```
# This is formatted as code
```

DMA Fall 22

Note : This entire lab will be manually evaluated.

Name : 'Lilly Liu' Collaborator : ''

Lab 4: Neural Networks

```
import pandas as pd
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC

from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.feature_extraction import DictVectorizer

from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV, ParameterGrid

import numpy as np

import warnings
warnings.filterwarnings("ignore")


!wget http://askoski.berkeley.edu/~zp/lab_4_training.csv
!wget http://askoski.berkeley.edu/~zp/lab_4_test.csv
```

```
df_train = pd.read_csv('./lab_4_training.csv')
df_test = pd.read_csv('./lab_4_test.csv')
df_train.head()
```

```
--2022-09-28 00:31:58--  http://askoski.berkeley.edu/~zp/lab_4_training.csv
```

```
df_test.head()
```

	Unnamed: 0	gender	age	year	eyecolor	height	miles	brothers	sisters	computertime	exercise	exercisehc
0	1303	male	20	second	green	73.0	210.0	0	1	10.0	Yes	
1	36	male	20	third	other	71.0	90.0	1	0	15.0	Yes	
2	489	male	22	fourth	hazel	75.0	200.0	0	1	1.0	Yes	
3	1415	male	19	second	brown	72.0	35.0	2	2	20.0	Yes	
4	616	male	22	fourth	hazel	71.0	15.0	2	1	10.0	Yes	

```
2022-09-28 00:31:58 (1.55 MB/s) - 'lab_4_test.csv.1' saved [26519/26519]
```

Question 1

Calculate a baseline accuracy measure using the majority class, assuming a target variable of `gender`. The majority class is the most common value of the target variable in a particular dataset. Accuracy is calculated as (true positives + true negatives) / (all negatives and positives).

2	1738	male	20	second	brown	63.0	55.0	1	2	15.0	Yes	

Question 1.a

Find the majority class in the training set. If you always predicted this class in the training set, what would your accuracy be?

```
# YOUR CODE HERE
majority = "female"
tp = (df_train['gender'] == 'female').sum()
tn = 0
total = df_train['gender'].count()
```

```
train_acc = (tp + tn)/total
train_acc
```

```
0.5427852348993288
```

ANSWER: 0.5427852348993288

Question 1.b

If you always predicted this same class (majority from the training set) in the test set, what would your accuracy be?

```
majority = "female"
tp = (df_test['gender'] == 'female').sum()
tn = 0
total = df_test['gender'].count()
test_acc = (tp + tn)/total
test_acc
```

```
0.5226130653266332
```

ANSWER: 0.5226130653266332

Question 2

Get started with Neural Networks.

Choose a NN implementation (eg: scikit-learn) and specify which you choose. Be sure the implementation allows you to modify the number of hidden layers and hidden nodes per layer.

NOTE: When possible, specify the logsig (`sigmoid` / `logistic`) function as the transfer function (another word for activation function) and use Levenberg-Marquardt backpropagation (`lbfgs`). It is possible to specify logistic in Sklearn MLPclassifier (Neural net).

Question 2.a

Train a neural network with a single 10 node hidden layer. Only use the `height` feature of the dataset to predict the `gender`. You will have to change `gender` to a 0 and 1 class. After training, use your trained model to predict the class (`gender`) using the `height` feature from the training set. What is the accuracy of this prediction?

```
print(df_train.shape, df_test.shape)
```

```
    (1192, 15) (398, 15)
```

```
bin_dftrain = df_train.replace('male', 0).replace('female', 1)
bin_dftest = df_test.replace('male', 0).replace('female', 1)
X_train = bin_dftrain[["height"]]
Y_train = bin_dftrain["gender"]
print(X_train.shape, Y_train.shape)
```

```
    (1192, 1) (1192,)
```

```
classify = MLPClassifier(hidden_layer_sizes=(10),
                         activation = 'logistic', solver='lbfgs', random_state=42)
classify.fit(X_train,Y_train)
y_pred_train=classify.predict(X_train)
print(accuracy_score(Y_train,y_pred_train))
```

```
    0.8439597315436241
```

