Using NN & ML to analyse which model will deliver better forecast of sales opps.

```
# Importing the required libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

from sklearn.compose import make_column_transformer
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error

#uploading the datasets

from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in

```
Saving dataset_som_training_30d_20220130.csv to dataset_som_training_30d_20220130.csv
```

```
import io
#dataset = pd.read_csv(io.BytesIO(uploaded['CellDNA.csv']),header=None)

#Reading in Sales Opportunities Report data
dataset = pd.read_csv(io.BytesIO(uploaded['dataset_som_training_30d_20220130.csv']))

dataset.shape
```

```
(8546, 17)
```

```
print(dataset.iloc[0,:])
```

```
Instance_ID                    13191543922
Opportunity ID                      131915
Report date                          43921
Last RG reached          Quotation approved
Business Type              Customer Service
```

```
Chance of Realization                  0.4
Chance for Bühler                      0.6
Exp Sales Vol in CHF               11593.2
PlanDate: Order Rel.                 43917
Start Date                           43577
Latitude                           19.2925
Longitude                         -99.6569
Opps_last_5y                            11
Opps_empl_5y                             0
new_BUs                                 CM
Updated_Type                     Undefined
Final_Status           Lost or Cancelled
Name: 0, dtype: object
```

Formating data for model predictions

```
# Assign X and y
X = dataset.iloc[:, 2:-1].values
y = dataset.iloc[:,-1].values

print(X[0:2,:])
X.shape
```

```
[[43921 'Quotation approved' 'Customer Service' 0.4 0.6 11593.2 43917
  43577 19.292545 -99.6569007 11.0 0.0 'CM' 'Undefined']
 [43921 'RG2: Ready to Quote' 'Customer Service' 0.4 0.2 9.66 43921 43536
  19.2667063 -99.1269596 9.0 0.0 'CM' 'Binding Quotation']]
(8546, 14)
```

```
#Dealing with categorical variables -> Last RG Reached
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import OneHotEncoder
col_trans = make_column_transformer((OneHotEncoder(), [1]), remainder='passthrough')
X = col_trans.fit_transform(X)

X = np.delete(X,0,1)

#Dealing with categorical variables Opport. Group
col_trans = make_column_transformer((OneHotEncoder(), [5]), remainder='passthrough')
X = col_trans.fit_transform(X)

X = np.delete(X,0,1)

#Dealing with categorical variables -> new_BUs
col_trans = make_column_transformer((OneHotEncoder(), [-2]), remainder='passthrough')
X = col_trans.fit_transform(X)

X = np.delete(X,0,1)
```

```
#Dealing with categorical variables -> Quotation type: binging, price indication
col_trans = make_column_transformer((OneHotEncoder(), [-1]), remainder='passthrough')
X = col_trans.fit_transform(X)

X = np.delete(X,0,1)


print(X[0:2,:])

X.shape
```

```
[[0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0
  0.0 43921 0.4 0.6 11593.2 43917 43577 19.292545 -99.6569007 11.0 0.0]
 [0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0
  0.0 43921 0.4 0.2 9.66 43921 43536 19.2667063 -99.1269596 9.0 0.0]]
(8546, 29)
```

```
for p in np.unique(y):
    print("Count of", p,"is ", (sum(y==p)))

#Dependent variable ->y
from sklearn.preprocessing import LabelEncoder
labelencoder_y = LabelEncoder()
y = labelencoder_y.fit_transform(y)

print()

for p in np.unique(y):
    print("Count of", p,"is ", (sum(y==p)))
```

```
Count of Active is  5417
Count of Lost or Cancelled is  2292
Count of Won is  837

Count of 0 is  5417
Count of 1 is  2292
Count of 2 is  837
```

```
# Splitting the data into Training Set and Test Set, stratified on 'quality'
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0, stra


#Normalizing the features
from sklearn.preprocessing import StandardScaler
```

```
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)

