In [ ]:
```
import os
import numpy as np
import pandas as pd

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

%matplotlib inline
```

In [ ]:
```
path_ = './Churn_Modelling.csv'
df = pd.read_csv(path_, delimiter = ',')
```

In [ ]:
```
df
```

Out[39]:

	RowNumber	CustomerId	Surname	CreditScore	Geography	Gender	Age	Tenure	Balanc
0	1	15634602	Hargrave	619	France	Female	42	2	0.(
1	2	15647311	Hill	608	Spain	Female	41	1	83807.8
2	3	15619304	Onio	502	France	Female	42	8	159660.8
3	4	15701354	Boni	699	France	Female	39	1	0.(
4	5	15737888	Mitchell	850	Spain	Female	43	2	125510.8
...	...	...	...	...	...	...	...	...	
9995	9996	15606229	Obijiaku	771	France	Male	39	5	0.(
9996	9997	15569892	Johnstone	516	France	Male	35	10	57369.6
9997	9998	15584532	Liu	709	France	Female	36	7	0.(
9998	9999	15682355	Sabbatini	772	Germany	Male	42	3	75075.3
9999	10000	15628319	Walker	792	France	Female	28	4	130142.7

10000 rows × 14 columns

In [ ]:
```
df.columns
```

Out[40]:
```
Index(['RowNumber', 'CustomerId', 'Surname', 'CreditScore', 'Geography',
       'Gender', 'Age', 'Tenure', 'Balance', 'NumOfProducts', 'HasCrCard',
       'IsActiveMember', 'EstimatedSalary', 'Exited'],
      dtype='object')
```

In [ ]:
```
df = df.drop(columns = ['RowNumber', 'CustomerId', 'Surname'])
df.head(3)
```

Out[41]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	

In [ ]:
```
df.Geography.nunique(),df.Geography.unique()
```

Out[42]:
```
(3, array(['France', 'Spain', 'Germany'], dtype=object))
```

In [ ]:
```
df.Gender.nunique(),df.Gender.unique()
```

Out[43]:
```
(2, array(['Female', 'Male'], dtype=object))
```

In [ ]:
```
geo_ohe = pd.get_dummies(data = df.Geography, prefix = 'Geography')
geo_ohe.head()
```

Out[44]:

	Geography_France	Geography_Germany	Geography_Spain
0	1	0	0
1	0	0	1
2	1	0	0
3	1	0	0
4	0	0	1

In [ ]:
```
df['Gender_encoded'] = df.Gender.replace({'Female':0,'Male':1})
df.head()
```

Out[45]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

In [ ]:
```
final_df = pd.concat((df,geo_ohe), axis = 'columns')
final_df.head()
```

Out[46]:

	CreditScore	Geography	Gender	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActive
0	619	France	Female	42	2	0.00	1	1	
1	608	Spain	Female	41	1	83807.86	1	0	
2	502	France	Female	42	8	159660.80	3	1	
3	699	France	Female	39	1	0.00	2	0	
4	850	Spain	Female	43	2	125510.82	1	1	

In [ ]:
```
final_df = final_df.drop(columns = ['Geography','Gender'])
final_df.head()
```

Out[47]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsActiveMember	EstimatedSa
0	619	42	2	0.00	1	1	1	101348
1	608	41	1	83807.86	1	0	1	112542
2	502	42	8	159660.80	3	1	0	113931
3	699	39	1	0.00	2	0	0	93826
4	850	43	2	125510.82	1	1	1	79084

In [ ]:
```
final_df.describe()
```

Out[48]:

