Technological Institute of the Philippines	**Quezon City - Computer Engineering**
Course Code:	CPE 019
Code Title:	Emerging Technologies in CpE 2
2nd Semester	AY 2024 - 2025

ASSIGNMENT 7.1	**Saving Models**
Name	Calvadores, Kelly Joseph
Section	CPE32S3
Date Performed:	Arpil 16, 2024
Date Submitted:	April 19, 2024
Instructor:	Engr. Roman M. Richard

Choose any dataset applicable to either a classification problem or a regression problem.

Resource:https://archive.ics.uci.edu/dataset/852/gender+gap+in+spanish+wp

```
import pandas as pd
import numpy as np
import tensorflow as tf

Data = pd.read_csv('data.csv');
Data.head()
```

```
     gender  C_api  C_man  E_NEds  E_Bpag        firstDay         lastDay  NEds  NDa
  0       1   male      1       2       2  20170527205915  20170721044501   543
  1       0 unknown     3       3       1  20110301072441  20170731213735  2764   23
  2       1   male      1       0       2  20060907204302  20140911191722    57   29
  3       1   male      1       1       2  20121003144916  20121208180528   104
  4       0 unknown     3       1       1  20070311125035  20141106121057   184   27
5 rows × 21 columns
```

Explain your datasets and the problem being addressed.

- The problem that is being adressed is that the inequality between 2 genders in the participation of contributing to the Spanish Wikapedia, this activity may potential mitiagation the factors the imbalance for editing practices and, researchers and practioners. The goal is to identify the challenges that may discourage the women from participating in the Wikapedia editing.

Show evidence that you can do the following:

Pre-Processing data

```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
```

```
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   gender         4746 non-null   int64  
 1   C_api          4746 non-null   object 
 2   C_man          4746 non-null   int64  
 3   E_NEds         4746 non-null   int64  
 4   E_Bpag         4746 non-null   int64  
 5   firstDay       4746 non-null   int64  
 6   lastDay        4746 non-null   int64  
 7   NEds           4746 non-null   int64  
 8   NDays          4746 non-null   int64  
 9   NActDays       4746 non-null   int64  
 10  NPages         4746 non-null   int64  
 11  NPcreated      4746 non-null   int64  
 12  pagesWomen     4746 non-null   int64  
 13  wikiprojWomen  4746 non-null   int64  
 14  ns_user        4746 non-null   int64  
 15  ns_wikipedia   4746 non-null   int64  
 16  ns_talk        4746 non-null   int64  
 17  ns_userTalk    4746 non-null   int64  
 18  ns_content     4746 non-null   int64  
 19  weightIJ       4746 non-null   float64
 20  NIJ            4746 non-null   int64  
dtypes: float64(1), int64(19), object(1)
memory usage: 778.8+ KB
```

```
from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
for i in Data:
  if Data[i].dtypes == 'object':
    Data[i] = LE.fit_transform(Data[i])
  else:
    pass
Data
```

	gender	C_api	C_man	E_NEds	E_Bpag	firstDay	lastDay	NEds	ND
0	1	1	1	2	2	20170527205915	20170721044501	543	
1	0	2	3	3	1	20110301072441	20170731213735	2764	2
2	1	1	1	0	2	20060907204302	20140911191722	57	2
3	1	1	1	1	2	20121003144916	20121208180528	104	
4	0	2	3	1	1	20070311125035	20141106121057	184	2
...	...	...	...	...	...	...	...	...	
4741	1	1	3	2	2	20120227100614	20170930073013	266	2
4742	0	2	3	3	1	20111108054659	20170906055641	1217	2
4743	2	2	2	1	2	20120405102902	20170302073010	122	1
4744	2	0	3	3	2	20091014131349	20161112122730	962	2
4745	1	2	1	2	0	20050901045004	20151022222845	284	3

4746 rows × 21 columns

```
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   gender         4746 non-null   int64  
 1   C_api          4746 non-null   int64  
 2   C_man          4746 non-null   int64  
 3   E_NEds         4746 non-null   int64  
 4   E_Bpag         4746 non-null   int64  
 5   firstDay       4746 non-null   int64  
 6   lastDay        4746 non-null   int64  
 7   NEds           4746 non-null   int64
 8   NDays          4746 non-null   int64
 9   NActDays       4746 non-null   int64
 10  NPages         4746 non-null   int64
 11  NPcreated      4746 non-null   int64
 12  pagesWomen     4746 non-null   int64
 13  wikiprojWomen  4746 non-null   int64
 14  ns_user        4746 non-null   int64
 15  ns_wikipedia   4746 non-null   int64
 16  ns_talk        4746 non-null   int64
 17  ns_userTalk    4746 non-null   int64
 18  ns_content     4746 non-null   int64
 19  weightIJ       4746 non-null   float64
 20  NIJ            4746 non-null   int64
dtypes: float64(1), int64(20)
memory usage: 778.8 KB
```

```
Data.describe()
```

