimatedSalary"]

for col in Scale_cols:
    X_data['normalized'+col] = StandardScaler().fit_transform(X_data[col].values.reshape(-1,1))
    X_data= X_data.drop(col,axis=1)
```

In [41]:

```
X_train, X_test, y_train, y_test = train_test_split(
    X_data, Y_data, test_size=0.25, random_state=1, stratify=Y_data
)
print(X_train.shape, X_test.shape)
```

```
(7500, 22) (2500, 22)
```

In [42]:

```
X_train.head()
```

Out[42]:

	HasCrCard	IsActiveMember	Geography_Germany	Geography_Spain	Gender_Male	Tenure_1	Tenure_2	Tenure_
7971	0	1	0	1	1	0	0	
9152	1	0	1	0	0	0	0	
6732	0	0	0	0	0	0	0	
902	1	1	0	0	0	0	0	
2996	1	0	0	1	0	0	0	

Training [Forward pass and Backpropagation]

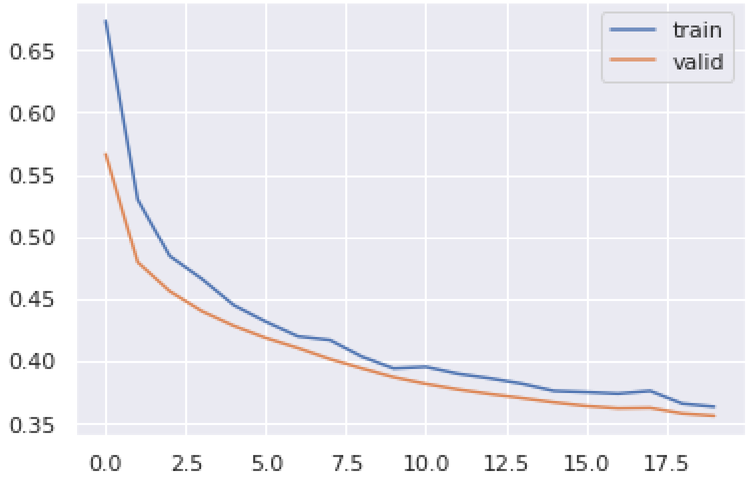
In [48]:

```
# Capturing learning history per epoch
hist  = pd.DataFrame(history.history)
hist['epoch'] = history.epoch

# Plotting accuracy at different epochs
plt.plot(hist['loss'])
plt.plot(hist['val_loss'])
plt.legend(("train" , "valid") , loc =0)
```

Out[48]:

```
<matplotlib.legend.Legend at 0x7f3dfcf96b10>
```



Evaluation

In [49]:

```
score1 = model.evaluate(X_test, y_test)
```

```
79/79 [==============================] - 0s 2ms/step - loss: 0.3798 - accuracy: 0.8464
```

In [50]:

```
yprednn1=model.predict(X_test)
yprednn1=yprednn1.round()
print('Neural Network with relu:\n {}\n'.format(
    metrics.classification_report(yprednn1, y_test)))
```

```
Neural Network with relu:
               precision    recall  f1-score   support

         0.0       0.95      0.87      0.91      2177
         1.0       0.44      0.69      0.54       323

    accuracy                           0.85      2500
   macro avg       0.70      0.78      0.72      2500
weighted avg       0.88      0.85      0.86      2500
```