#Preparing Y_Train
from tensorflow.keras import utils
y_train_cat = utils.to_categorical(y_train, 3)
y_test_cat = utils.to_categorical(y_test, 3)

print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_train_cat.shape)
```

```
(6836, 29)
(1710, 29)
(6836,)
(6836, 3)
```

```
print(y_train_cat[1:10,:])
```

```
[[0. 1. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]
 [1. 0. 0.]]
```

```
# mount gdrive with this code
from google.colab import drive
drive.mount('/content/drive')
#below where the file is in gdrive, change with your
data_path = "/content/drive/My Drive/Machine_learning/Sales_forecast"
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mour
```

```
# import packages
from tensorflow import keras
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Activation, Dense, Dropout, Flatten, MaxPooling2D, BatchN
from tensorflow.keras.layers import Conv2D
from keras.layers import Convolution2D
from keras.layers import BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
from sklearn.metrics import confusion_matrix
```

```
from math import ceil
import numpy as np

np.random.seed(200)

visible = Input(shape=(29,))     # input layer
#visible = Dense(units=27, input_dim=27, activation = 'relu')
x1 = Dense(units = 27, activation = 'relu', name = 'h1') (visible)
x2 = Dense(units = 25, activation = 'relu', name = 'h2') (x1)
d1 = Dropout(0.15, name = 'do1') (x2)
x3 = Dense(units = 20, activation = 'relu', name = 'h3') (d1)
d2 = Dropout(0.15, name = 'do2') (x3)
x4 = Dense(units = 18, activation = 'relu', name = 'h4') (d2)
d3 = Dropout(0.1,  name = 'do3') (x4)
x5 = Dense(units = 15, activation = 'relu', name = 'h5') (d3)
x6 = Dense(units = 12, activation = 'relu', name = 'h6') (x5)
x7 = Dense(units = 8, activation = 'relu', name = 'h7') (x6)
x8 = Dense(units = 5, activation = 'relu', name = 'h8') (x7)
output = Dense(units = 3, activation = 'softmax') (x8)                  # output layer

model = Model(inputs = visible, outputs = output) #here we set the model, informing the input

model.summary()
```

```
Model: "model_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_7 (InputLayer)        [(None, 29)]              0

 h1 (Dense)                  (None, 27)                810

 h2 (Dense)                  (None, 25)                700

 do1 (Dropout)               (None, 25)                0

 h3 (Dense)                  (None, 20)                520

 do2 (Dropout)               (None, 20)                0

 h4 (Dense)                  (None, 18)                378

 do3 (Dropout)               (None, 18)                0

 h5 (Dense)                  (None, 15)                285

 h6 (Dense)                  (None, 12)                192

 h7 (Dense)                  (None, 8)                 104

 h8 (Dense)                  (None, 5)                 45

 dense_6 (Dense)             (None, 3)                 18
```

```
=================================================================
Total params: 3,052
Trainable params: 3,052
Non-trainable params: 0
_________________________________________________________________
```

```
#!ls /content/drive/MyDrive/Machine_learning/Sales_forecast/


MyEpochs = 60
MyBatches = 10

#opt = keras.optimizers.Adam(learning_rate=0.01)
opt = keras.optimizers.SGD(learning_rate=0.001, momentum=0.8) #learning_rate was 0.001


model.compile(loss=keras.losses.categorical_crossentropy,
              optimizer=opt,
              metrics=['accuracy'])

history = model.fit(X_train, y_train_cat,
                        validation_data=(X_test, y_test_cat),
                        batch_size = MyBatches,
                        epochs = MyEpochs,
                        verbose = 2)

# Save the entire model as a SavedModel.
model.save('/content/drive/My Drive/Machine_learning/Sales_forecast/saved_model/SF_Draft8_202
```