	gender	C_api	C_man	E_NEds	E_Bpag	firstDay	
count	4746.000000	4746.000000	4746.000000	4746.000000	4746.000000	4.746000e+03	4
mean	0.737042	1.573746	2.082807	1.484197	1.646228	2.009942e+13	2
std	0.585355	0.566484	0.964978	1.099795	1.079263	3.516337e+10	1
min	0.000000	0.000000	1.000000	0.000000	0.000000	2.002011e+13	2
25%	0.000000	1.000000	1.000000	1.000000	1.000000	2.007042e+13	2
50%	1.000000	2.000000	3.000000	1.000000	2.000000	2.009121e+13	2
75%	1.000000	2.000000	3.000000	2.000000	3.000000	2.013040e+13	2
max	2.000000	2.000000	3.000000	3.000000	3.000000	2.017093e+13	2

8 rows × 21 columns

```
Data.isnull()
```

	gender	C_api	C_man	E_NEds	E_Bpag	firstDay	lastDay	NEds	NDays	NActDays
0	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
4741	False	False	False	False	False	False	False	False	False	False
4742	False	False	False	False	False	False	False	False	False	False
4743	False	False	False	False	False	False	False	False	False	False
4744	False	False	False	False	False	False	False	False	False	False
4745	False	False	False	False	False	False	False	False	False	False

4746 rows × 21 columns

Find and remove Outlier

```
Data.corr()
```

	gender	C_api	C_man	E_NEds	E_Bpag	firstDay	lastDay
gender	1.000000	-0.568169	-0.537139	-0.048359	0.172966	0.099542	0.009226
C_api	-0.568169	1.000000	0.071908	-0.081682	-0.116400	-0.090105	-0.058939
C_man	-0.537139	0.071908	1.000000	0.051772	-0.254762	0.019310	-0.000810
E_NEds	-0.048359	-0.081682	0.051772	1.000000	0.127476	-0.186356	0.259000
E_Bpag	0.172966	-0.116400	-0.254762	0.127476	1.000000	-0.126777	-0.025059
firstDay	0.099542	-0.090105	0.019310	-0.186356	-0.126777	1.000000	0.161636
lastDay	0.009226	-0.058939	-0.000810	0.259000	-0.025059	0.161636	1.000000
NEds	0.030629	-0.106641	0.039084	0.330999	0.085689	-0.099369	0.163663
NDays	-0.092550	0.056437	-0.014578	0.311284	0.108217	-0.875006	0.327911
NActDays	0.011996	-0.118119	0.045706	0.524455	0.120005	-0.199639	0.253744
NPages	0.033252	-0.096279	0.037013	0.260731	0.059046	-0.100343	0.135295
NPcreated	0.007375	-0.047954	-0.003065	0.181841	0.077025	-0.085885	0.084728
pagesWomen	0.046884	-0.078081	0.031177	0.106484	0.038336	-0.030236	0.063984
wikiprojWomen	0.052152	-0.066906	0.000454	0.030551	0.027241	-0.005189	0.022284
ns_user	0.027432	-0.078333	0.014529	0.298505	0.127708	-0.048975	0.120089
ns_wikipedia	0.050914	-0.098230	0.039088	0.174467	0.055520	-0.068091	0.085469
ns_talk	0.033298	-0.101005	0.041460	0.279854	0.086462	-0.113621	0.125636
ns_userTalk	0.050727	-0.111374	0.038999	0.225501	0.074981	-0.085746	0.104564
ns_content	0.023635	-0.093864	0.035462	0.318804	0.079052	-0.095595	0.159510
weightIJ	-0.061613	0.024002	0.109505	0.040102	-0.417940	0.034635	0.069976
NIJ	0.022187	-0.018834	-0.015265	0.115894	0.021281	-0.042541	0.062573

21 rows × 21 columns

```
CorrData = Data.corr()
TargCorr = CorrData['gender']
AbstarCor = TargCorr.abs()
LowCorrFeat = AbstarCor[AbstarCor <= 0.01].index.tolist()
print(LowCorrFeat)
```

```
['lastDay', 'NPcreated']
```

```
Data = Data.drop(columns = LowCorrFeat)
Data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 19 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   gender         4746 non-null   int64
 1   C_api          4746 non-null   int64
 2   C_man          4746 non-null   int64
 3   E_NEds         4746 non-null   int64
 4   E_Bpag         4746 non-null   int64
 5   firstDay       4746 non-null   int64
 6   NEds           4746 non-null   int64
 7   NDays          4746 non-null   int64
 8   NActDays       4746 non-null   int64
 9   NPages         4746 non-null   int64
 10  pagesWomen     4746 non-null   int64
 11  wikiprojWomen  4746 non-null   int64
 12  ns_user        4746 non-null   int64
 13  ns_wikipedia   4746 non-null   int64
 14  ns_talk        4746 non-null   int64
 15  ns_userTalk    4746 non-null   int64
 16  ns_content     4746 non-null   int64  
 17  weightIJ       4746 non-null   float64
 18  NIJ            4746 non-null   int64  
dtypes: float64(1), int64(18)
memory usage: 704.6 KB
```