	CreditScore	Age	Tenure	Balance	NumOfProducts	HasCrCard	IsA
count	10000.000000	10000.000000	10000.000000	10000.000000	10000.000000	10000.00000	
mean	650.528800	38.921800	5.012800	76485.889288	1.530200	0.70550	
std	96.653299	10.487806	2.892174	62397.405202	0.581654	0.45584	
min	350.000000	18.000000	0.000000	0.000000	1.000000	0.00000	
25%	584.000000	32.000000	3.000000	0.000000	1.000000	0.00000	
50%	652.000000	37.000000	5.000000	97198.540000	1.000000	1.00000	
75%	718.000000	44.000000	7.000000	127644.240000	2.000000	1.00000	
max	850.000000	92.000000	10.000000	250898.090000	4.000000	1.00000	

In [ ]:
```
train, test = train_test_split(final_df, test_size = 0.2, random_state = 64)

train.shape, test.shape
```

Out[49]:
```
((8000, 13), (2000, 13))
```

In [ ]:
```
train_scaler, test_scaler = MinMaxScaler(), MinMaxScaler()

train_scaled = train_scaler.fit_transform(train)
test_scaled = test_scaler.fit_transform(test)
```

In [ ]:
```
train_scaled = pd.DataFrame(train_scaled, columns = train.columns)
test_scaled = pd.DataFrame(test_scaled, columns = test.columns)
```

In [ ]:
```
train_x, train_y = train_scaled.drop(columns = 'Exited'), train_scaled.Exited
test_x, test_y = test_scaled.drop(columns = 'Exited'), test_scaled.Exited
```

In [ ]:
```
train_x.shape
```

Out[53]:
```
(8000, 12)
```

Creating the NN

Model building

- Generating the `Sequential container`
- Iteratively creating `new layers` with the specs like `activation functions, kernel_initializers, number of neurons`
- Adding these `layers` to the `container`

`Once the model is built, it is compiled`

Model compiling

- Setting up the `Optimization function`
- Setting up the `Loss function`
- Setting up the `metrics`

In [ ]:
```
import tensorflow as tf
```

Instantiating ELU

- elu = tf.keras.layers.ELU(alpha = 0.95)

Adding layers

1.

- model = tf.keras.Sequential()

- input_layer = tf.keras.layers.Input(shape = (12,))
- hidden_layer_1 = tf.keras.layers.Dense(units = 6,activation = 'relu',kernel_initializer = 'he_uniform')
- hidden_layer_2 = tf.keras.layers.Dense(units = 8,activation = elu,kernel_initializer = 'he_uniform')
- output_layer = tf.keras.layers.Dense(units = 1, activation = 'sigmoid', kernel_initializer = 'gorot_uniform')
- model.add([input_layer,hidden_layer_1, hidden_layer_2, output_layer])

OR

2.

- model = tf.keras.Sequential()
- input_layer = tf.keras.layers.Input(shape = (12,))
- hidden_layer_1 = tf.keras.layers.Dense(units = 6,activation = 'relu',kernel_initializer = 'he_uniform')
- hidden_layer_2 = tf.keras.layers.Dense(units = 8,activation = elu,kernel_initializer = 'he_uniform')
- output_layer = tf.keras.layers.Dense(units = 1, activation = 'sigmoid', kernel_initializer = 'gorot_uniform')
- model.add(input_layer)
- model.add(hidden_layer_1)
- model.add(hidden_layer_2)
- model.add(output_layer)

OR

3.

- model = tf.keras.Sequential()
- model.add(tf.keras.layers.Input(shape = (12,)))
- model.add(tf.keras.layers.Dense(units = 6,activation = 'relu',kernel_initializer = 'he_uniform'))
- model.add(tf.keras.layers.Dense(units = 8,activation = elu,kernel_initializer = 'he_uniform'))
- model.add(tf.keras.layers.Dense(units = 1, activation = 'sigmoid', kernel_initializer = 'gorot_uniform'))

OR

4.