```
Epoch 34/60
684/684 - 2s - loss: 0.7841 - accuracy: 0.6448 - val_loss: 0.7958 - val_accuracy:
Epoch 35/60
684/684 - 2s - loss: 0.7819 - accuracy: 0.6448 - val_loss: 0.7962 - val_accuracy:
Epoch 36/60
684/684 - 2s - loss: 0.7874 - accuracy: 0.6448 - val_loss: 0.7970 - val_accuracy:
Epoch 37/60
684/684 - 2s - loss: 0.7825 - accuracy: 0.6454 - val_loss: 0.7960 - val_accuracy:
Epoch 38/60
684/684 - 2s - loss: 0.7817 - accuracy: 0.6458 - val_loss: 0.7948 - val_accuracy:
Epoch 39/60
684/684 - 2s - loss: 0.7813 - accuracy: 0.6453 - val_loss: 0.7955 - val_accuracy:
Epoch 40/60
684/684 - 2s - loss: 0.7797 - accuracy: 0.6441 - val_loss: 0.7954 - val_accuracy:
Epoch 41/60
684/684 - 2s - loss: 0.7826 - accuracy: 0.6451 - val_loss: 0.7953 - val_accuracy:
Epoch 42/60
684/684 - 2s - loss: 0.7818 - accuracy: 0.6435 - val_loss: 0.7951 - val_accuracy:
Epoch 43/60
684/684 - 2s - loss: 0.7781 - accuracy: 0.6460 - val_loss: 0.7942 - val_accuracy:
Epoch 44/60
684/684 - 2s - loss: 0.7785 - accuracy: 0.6451 - val_loss: 0.7939 - val_accuracy:
Epoch 45/60
```

```
684/684 - 2s - loss: 0.7783 - accuracy: 0.6456 - val_loss: 0.7947 - val_accuracy:
Epoch 46/60
684/684 - 2s - loss: 0.7794 - accuracy: 0.6442 - val_loss: 0.7960 - val_accuracy:
Epoch 47/60
684/684 - 2s - loss: 0.7765 - accuracy: 0.6432 - val_loss: 0.7946 - val_accuracy:
Epoch 48/60