Splitting Data

```
X = Data.drop(columns = 'gender')
Y = Data['gender']
```

Normalize the data

```
from sklearn.preprocessing import StandardScaler
Standard = StandardScaler()
Xnorm = Standard.fit_transform(X)
```

Splitting Training and Testing

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(Xnorm, Y, test_size = 0.001, random_state = 123)
```

```
import tensorflow as tf
from tensorflow.keras.utils import to_categorical

LE = LabelEncoder()
LE.fit(y_train)
LEy = LE.transform(y_train)
NY = to_categorical(LEy)
```

Creating and training Model

Create a Base Model

```
from tensorflow.keras.models import Sequential
from keras.layers import Dense

Model = Sequential()
Model.add(Dense(18, input_shape = (18,), activation = 'relu'))

Model.add(Dense(3, activation = 'softmax'))

Model.compile(loss = 'categorical_crossentropy', optimizer = 'SGD', metrics = ['accuracy'])
```

Training Model

```
Model.fit(X_train, NY, epochs = 300, batch_size = 5000, verbose = 0)
Results = Model.evaluate(X_train, NY, verbose = 0)
print("%s: %.2f%%" % (Model.metrics_names[1], Results[1]*100))
```

```
accuracy: 90.87%
```

Save a model and load the model in a JSON format

```
import os
from tensorflow.keras.models import  model_from_json

Model_json = Model.to_json()
with open("model.json", "w") as json_file:
    json_file.write(Model_json)
```

Load the json file

```
json_file = open('model.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
loaded_model = model_from_json(loaded_model_json)
```

Save a model in HDF5 format

```
Model.save_weights("/content/model.weights.h5")
print("Saved model to disk")
```

```
Saved model to disk
```

Load HDF5 Load Weights

```
loaded_model.load_weights("/content/model.weights.h5")
print("Loaded model from disk")
```

```
Loaded model from disk
```

Evaluate the loaded model

```
loaded_model.compile(loss='categorical_crossentropy', optimizer='SGD', metrics=['accuracy'])
score = loaded_model.evaluate(X_train, NY, verbose=0)
print("%s: %.2f%%" % (loaded_model.metrics_names[1], Results[1]*100))
```

```
accuracy: 90.87%
```

Save a model and load the model in a YAML format

```
from tensorflow.keras.models import model_from_yaml

Model_yaml = Model.to_json()
with open("Model.yaml", "w") as yaml_file:
  yaml_file.write(Model_yaml)

#Save yaml as HDF5 format for yaml
Model.save_weights("/content/model_yaml.weights.h5")
print("Saved model to disk as yaml format")
```

```
Saved model to disk as yaml format
```

Load the Yaml file

```
yaml_file = open("Model.yaml", "r")
LoadedYamlFile = yaml_file.read()
yaml_file.close()

LoadedModel = model_from_json(LoadedYamlFile)

LoadedModel.load_weights("/content/model_yaml.weights.h5")
print("Loaded model from yaml disk")
```

```
Loaded model from yaml disk
```

Evaluate the Loaded model from yaml format

```
LoadedModel.compile(loss='categorical_crossentropy', optimizer='SGD', metrics=['accuracy'])
YResults = LoadedModel.evaluate(X_train, NY, verbose = 0)
print("%s: %.2f%%" % (LoadedModel.metrics_names[1], YResults[1]*100))
```

```
accuracy: 90.87%
```

Checkpoint Neural Network Model Improvements

Create Model of the improvement

```
from keras.callbacks import ModelCheckpoint

tf.random.set_seed(42)
Model = Sequential()
Model.add(Dense(18, input_shape = (18, ), activation = 'relu'))
Model.add(Dense(12, activation = 'relu'))

Model.add(Dense(3, activation = 'softmax'))
Model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
```

Checkpoint

```
filepath="weights-improvement-{epoch:02d}-{val_accuracy:.2f}.hdf5"
Checkpoint = ModelCheckpoint(filepath, monitor = 'val_accuracy', verbose = 1, save_best_only = True, mode = 'max')
CallbackList = [Checkpoint]
```

Fit the new improve model

```
Model.fit(X_train, NY, validation_split = 0.33, epochs = 300, batch_size = 5000, callbacks = CallbackList, verbose =
```

```
Epoch 1: val_accuracy improved from -inf to 0.53994, saving model to weights-improvement-01-0.54.hdf5

Epoch 2: val_accuracy improved from 0.53994 to 0.54633, saving model to weights-improvement-02-0.55.hdf5

Epoch 3: val_accuracy improved from 0.54633 to 0.55783, saving model to weights-improvement-03-0.56.hdf5