- model = tf.keras.Sequential(layers = [tf.keras.layers.Input(shape = (12,)),tf.keras.layers.Dense(units = 6,activation = 'relu',kernel_initializer = 'he_uniform'),tf.keras.layers.Dense(units = 8,activation = elu,kernel_initializer =

'he_uniform'),tf.keras.layers.Dense(units = 1, activation = 'sigmoid', kernel_initializer = 'gorot_uniform')])

In [ ]:
```
# Instantiating ELU
elu = tf.keras.layers.ELU(alpha = 0.95)

# Instantiating the Sequential model
model = tf.keras.Sequential()

# Adding the input layer
model.add(tf.keras.layers.Input(shape = (12,)))

# Adding the hidden layers
model.add(tf.keras.layers.Dense(units = 6,
                                activation = 'relu',
                                kernel_initializer = 'he_uniform'))
model.add(tf.keras.layers.Dense(units = 8,
                                activation = elu,
                                kernel_initializer = 'he_uniform'))

# Adding the output layer
model.add(tf.keras.layers.Dense(units = 1,
                                activation = 'sigmoid',
                                kernel_initializer = 'glorot_uniform'))
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 6)                 78

 dense_4 (Dense)             (None, 8)                 56

 dense_5 (Dense)             (None, 1)                 9

=================================================================
Total params: 143
Trainable params: 143
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
# Compiling the model
model.compile(optimizer = 'Adam',
              loss = 'binary_crossentropy',
              metrics = ['accuracy','Recall'])
```

In [ ]:
```
# # Training the model
# model.fit(x = train_x,
#           y = train_y,
#           batch_size = 256,
#           epochs = 10,
#           validation_data = (test_x, test_y),
#           use_multiprocessing = True,
#           workers = 20)
```

In [ ]:
```
# Training the model and saving the training logs
history_object = model.fit(x = train_x,
                           y = train_y,
                           batch_size = 256,
                           epochs = 100,
                           validation_data = (test_x, test_y),
                           use_multiprocessing = True,
                           workers = 20)
```

In [ ]:
```
# Playing with the history object
dir(history_object)
```

In [ ]:
```
hist = history_object.history
hist.keys()
```

Out[61]:
```
dict_keys(['loss', 'accuracy', 'recall', 'val_loss', 'val_accuracy', 'val_recal
l'])
```

In [ ]:
```
def history_visualizer(history_obj, keyword):
  epochs = history_obj.epoch
  hist = history_obj.history

  train_key = keyword
  val_key = f'val_{keyword}'

  train_data = hist.get(train_key)
  val_data = hist.get(val_key)

  plt.figure(figsize = (15,8))
  sns.lineplot(x = epochs, y = train_data)
  sns.lineplot(x = epochs, y = val_data)
  plt.legend(labels = [keyword.title(), val_key.title()])

  plt.show()
```

In [ ]:
```
history_visualizer(history_object,'loss')
```

In [ ]:
```
history_visualizer(history_object,'accuracy')
```

In [ ]:
```
history_visualizer(history_object,'recall')
```

In [ ]:
```
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 6)                 78

 dense_4 (Dense)             (None, 8)                 56

 dense_5 (Dense)             (None, 1)                 9

=================================================================
Total params: 143
Trainable params: 143
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```
[i for i in dir(model) if i.lower().__contains__('conf')]
```

In [ ]:
```
model.get_config()
```

Saving and loading the models

In [ ]:
```
# Saving the trained models
model_weights_path = 'TrainedModels'
os.makedirs(model_weights_path, exist_ok = True)

model.save(os.path.join(model_weights_path,'base_model.tf'),save_format = 'tf')
```

```
WARNING:absl:Found untraced functions such as elu_1_layer_call_fn, elu_1_layer_
call_and_return_conditional_losses while saving (showing 2 of 2). These functio
ns will not be directly callable after loading.
```

In [ ]:
```
# Loading the saved models
loaded_model = tf.keras.models.load_model(os.path.join(model_weights_path,'base_m
```