684/684 - 2s - loss: 0.7767 - accuracy: 0.6480 - val_loss: 0.7940 - val_accuracy:
Epoch 49/60
684/684 - 2s - loss: 0.7751 - accuracy: 0.6457 - val_loss: 0.7942 - val_accuracy:
Epoch 50/60
684/684 - 2s - loss: 0.7740 - accuracy: 0.6470 - val_loss: 0.7937 - val_accuracy:
Epoch 51/60
684/684 - 2s - loss: 0.7753 - accuracy: 0.6464 - val_loss: 0.7956 - val_accuracy:
Epoch 52/60
684/684 - 2s - loss: 0.7765 - accuracy: 0.6463 - val_loss: 0.7934 - val_accuracy:
Epoch 53/60
684/684 - 2s - loss: 0.7730 - accuracy: 0.6441 - val_loss: 0.7965 - val_accuracy:
Epoch 54/60
684/684 - 2s - loss: 0.7733 - accuracy: 0.6457 - val_loss: 0.7952 - val_accuracy:
Epoch 55/60
684/684 - 2s - loss: 0.7739 - accuracy: 0.6457 - val_loss: 0.7940 - val_accuracy:
Epoch 56/60
684/684 - 2s - loss: 0.7738 - accuracy: 0.6456 - val_loss: 0.7933 - val_accuracy:
Epoch 57/60
684/684 - 2s - loss: 0.7694 - accuracy: 0.6450 - val_loss: 0.7940 - val_accuracy:
Epoch 58/60
684/684 - 2s - loss: 0.7721 - accuracy: 0.6458 - val_loss: 0.7932 - val_accuracy:
Epoch 59/60
684/684 - 2s - loss: 0.7726 - accuracy: 0.6456 - val_loss: 0.7941 - val_accuracy:
Epoch 60/60
684/684 - 2s - loss: 0.7707 - accuracy: 0.6458 - val_loss: 0.7971 - val_accuracy:
INFO:tensorflow:Assets written to: /content/drive/My Drive/Machine_learning/Sales
```

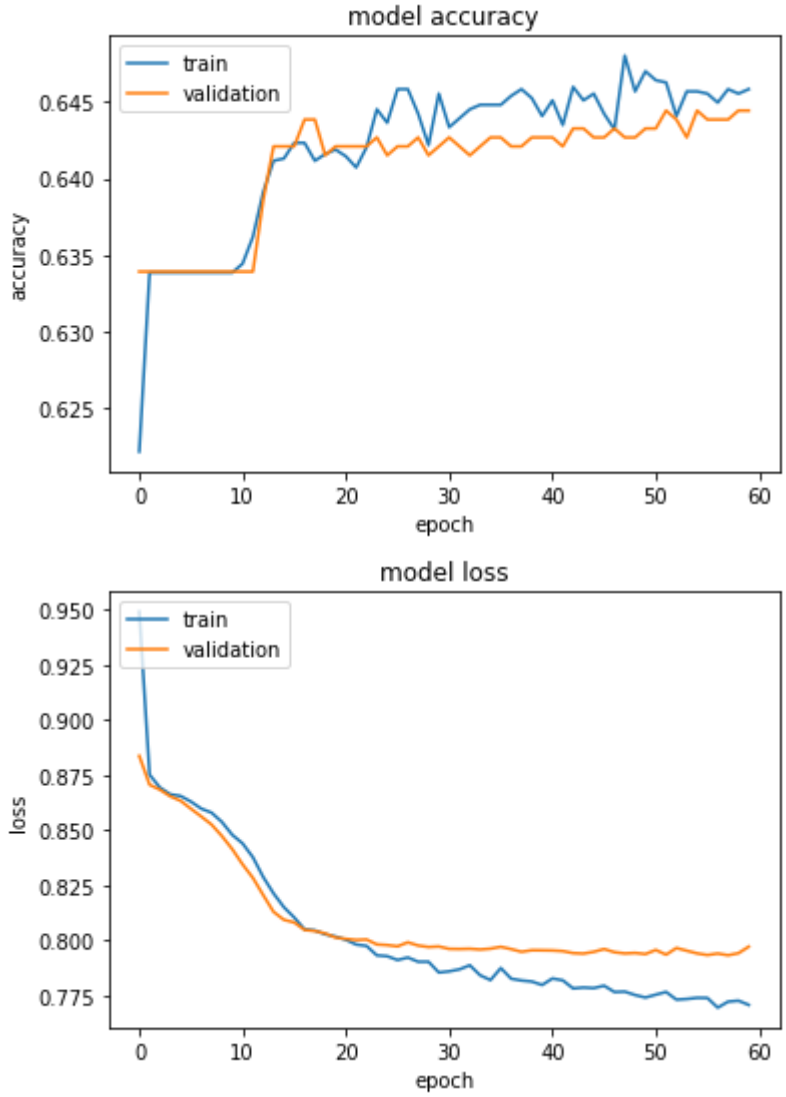
```
!ls /content/drive/MyDrive/Machine_learning/Sales_forecast/saved_model/
```