Epoch 4: val_accuracy improved from 0.55783 to 0.56486, saving model to weights-improvement-04-0.56.hdf5

Epoch 5: val_accuracy improved from 0.56486 to 0.57636, saving model to weights-improvement-05-0.58.hdf5

Epoch 6: val_accuracy improved from 0.57636 to 0.58594, saving model to weights-improvement-06-0.59.hdf5

Epoch 7: val_accuracy improved from 0.58594 to 0.59361, saving model to weights-improvement-07-0.59.hdf5

Epoch 8: val_accuracy improved from 0.59361 to 0.60383, saving model to weights-improvement-08-0.60.hdf5

Epoch 9: val_accuracy improved from 0.60383 to 0.61725, saving model to weights-improvement-09-0.62.hdf5
Epoch 10: val_accuracy improved from 0.61725 to 0.62748, saving model to weights-improvement-10-0.63.hdf5

Epoch 11: val_accuracy improved from 0.62748 to 0.63834, saving model to weights-improvement-11-0.64.hdf5

Epoch 12: val_accuracy improved from 0.63834 to 0.64984, saving model to weights-improvement-12-0.65.hdf5

Epoch 13: val_accuracy improved from 0.64984 to 0.65942, saving model to weights-improvement-13-0.66.hdf5

Epoch 14: val_accuracy improved from 0.65942 to 0.67093, saving model to weights-improvement-14-0.67.hdf5

Epoch 15: val_accuracy improved from 0.67093 to 0.67987, saving model to weights-improvement-15-0.68.hdf5

Epoch 16: val_accuracy improved from 0.67987 to 0.68946, saving model to weights-improvement-16-0.69.hdf5

Epoch 17: val_accuracy improved from 0.68946 to 0.69265, saving model to weights-improvement-17-0.69.hdf5

Epoch 18: val_accuracy improved from 0.69265 to 0.70096, saving model to weights-improvement-18-0.70.hdf5

Epoch 19: val_accuracy improved from 0.70096 to 0.71182, saving model to weights-improvement-19-0.71.hdf5

Epoch 20: val_accuracy improved from 0.71182 to 0.72332, saving model to weights-improvement-20-0.72.hdf5

Epoch 21: val_accuracy improved from 0.72332 to 0.73610, saving model to weights-improvement-21-0.74.hdf5

Epoch 22: val_accuracy improved from 0.73610 to 0.74888, saving model to weights-improvement-22-0.75.hdf5

Epoch 23: val_accuracy improved from 0.74888 to 0.76230, saving model to weights-improvement-23-0.76.hdf5

Epoch 24: val_accuracy improved from 0.76230 to 0.76869, saving model to weights-improvement-24-0.77.hdf5

Epoch 25: val_accuracy improved from 0.76869 to 0.77636, saving model to weights-improvement-25-0.78.hdf5

Epoch 26: val_accuracy improved from 0.77636 to 0.78147, saving model to weights-improvement-26-0.78.hdf5

Epoch 27: val_accuracy improved from 0.78147 to 0.78530, saving model to weights-improvement-27-0.79.hdf5

Epoch 28: val_accuracy improved from 0.78530 to 0.78850, saving model to weights-improvement-28-0.79.hdf5

Epoch 29: val_accuracy improved from 0.78850 to 0.79489, saving model to weights-improvement-29-0.79.hdf5
```

Load the save Neural Network for improve Model

```
IResults = Model.evaluate(X_train, NY, verbose = 0)
print("%s: %.2f%%" % (Model.metrics_names[1], IResults[1]*100))
```

```
accuracy: 97.49%
```

Checkpoint Best Neural Network Model only

Create a new model for best improvement

```
#Create new Model
tf.random.set_seed(42)
Model = Sequential()
Model.add(Dense(64, input_shape = (18, ), activation = 'relu'))
Model.add(Dense(32, activation = 'relu'))

Model.add(Dense(3, activation = 'softmax'))
Model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

#Checkpoint for the best improvement
filepath="weights.best.hdf5"
Checkpoint = ModelCheckpoint(filepath, monitor = 'val_accuracy', verbose = 1, save_best_only = True, mode = 'max')
CallbackList = [Checkpoint]

#Fit the Model
Model.fit(X_train, NY, validation_split = 0.33, epochs = 300, batch_size = 5000, callbacks = CallbackList, verbose =
```

```
Epoch 1: val_accuracy improved from -inf to 0.50990, saving model to weights.best.hdf5
Epoch 2: val_accuracy improved from 0.50990 to 0.52588, saving model to weights.best.hdf5

Epoch 3: val_accuracy improved from 0.52588 to 0.54952, saving model to weights.best.hdf5

Epoch 4: val_accuracy improved from 0.54952 to 0.56422, saving model to weights.best.hdf5

Epoch 5: val_accuracy improved from 0.56422 to 0.58530, saving model to weights.best.hdf5