In [ ]:
```
loaded_model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 6)                 78

 dense_4 (Dense)             (None, 8)                 56

 dense_5 (Dense)             (None, 1)                 9

=================================================================
Total params: 143
Trainable params: 143
Non-trainable params: 0
_________________________________________________________________
```

tensorflow calbacks

In [ ]:
```
dir(tf.keras.callbacks)
```

In [ ]:
```
class MyCalbacks(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epochs, logs = {}):
    if (logs.get('accuracy')>0.8) & (logs.get('val_accuracy')>0.8):
      self.model.stop_training = True
      print()
      print('Condition Satisfied !')
      print()
```

In [ ]:
```
conditional_callback = MyCalbacks()
```

In [ ]:
```
# Building the model

# Instantiating ELU
elu = tf.keras.layers.ELU(alpha = 0.95)

# Instantiating the Sequential model
model_c = tf.keras.Sequential()

# Adding the input layer
model_c.add(tf.keras.layers.Input(shape = (12,)))

# Adding the hidden layers
model_c.add(tf.keras.layers.Dense(units = 6,
                                  activation = 'relu',
                                  kernel_initializer = 'he_uniform'))
model_c.add(tf.keras.layers.Dense(units = 8,
                                  activation = elu,
                                  kernel_initializer = 'he_uniform'))

# Adding the output layer
model_c.add(tf.keras.layers.Dense(units = 1,
                                  activation = 'sigmoid',
                                  kernel_initializer = 'glorot_uniform'))


# Compiling the model
model_c.compile(optimizer = 'Adam',
                loss = 'binary_crossentropy',
                metrics = ['accuracy','Precision','Recall'])
```

In [ ]:
```
# Training the model and saving the training logs
history_object_c = model_c.fit(x = train_x,
                               y = train_y,
                               batch_size = 256,
                               epochs = 100,
                               validation_data = (test_x, test_y),
                               use_multiprocessing = True,
                               workers = 20,
                               callbacks = [conditional_callback])
```

```
Epoch 1/100
32/32 [==============================] - 0s 10ms/step - loss: 0.6020 - accura
cy: 0.7924 - precision: 0.3261 - recall: 0.0091 - val_loss: 0.5801 - val_accu
racy: 0.8030 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 2/100
32/32 [==============================] - 0s 8ms/step - loss: 0.5696 - accurac
y: 0.7940 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.5471 - v
al_accuracy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 3/100
32/32 [==============================] - 0s 11ms/step - loss: 0.5391 - accura
cy: 0.7941 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.5151 -
val_accuracy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 4/100
32/32 [==============================] - 0s 10ms/step - loss: 0.5122 - accura
cy: 0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4904 -
val_accuracy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 5/100
32/32 [==============================] - 0s 9ms/step - loss: 0.4962 - accurac
y: 0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4789 - v
al_accuracy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 6/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4908 - accurac
y: 0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4741 - v
al_accuracy: 0.8045 - val_precision: 0.6667 - val_recall: 0.0051
Epoch 7/100
32/32 [==============================] - 0s 6ms/step - loss: 0.4873 - accurac
y: 0.7949 - precision: 0.6250 - recall: 0.0061 - val_loss: 0.4710 - val_accur
acy: 0.8060 - val_precision: 0.6429 - val_recall: 0.0230
Epoch 8/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4846 - accurac
y: 0.7959 - precision: 0.6200 - recall: 0.0188 - val_loss: 0.4680 - val_accur
acy: 0.8080 - val_precision: 0.6538 - val_recall: 0.0434
Epoch 9/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4821 - accurac
y: 0.7969 - precision: 0.6351 - recall: 0.0286 - val_loss: 0.4657 - val_accur
acy: 0.8060 - val_precision: 0.5588 - val_recall: 0.0485
Epoch 10/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4798 - accurac
y: 0.7972 - precision: 0.6139 - recall: 0.0377 - val_loss: 0.4641 - val_accur
acy: 0.8075 - val_precision: 0.5854 - val_recall: 0.0612
Epoch 11/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4778 - accurac
y: 0.7983 - precision: 0.6165 - recall: 0.0498 - val_loss: 0.4617 - val_accur
acy: 0.8070 - val_precision: 0.5714 - val_recall: 0.0612
Epoch 12/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4761 - accurac
```

```
y: 0.7981 - precision: 0.6071 - recall: 0.0517 - val_loss: 0.4601 - val_accur
acy: 0.8070 - val_precision: 0.5556 - val_recall: 0.0765
Epoch 13/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4745 - accurac
y: 0.7987 - precision: 0.6048 - recall: 0.0614 - val_loss: 0.4590 - val_accur
acy: 0.8060 - val_precision: 0.5357 - val_recall: 0.0765
Epoch 14/100
32/32 [==============================] - 0s 5ms/step - loss: 0.4732 - accurac
y: 0.7995 - precision: 0.6158 - recall: 0.0663 - val_loss: 0.4578 - val_accur
acy: 0.8050 - val_precision: 0.5156 - val_recall: 0.0842
Epoch 15/100
30/32 [===========================>..] - ETA: 0s - loss: 0.4716 - accuracy:
0.8010 - precision: 0.6646 - recall: 0.0689
Condition Satisfied !

32/32 [==============================] - 0s 5ms/step - loss: 0.4717 - accurac
y: 0.8016 - precision: 0.6686 - recall: 0.0699 - val_loss: 0.4566 - val_accur
acy: 0.8050 - val_precision: 0.5161 - val_recall: 0.0816
```