```
SFDraft620211211  SFDraft720211211  SF_Draft8_20211211  SF_Draft8_20220130
```

```
import matplotlib.pyplot as plt
print(history.history.keys())
#history of accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
#history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper left')
plt.show()
```

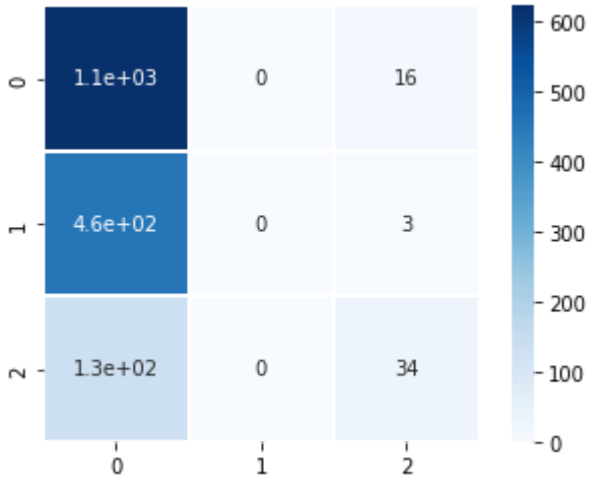
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```




```
# Checking the model with test data
from sklearn.metrics import confusion_matrix
from sklearn.metrics import balanced_accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score

yhat = model.predict(X_test)

y_pred = yhat.argmax(axis=-1) #getting the labels of the yhat predicted with X_test
print(np.unique(y_pred))

acc_score = accuracy_score(y_test, y_pred)
```

```
print("\n", "Accuracy: ", format(format(acc_score,'.3f')) )
print("\n", 'This is the confusion matrix between Y (train) and yhat (prediction)',"\n")


import seaborn as sns

CFM = confusion_matrix(y_test , y_pred)
matrix=sns.heatmap(CFM, linewidths=1,vmax=622,
                   square=True, cmap="Blues",annot=True)
#print(confusion_matrix(y_test, y_classes))
print("\n", "Classification Report", "\n", classification_report(y_test, y_pred))
```

```
     [0 2]

      Accuracy:  0.644

      This is the confusion matrix between Y (train) and yhat (prediction)


      Classification Report
                   precision    recall  f1-score   support

                0       0.64      0.99      0.78      1084
                1       0.00      0.00      0.00       459
                2       0.64      0.20      0.31       167

           accuracy                           0.64      1710
          macro avg       0.43      0.40      0.36      1710
       weighted avg       0.47      0.64      0.52      1710

     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
       _warn_prf(average, modifier, msg_start, len(result))
     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
       _warn_prf(average, modifier, msg_start, len(result))
     /usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1318: Undefine
       _warn_prf(average, modifier, msg_start, len(result))
```




```
# create a newmodel by copying from existing model
```

```
# for reference only, these are my training tests X_train, X_test, y_train, y_test

new_mid_layer_model_2 = Model(inputs=model.input,
                              outputs=model.get_layer('h8').output)

Z_Code_train = new_mid_layer_model_2.predict(X_train)
Z_Code_test = new_mid_layer_model_2.predict(X_test)


# Setting X and y -> train & test

X_train = Z_Code_train
X_test = Z_Code_test




#Naive Bayes
#Fitting Classifier to Training Set. Create a classifier object here and call it classifierOb
from sklearn.naive_bayes import GaussianNB
classifierObj = GaussianNB()
classifierObj.fit(X_train , y_train)

#Making predictions on the Test Set
y_pred = classifierObj.predict(X_test)


#Evaluating the combination of Deep Learning with ML
import seaborn as sns

CFM = confusion_matrix(y_test , y_pred)
matrix=sns.heatmap(CFM, linewidths=1,vmax=622,
                   square=True, cmap="Blues",annot=True)
#print("\n", "Accuracy: " + str(format(acc_score,'.3f')))
#print("\n", "CFM: \n", confusion_matrix(y_test , y_pred))
print("\n", "Classification report: \n", classification_report(y_test , y_pred))
```

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

           0       0.50      0.00      0.01      1084
           1       0.29      0.90      0.44       459
           2       0.35      0.55      0.43       167

    accuracy                           0.30      1710
   macro avg       0.38      0.49      0.29      1710
weighted avg       0.43      0.30      0.16      1710
```

Importing SOM Report: Jan-07-2022

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in

```
Saving 20220107_som.csv to 20220107_som (3).csv
```

```
import io
#dataset = pd.read_csv(io.BytesIO(uploaded['CellDNA.csv']),header=None)

#Reading in Sales Opportunities Report data
dataset_real = pd.read_csv(io.BytesIO(uploaded['20220107_som.csv']))

dataset_real.shape
```

```
(2762, 15)
```

```
display(dataset_real.iloc[1,:])
```

```
Opportunity ID                      175299
Report date                          44568
Last RG reached          Quotation approved
Business Type              Customer Service
Chance of Realization                   0.6
Chance for Bühler                       0.8
Exp. Sales Vol. in CHF              6807.75
PlanDate: Order Rel.                  44562
Start Date                            44312
Latitude                              43.42
Longitude                           -83.949
Opps_last_5y                             39
Opps_empl_5y                            112
new_BUs                                  HN
Quotation_Type                    Undefined
Name: 1, dtype: object
```