Epoch 6: val_accuracy improved from 0.58530 to 0.59105, saving model to weights.best.hdf5

Epoch 7: val_accuracy improved from 0.59105 to 0.59553, saving model to weights.best.hdf5

Epoch 8: val_accuracy improved from 0.59553 to 0.60192, saving model to weights.best.hdf5

Epoch 9: val_accuracy improved from 0.60192 to 0.60831, saving model to weights.best.hdf5

Epoch 10: val_accuracy improved from 0.60831 to 0.61789, saving model to weights.best.hdf5

Epoch 11: val_accuracy improved from 0.61789 to 0.63834, saving model to weights.best.hdf5

Epoch 12: val_accuracy improved from 0.63834 to 0.65112, saving model to weights.best.hdf5

Epoch 13: val_accuracy improved from 0.65112 to 0.66326, saving model to weights.best.hdf5

Epoch 14: val_accuracy improved from 0.66326 to 0.68498, saving model to weights.best.hdf5

Epoch 15: val_accuracy improved from 0.68498 to 0.70096, saving model to weights.best.hdf5

Epoch 16: val_accuracy improved from 0.70096 to 0.72141, saving model to weights.best.hdf5

Epoch 17: val_accuracy improved from 0.72141 to 0.73930, saving model to weights.best.hdf5

Epoch 18: val_accuracy improved from 0.73930 to 0.75527, saving model to weights.best.hdf5

Epoch 19: val_accuracy improved from 0.75527 to 0.77316, saving model to weights.best.hdf5

Epoch 20: val_accuracy improved from 0.77316 to 0.79042, saving model to weights.best.hdf5

Epoch 21: val_accuracy improved from 0.79042 to 0.80831, saving model to weights.best.hdf5

Epoch 22: val_accuracy improved from 0.80831 to 0.81853, saving model to weights.best.hdf5

Epoch 23: val_accuracy improved from 0.81853 to 0.82492, saving model to weights.best.hdf5

Epoch 24: val_accuracy improved from 0.82492 to 0.83131, saving model to weights.best.hdf5

Epoch 25: val_accuracy improved from 0.83131 to 0.83834, saving model to weights.best.hdf5

Epoch 26: val_accuracy improved from 0.83834 to 0.84345, saving model to weights.best.hdf5

Epoch 27: val_accuracy improved from 0.84345 to 0.85367, saving model to weights.best.hdf5

Epoch 28: val_accuracy improved from 0.85367 to 0.86006, saving model to weights.best.hdf5

Epoch 29: val_accuracy improved from 0.86006 to 0.86518, saving model to weights.best.hdf5
```

Load the save Neural Network For best improvement

```
BIResults = Model.evaluate(X_train, NY, verbose = 0)
print("%s: %.2f%%" % (Model.metrics_names[1], BIResults[1]*100))
```

```
accuracy: 99.89%
```

Load a saved Neural Network model

Create new Model

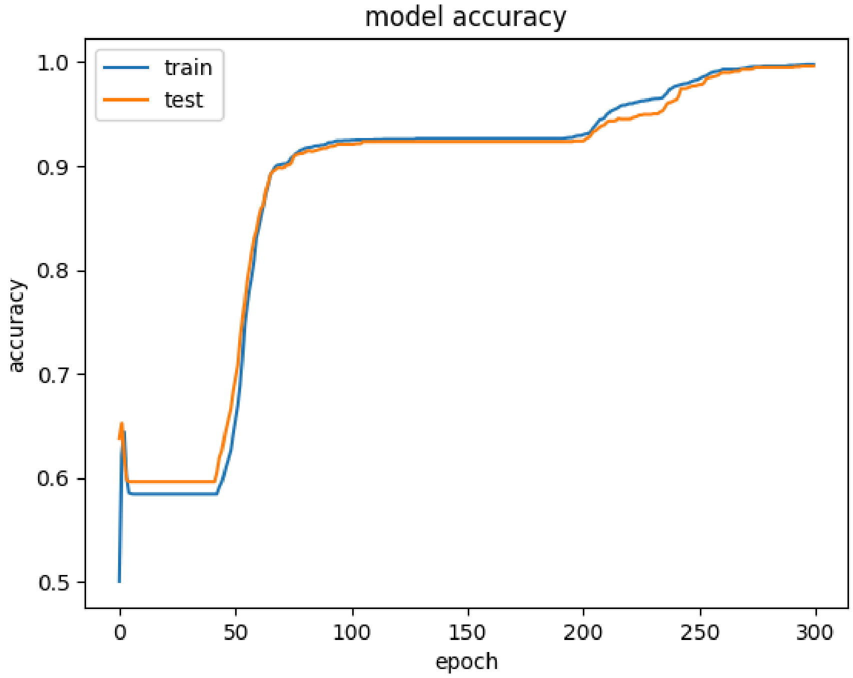
```
import matplotlib.pyplot as plt