In [ ]:
```
history_object_c.epoch
```

Out[95]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

In [ ]:
```
history_visualizer(history_object_c,'loss')
```

In [ ]:
```
history_visualizer(history_object_c,'accuracy')
```

In [ ]:
```
history_visualizer(history_object_c,'precision')
```

In [ ]:
```
history_visualizer(history_object_c,'recall')
```

In [ ]:
```
# Early Stopper Callbacks
EarlyStopper = tf.keras.callbacks.EarlyStopping(monitor = 'accuracy', patience =

# Conditional callback
class MyCallbacks(tf.keras.callbacks.Callback):
  def on_epoch_end(self, epochs, logs = {}):
    if (logs.get('accuracy')>0.94) & (logs.get('val_accuracy')>0.94):
      self.model.stop_training = True
      print()
      print('Condition Satisfied !')
      print()

cond_callback2 = MyCallbacks()
```

In [ ]:
```
# Building the model

# Instantiating ELU
elu = tf.keras.layers.ELU(alpha = 0.95)

# Instantiating the Sequential model
model_es = tf.keras.Sequential()

# Adding the input layer
model_es.add(tf.keras.layers.Input(shape = (12,)))

# Adding the hidden layers
model_es.add(tf.keras.layers.Dense(units = 6,
                                   activation = 'relu',
                                   kernel_initializer = 'he_uniform'))
model_es.add(tf.keras.layers.Dense(units = 8,
                                   activation = elu,
                                   kernel_initializer = 'he_uniform'))

# Adding the output layer
model_es.add(tf.keras.layers.Dense(units = 1,
                                   activation = 'sigmoid',
                                   kernel_initializer = 'glorot_uniform'))


# Compiling the model
model_es.compile(optimizer = 'rmsprop',
                 loss = 'binary_crossentropy',
                 metrics = ['accuracy','Precision','Recall'])
```