Model = Sequential()
Model.add(Dense(64, input_shape = (18,), kernel_initializer = 'uniform', activation = 'relu'))
Model.add(Dense(32, kernel_initializer = 'uniform', activation = 'relu'))

Model.add(Dense(3, kernel_initializer = 'uniform', activation = 'softmax'))

Model.load_weights('weights.best.hdf5')

Model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
print("Created model and loaded weights from file")

LNNResults = Model.evaluate(X_train, NY, verbose = 0)
print("%s: %.2f%%" % (Model.metrics_names[1], LNNResults[1]*100))
```

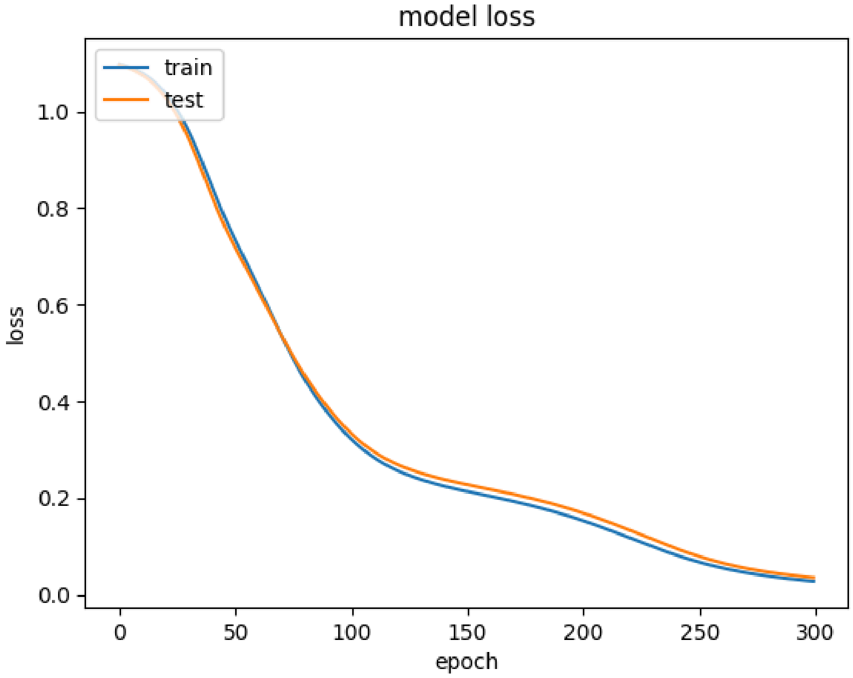
```
Created model and loaded weights from file
accuracy: 99.87%
```

Visualize Model Training History in Keras

```
Model = Sequential()
Model.add(Dense(64, input_shape = (18,), kernel_initializer = 'uniform', activation = 'relu'))
Model.add(Dense(32, kernel_initializer = 'uniform', activation = 'relu'))

Model.add(Dense(3, kernel_initializer = 'uniform', activation = 'softmax'))
Model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

history = Model.fit(X_train, NY, validation_split=0.33, epochs=300, batch_size=5000, verbose=0)

print(history.history.keys())

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

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

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





Show the application of Dropout Regularization

```
pip uninstall tensorflow
```

```
pip install tensorflow==2.1.0
```

```
!pip install scikeras
```

Load Dataset

```
import pandas as pd
import numpy as np
import tensorflow as tf
from scikeras.wrappers import KerasClassifier
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import SGD

Dataset = pd.read_csv('data.csv')
Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   gender         4746 non-null   int64  
 1   C_api          4746 non-null   object 
 2   C_man          4746 non-null   int64  
 3   E_NEds         4746 non-null   int64  
 4   E_Bpag         4746 non-null   int64  
 5   firstDay       4746 non-null   int64  
 6   lastDay        4746 non-null   int64  
 7   NEds           4746 non-null   int64  
 8   NDays          4746 non-null   int64  
 9   NActDays       4746 non-null   int64  
 10  NPages         4746 non-null   int64  
 11  NPcreated      4746 non-null   int64  
 12  pagesWomen     4746 non-null   int64  
 13  wikiprojWomen  4746 non-null   int64  
 14  ns_user        4746 non-null   int64  
 15  ns_wikipedia   4746 non-null   int64  
 16  ns_talk        4746 non-null   int64  
 17  ns_userTalk    4746 non-null   int64  
 18  ns_content     4746 non-null   int64  
 19  weightIJ       4746 non-null   float64
 20  NIJ            4746 non-null   int64  
dtypes: float64(1), int64(19), object(1)
memory usage: 778.8+ KB
```

Preprocess dataset

```
from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
for i in Dataset:
  if Dataset[i].dtypes == 'object':
    Dataset[i] = LE.fit_transform(Dataset[i])
  else:
    pass
Dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   gender         4746 non-null   int64  
 1   C_api          4746 non-null   int64  
 2   C_man          4746 non-null   int64  
 3   E_NEds         4746 non-null   int64  
 4   E_Bpag         4746 non-null   int64  
 5   firstDay       4746 non-null   int64  
 6   lastDay        4746 non-null   int64  
 7   NEds           4746 non-null   int64  
 8   NDays          4746 non-null   int64  
 9   NActDays       4746 non-null   int64  
 10  NPages         4746 non-null   int64  
 11  NPcreated      4746 non-null   int64  
 12  pagesWomen     4746 non-null   int64
 13  wikiprojWomen  4746 non-null   int64
 14  ns_user        4746 non-null   int64
 15  ns_wikipedia   4746 non-null   int64
 16  ns_talk        4746 non-null   int64
 17  ns_userTalk    4746 non-null   int64
 18  ns_content     4746 non-null   int64
 19  weightIJ       4746 non-null   float64
 20  NIJ            4746 non-null   int64
dtypes: float64(1), int64(20)
memory usage: 778.8 KB
```

```
CorrData = Dataset.corr()
TargCorr = CorrData['gender']
AbstarCor = TargCorr.abs()
LowCorrFeat = AbstarCor[AbstarCor <= 0.01].index.tolist()
print(LowCorrFeat)
```

```
['lastDay', 'NPcreated']
```

```
Dataset = Dataset.drop(columns = LowCorrFeat)
```

Splitting Dataset

```
X = Dataset.drop(columns = 'gender')
Y = Dataset['gender']
```

Encode class values

```
Encoder = LabelEncoder()
Encoder.fit(Y)
ENY = Encoder.transform(Y)
```

Create a baseline Model

```
def Base():
  Model = Sequential()
  Model.add(Dense(64, input_shape=(18,), activation='relu'))
  Model.add(Dense(32,  activation='relu'))

  Model.add(Dense(1, activation='sigmoid'))

  sgd = SGD(learning_rate=0.01, momentum=0.8)
  Model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
  return Model
```

Train the model

```
Estimators = []
Estimators.append(('Standardize', StandardScaler()))
Estimators.append(('mlp', KerasClassifier(model=Base, epochs=300, batch_size=5000, verbose=0)))
PL = Pipeline(Estimators)
FoldK = StratifiedKFold(n_splits=10, shuffle=True)
Results = cross_val_score(PL, X, ENY, cv=FoldK)
print("Baseline: %.2f%% (%.2f%%)" % (Results.mean()*100, Results.std()*100))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
```

```
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:86: UserWarning: Do not pass an `input_sh
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Baseline: 33.73% (0.06%)
```

Show the application of Dropout on the visible layer

Create a model with dropout

```
from tensorflow.keras.layers import Dropout
from tensorflow.keras.constraints import MaxNorm

def Base1():
  Model = Sequential()
  Model.add(Dropout(0.2, input_shape=(18,)))
  Model.add(Dense(64, activation='relu', kernel_constraint=MaxNorm(3)))
  Model.add(Dense(32, activation='relu', kernel_constraint=MaxNorm(3)))

  Model.add(Dense(1, activation='sigmoid'))

  sgd = SGD(learning_rate=0.01, momentum=0.8)
  Model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
  return Model
```

Train the model

```
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(model=Base1, epochs=300, batch_size=5000, verbose=0)))
PL = Pipeline(estimators)
FoldK = StratifiedKFold(n_splits=10, shuffle=True)
Results = cross_val_score(PL, X, ENY, cv=FoldK)
print("Visible: %.2f%% (%.2f%%)" % (Results.mean()*100, Results.std()*100))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
Visible: 33.73% (0.06%)
```

Show the application of Dropout on the hidden layer

Create a new Model for dropout on the hidden layer

```
def Base2():
    Model = Sequential()
    Model.add(Dropout(0.5, input_shape=(18,)))
    Model.add(Dense(60, activation='relu', kernel_constraint=MaxNorm(3)))
    Model.add(Dense(30, activation='relu', kernel_constraint=MaxNorm(3)))

    Model.add(Dense(1, activation='sigmoid'))
    # Compile model
    sgd = SGD(learning_rate=0.1, momentum=0.9)
    Model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])
    return Model
```

Train the Model

```
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasClassifier(model=Base2, epochs=300, batch_size=5000, verbose=0)))
pipeline = Pipeline(estimators)
kfold = StratifiedKFold(n_splits=10, shuffle=True)
results = cross_val_score(pipeline, X, ENY, cv=kfold)
print("Visible: %.2f%% (%.2f%%)" % (results.mean()*100, results.std()*100))
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/regularization/dropout.py:42: UserWarning: Do not pass
  super().__init__(**kwargs)
Visible: 33.73% (0.06%)
```

Remarks: As seen from visible results, from Show the application of Dropout on the visible layer, the results is always 33.73, the reason that I come is that the dataset is not overfitting therefore the result remain the same, the other reason is that my implementation is not good enough to able do work on Dropout, the last reason is that my training for the dataset is insuficient.