In [ ]:
```
# Training the model and saving the training logs
history_object_es = model_es.fit(x = train_x,
                                 y = train_y,
                                 batch_size = 256,
                                 epochs = 500,
                                 validation_data = (test_x, test_y),
                                 use_multiprocessing = True,
                                 workers = 20,
                                 callbacks = [EarlyStopper,cond_callback2])
```

```
Epoch 1/500
32/32 [==============================] - 2s 28ms/step - loss: 0.5200 - accurac
y: 0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.5027 - val
_accuracy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 2/500
32/32 [==============================] - 0s 6ms/step - loss: 0.5025 - accuracy:
0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4862 - val_ac
curacy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 3/500
32/32 [==============================] - 0s 5ms/step - loss: 0.4899 - accuracy:
0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4740 - val_ac
curacy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 4/500
32/32 [==============================] - 0s 5ms/step - loss: 0.4819 - accuracy:
0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4664 - val_ac
curacy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 5/500
32/32 [==============================] - 0s 5ms/step - loss: 0.4774 - accuracy:
0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4628 - val_ac
curacy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
Epoch 6/500
32/32 [==============================] - 0s 5ms/step - loss: 0.4745 - accuracy:
0.7944 - precision: 0.0000e+00 - recall: 0.0000e+00 - val_loss: 0.4592 - val_ac
curacy: 0.8040 - val_precision: 0.0000e+00 - val_recall: 0.0000e+00
```

Hyper-Parameter tuning

In [ ]:
```
!pip install keras-tuner
```

```
Looking in indexes: https://pypi.org/simple, (https://pypi.org/simple,) http
s://us-python.pkg.dev/colab-wheels/public/simple/ (https://us-python.pkg.dev/co
lab-wheels/public/simple/)
Collecting keras-tuner
  Downloading keras_tuner-1.1.3-py3-none-any.whl (135 kB)
     |████████████████████████████████| 135 kB 15.0 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packag
es (from keras-tuner) (2.23.0)
Collecting kt-legacy
  Downloading kt_legacy-1.0.4-py3-none-any.whl (9.6 kB)
Requirement already satisfied: ipython in /usr/local/lib/python3.7/dist-package
s (from keras-tuner) (7.9.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages
 (from keras-tuner) (1.21.6)
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packa
ges (from keras-tuner) (21.3)
Requirement already satisfied: tensorboard in /usr/local/lib/python3.7/dist-pac
kages (from keras-tuner) (2.8.0)
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-package
s (from ipython->keras-tuner) (4.8.0)
Requirement already satisfied: traitlets>=4.2 in /usr/local/lib/python3.7/dist-
packages (from ipython->keras-tuner) (5.1.1)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packa
ges (from ipython->keras-tuner) (4.4.2)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-pac
kages (from ipython->keras-tuner) (0.7.5)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packag
es (from ipython->keras-tuner) (2.6.1)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dis
t-packages (from ipython->keras-tuner) (57.4.0)
Collecting jedi>=0.10
  Downloading jedi-0.18.1-py2.py3-none-any.whl (1.6 MB)
     |████████████████████████████████| 1.6 MB 62.3 MB/s
Requirement already satisfied: prompt-toolkit<2.1.0,>=2.0.0 in /usr/local/lib/p
ython3.7/dist-packages (from ipython->keras-tuner) (2.0.10)
Requirement already satisfied: backcall in /usr/local/lib/python3.7/dist-packag
es (from ipython->keras-tuner) (0.2.0)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in /usr/local/lib/python3.7/
dist-packages (from jedi>=0.10->ipython->keras-tuner) (0.8.3)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-package
s (from prompt-toolkit<2.1.0,>=2.0.0->ipython->keras-tuner) (0.2.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-pack
ages (from prompt-toolkit<2.1.0,>=2.0.0->ipython->keras-tuner) (1.15.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/pytho
n3.7/dist-packages (from packaging->keras-tuner) (3.0.9)
Requirement already satisfied: ptyprocess>=0.5 in /usr/local/lib/python3.7/dist
-packages (from pexpect->ipython->keras-tuner) (0.7.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/d
ist-packages (from requests->keras-tuner) (2022.6.15)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/
local/lib/python3.7/dist-packages (from requests->keras-tuner) (1.24.3)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/di
st-packages (from requests->keras-tuner) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-pa
```

```
ckages (from requests->keras-tuner) (2.10)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.7/dist-pac
kages (from tensorboard->keras-tuner) (0.37.1)
Requirement already satisfied: protobuf>=3.6.0 in /usr/local/lib/python3.7/dist
-packages (from tensorboard->keras-tuner) (3.17.3)
Requirement already satisfied: grpcio>=1.24.3 in /usr/local/lib/python3.7/dist-
packages (from tensorboard->keras-tuner) (1.47.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist
-packages (from tensorboard->keras-tuner) (3.4.1)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /usr/local/l
ib/python3.7/dist-packages (from tensorboard->keras-tuner) (0.4.6)
Requirement already satisfied: werkzeug>=0.11.15 in /usr/local/lib/python3.7/di
st-packages (from tensorboard->keras-tuner) (1.0.1)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /usr/lo
cal/lib/python3.7/dist-packages (from tensorboard->keras-tuner) (0.6.1)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.
7/dist-packages (from tensorboard->keras-tuner) (1.35.0)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/
python3.7/dist-packages (from tensorboard->keras-tuner) (1.8.1)
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.7/dist-pa
ckages (from tensorboard->keras-tuner) (1.2.0)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /usr/local/lib/python
3.7/dist-packages (from google-auth<3,>=1.6.3->tensorboard->keras-tuner) (4.2.
4)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-p
ackages (from google-auth<3,>=1.6.3->tensorboard->keras-tuner) (4.9)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.
7/dist-packages (from google-auth<3,>=1.6.3->tensorboard->keras-tuner) (0.2.8)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/pytho
n3.7/dist-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard->keras-t
uner) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in /usr/local/lib/python
3.7/dist-packages (from markdown>=2.6.8->tensorboard->keras-tuner) (4.12.0)
Requirement already satisfied: typing-extensions>=3.6.4 in /usr/local/lib/pytho
n3.7/dist-packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard-
>keras-tuner) (4.1.1)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packa
ges (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard->keras-tuner)
 (3.8.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /usr/local/lib/python3.
7/dist-packages (from pyasn1-modules>=0.2.1->google-auth<3,>=1.6.3->tensorboard
->keras-tuner) (0.4.8)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist
-packages (from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->ten
sorboard->keras-tuner) (3.2.0)
Installing collected packages: jedi, kt-legacy, keras-tuner
Successfully installed jedi-0.18.1 keras-tuner-1.1.3 kt-legacy-1.0.4
```

In [ ]:
```
from keras_tuner.tuners import RandomSearch
```

In [ ]:
```
# RandomSearch -> hypermodel
# hypermodel >> function that would be generating and compiling the entire model

# Datatypes for hyperparameter tuning
# Int > for tuning from a range of elements/numbers.  eg. range(5,15)
#------------syntax : hyp_key.Int('keyword specifying the purpose of this data',m
# Choice > for tuning from a list of elements.     eg. [he_uniform, he_normal, gl
#------------syntax : hyp_key.Choice('keyword specifying the purpose of this data
```

In [ ]:
```
```

Out[145]:
```
[0.9, 0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.0]
```

In [ ]:
```
def hypermodel(hyp_key):
  model_ = tf.keras.Sequential()

  model_.add(tf.keras.layers.Input(shape = (12,)))

  for i in range(hyp_key.Int('number of hidden layers', 1,7)):
    model_.add(tf.keras.layers.Dense(units = hyp_key.Int(f'number of neurons in h
                                     activation = hyp_key.Choice(f'activation fun
                                     kernel_initializer = hyp_key.Choice(f'kernel

  model_.add(tf.keras.layers.Dense(units = 1,
                                   activation = 'sigmoid',
                                   kernel_initializer = hyp_key.Choice(f'kernel_i

  model_.compile(optimizer = hyp_key.Choice('Optimizer function',['Adam','rmsprop
                 loss = 'binary_crossentropy',
                 metrics = ['accuracy', 'Precision', 'Recall'])

  return model_
```