Show the application of a time-based learning rate schedule

```
pip install keras==2.1.0
```

```
import os
os.environ['TF_USE_LEGACY_KERAS'] = 'True'
```

```
import pandas as pd
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.optimizers.legacy import SGD

DataFrame = pd.read_csv('data.csv')


from sklearn.preprocessing import LabelEncoder
LE = LabelEncoder()
for i in DataFrame:
  if DataFrame[i].dtypes == 'object':
    DataFrame[i] = LE.fit_transform(DataFrame[i])
  else:
    pass
DataFrame.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4746 entries, 0 to 4745
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   gender         4746 non-null   int64  
 1   C_api          4746 non-null   int64  
 2   C_man          4746 non-null   int64  
 3   E_NEds         4746 non-null   int64  
 4   E_Bpag         4746 non-null   int64  
 5   firstDay       4746 non-null   int64  
 6   lastDay        4746 non-null   int64  
 7   NEds           4746 non-null   int64  
 8   NDays          4746 non-null   int64  
 9   NActDays       4746 non-null   int64  
 10  NPages         4746 non-null   int64  
 11  NPcreated      4746 non-null   int64  
 12  pagesWomen     4746 non-null   int64  
 13  wikiprojWomen  4746 non-null   int64  
 14  ns_user        4746 non-null   int64  
 15  ns_wikipedia   4746 non-null   int64  
 16  ns_talk        4746 non-null   int64  
 17  ns_userTalk    4746 non-null   int64  
 18  ns_content     4746 non-null   int64  
 19  weightIJ       4746 non-null   float64
 20  NIJ            4746 non-null   int64  
dtypes: float64(1), int64(20)
memory usage: 778.8 KB
```

```
Dataset2 = DataFrame.values
```

Splitting dataset

```
X = Dataset2[:, 0:20].astype(float)
Y = Dataset2[:, 20]
```

Encode class as integers

```
Enco = LabelEncoder()
Enco.fit(Y)
ELEY = Enco.transform(Y)
```

Create Model

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

Model = Sequential()
Model.add(Dense(20, input_shape = (20, ), activation = 'relu'))

Model.add(Dense(1, activation = 'sigmoid'))

epochs = 300
learning_rate = 0.1
decay_rate = learning_rate / epochs
momentum = 0.8
sgd = SGD(learning_rate = learning_rate, momentum = momentum, decay = decay_rate, nesterov = False)
Model.compile(loss = 'binary_crossentropy', optimizer = sgd, metrics = ['accuracy'])
```

Train the Model

```
Model.fit(X, ELEY, validation_split=0.33, epochs=epochs, batch_size=5000, verbose=2)
```

```
    1/1 - 0s - loss: nan - accuracy: 0.0528 - val_loss: nan - val_accuracy: 0.0581 - 43ms/epoch - 43ms/step
    <tf_keras.src.callbacks.History at 0x7b997a7a8640>
```

Show the application of a drop-based learning rate schedule

```
from tensorflow.keras.callbacks import LearningRateScheduler
import math

def step_decay(epoch):
  initial_lrate = 0.1
  drop = 0.5
  epochs_drop = 10.0
  lrate = initial_lrate * math.pow(drop, math.floor((1+epoch)/epochs_drop))
  return lrate
```

Create Model

```
X = Dataset2[:, 0:20].astype(float)
Y = Dataset2[:, 20]

Enco = LabelEncoder()
Enco.fit(Y)
ELEY = Enco.transform(Y)

Model = Sequential()
Model.add(Dense(20, input_shape=(20,), activation='relu'))
Model.add(Dense(1, activation='sigmoid'))

sgd = SGD(learning_rate=0.0, momentum=0.9)
Model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])

lrate = LearningRateScheduler(step_decay)
callbacks_list = [lrate]
```

Train the Model

```
Model.fit(X, ELEY, validation_split=0.33, epochs=50, batch_size=5000, callbacks=callbacks_list, verbose=2)
```

```
Epoch 38/50
1/1 - 0s - loss: nan - accuracy: 0.0528 - val_loss: nan - val_accuracy: 0.0581 - lr: 0.0125 - 126ms/epoch - 12
Epoch 39/50
1/1 - 0s - loss: nan - accuracy: 0.0528 - val_loss: nan - val_accuracy: 0.0581 - lr: 0.0125 - 201ms/epoch - 20
Epoch 40/50
1/1 - 0s - loss: nan - accuracy: 0.0528 - val_loss: nan - val_accuracy: 0.0581 - lr: 0.0063 - 77ms/epoch - 77m
Epoch 41/50
1/1 - 0s - loss: nan - accuracy: 0.0528 - val_loss: nan - val_accuracy: 0.0581 - lr: 0.0063 - 70ms/epoch - 70m
```