In [ ]:
```
# Generating the tuner object
tuner_obj = RandomSearch(hypermodel = hypermodel,
                         objective = 'accuracy',
                         max_trials = 20,
                         seed = 64)
```

In [ ]:
```
# validating the search space summary of the tuner object
tuner_obj.search_space_summary()
```

```
Search space summary
Default search space size: 6
number of hidden layers (Int)
{'default': None, 'conditions': [], 'min_value': 1, 'max_value': 7, 'step': 1,
'sampling': None}
number of neurons in hidden layer # 0 (Int)
{'default': None, 'conditions': [], 'min_value': 4, 'max_value': 13, 'step': 1,
'sampling': None}
activation function for hidden layer # 0 (Choice)
{'default': 'relu', 'conditions': [], 'values': ['relu', 'elu', 'sigmoid'], 'or
dered': False}
kernel_initializer for hidden layer # 0 (Choice)
{'default': 'he_normal', 'conditions': [], 'values': ['he_normal', 'he_unifor
m', 'glorot_normal', 'glorot_uniform'], 'ordered': False}
kernel_initializer for output layer (Choice)
{'default': 'he_normal', 'conditions': [], 'values': ['he_normal', 'he_unifor
m', 'glorot_normal', 'glorot_uniform'], 'ordered': False}
Optimizer function (Choice)
{'default': 'Adam', 'conditions': [], 'values': ['Adam', 'rmsprop'], 'ordered':
False}
```

In [ ]:
```
# Finding the best model
tuner_obj.search(train_x,
                 train_y,
                 batch_size = 64)
```

```
Trial 20 Complete [00h 00m 02s]
accuracy: 0.784375011920929

Best accuracy So Far: 0.7943750023841858
Total elapsed time: 00h 00m 32s
```

In [ ]:
```
dir(tuner_obj)
```

Out[171]:
```
['__class__',
 '__delattr__',
 '__dict__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__init__',
 '__init_subclass__',
 '__le__',
 '__lt__',
 '__module__',
 '__ne__',
 '__new__',
 '__reduce__',
 '__reduce_ex__',
 '__repr__',
 '__setattr__',
 '__sizeof__',
 '__str__',
 '__subclasshook__',
 '__weakref__',
 '_build_and_fit_model',
 '_build_hypermodel',
 '_configure_tensorboard_dir',
 '_deepcopy_callbacks',
 '_display',
 '_get_checkpoint_fname',
 '_get_tensorboard_dir',
 '_get_tuner_fname',
 '_override_compile_args',
 '_populate_initial_space',
 '_save_n_checkpoints',
 '_try_build',
 'directory',
 'distribution_strategy',
 'executions_per_trial',
 'get_best_hyperparameters',
 'get_best_models',
 'get_state',
 'get_trial_dir',
 'hypermodel',
 'load_model',
 'logger',
 'loss',
 'max_model_size',
 'metrics',
 'on_batch_begin',
 'on_batch_end',
 'on_epoch_begin',
```

```
 'on_epoch_end',
 'on_search_begin',
 'on_search_end',
 'on_trial_begin',
 'on_trial_end',
 'optimizer',
 'oracle',
 'pre_create_trial',
 'project_dir',
 'project_name',
 'reload',
 'remaining_trials',
 'results_summary',
 'run_trial',
 'save',
 'save_model',
 'search',
 'search_space_summary',
 'seed',
 'set_state',
 'tuner_id']
```

In [ ]:
```
tuner_obj.results_summary()
```

In [ ]:
```
best_5_models = tuner_obj.get_best_models(5)
```

In [ ]:
```
model_new = best_5_models[0]
```

In [ ]:
```
model_new.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 91

 dense_1 (Dense)             (None, 4)                 32

 dense_2 (Dense)             (None, 7)                 35

 dense_3 (Dense)             (None, 11)                88

 dense_4 (Dense)             (None, 1)                 12

=================================================================
Total params: 258
Trainable params: 258
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```

```