▾ ANSWER: 0.8439597315436241

Question 2.b

Take the trained model from question 2.a and use it to predict the test set. This can be accomplished by taking the trained model and giving it the `height` feature values from the test set. What is the accuracy of this model on the test set?

```
X_test = bin_dftest[["height"]]
Y_test = bin_dftest["gender"]
y_pred_test=classify.predict(X_test)
print(accuracy_score(Y_test,y_pred_test))
```

```
0.8542713567839196
```

ANSWER: 0.8542713567839196

Question 2.c

Neural Networks tend to prefer smaller, normalized feature values. Try taking the log of the `height` feature in both training and testing sets or use a Standard Scalar operation in SKlearn to centre and normalize the data between 0-1 for continuous values. Repeat question 2.a and 2.b with the log version or the normalized and centered version of this feature.

```
# YOUR CODE HERE
log_Xtrain = np.log(bin_dftrain[['height']])
log_Ytrain = bin_dftrain['gender']
log_Xtest = np.log(bin_dftest[['height']])
log_Ytest = bin_dftest['gender']
classify.fit(log_Xtrain, log_Ytrain)
log_ypredtrain=classify.predict(log_Xtrain)
print("training accuracy", accuracy_score(log_Ytrain,log_ypredtrain))
log_ypredtest=classify.predict(log_Xtest)
print("test accuracy",accuracy_score(log_Ytest,log_ypredtest))
```

```
training accuracy 0.8338926174496645
test accuracy 0.8291457286432161
```

ANSWER:

Question 3

The rest of features in this dataset except a few are categorical. No ML method accepts categorical features, so transform `year`, `eyecolor`, `exercise` into a set of binary features, one feature per unique original feature value, and mark the binary feature as '1' if the feature value matches the original value and '0' otherwise. Using only these binary variable transformed features, train and predict the class of the test set. What was your accuracy using Neural Network with a single 10 node hidden layer? During training, use a maximum number of iterations of 50.

```
df3trainhot= pd.get_dummies(df_train[['year', 'eyecolor', 'exercise']])
df3testhot = pd.get_dummies(df_test[['year', 'eyecolor', 'exercise']])
classify3 = MLPClassifier(hidden_layer_sizes = (10, ),
                          activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify3.fit(df3trainhot, bin_dftrain["gender"])
y_trainpred = classify3.predict(df3trainhot)
print("training:", accuracy_score(bin_dftrain["gender"], y_trainpred),
      " testing:", classify3.score(df3testhot, bin_dftest["gender"]))
```

```
    training: 0.5679530201342282  testing: 0.5527638190954773
```

ANSWER: The training accuracy using Neural Network with the one hot coded features is 0.5679530201342282, and the testing accuracy is 0.5527638190954773.

Question 4

Using a NN, report the accuracy on the test set of a model that trained only on `height` and the `eyecolor` features of instances in the training set.

Question 4.a

What is the accuracy on the test set using the original `height` values (no pre-processing) and `eyecolor` as a one-hot?

```
cols= ["eyecolor_blue", "eyecolor_brown", "eyecolor_green", "eyecolor_hazel", "eyecolor_other"]
```

```
train_he = pd.concat([df3trainhot[cols], bin_dftrain["height"]], axis = 1)
test_he = pd.concat([df3testhot[cols], bin_dftest["height"]], axis = 1)
classify4 = MLPClassifier(hidden_layer_sizes = (10,),
                          activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify4.fit(train_he, bin_dftrain["gender"])
print("training:", classify4.score(train_he, bin_dftrain["gender"]),
      " testing:", classify4.score(test_he, bin_dftest["gender"]))
```

```
training: 0.5427852348993288  testing: 0.5226130653266332
```

ANSWER: training: 0.5427852348993288 ; testing: 0.5226130653266332

Question 4.b

What is the accuracy on the test set using the log of `height` values (applied to both training and testing sets) and `eyecolor` as a one-hot?

```
log_dftrain = df_train.copy()
log_dftrain["height"] = np.log(log_dftrain["height"])
log_dftest = df_test.copy()
log_dftest["height"] = np.log(log_dftest["height"])
log_dftrain["gender"] = log_dftrain["gender"].replace("female", 1).replace("male", 0)
log_dftest["gender"] = log_dftest["gender"].replace("female", 1).replace("male", 0)
```

```
train_loghe = pd.concat([df3trainhot[cols], log_dftrain["height"]], axis = 1)
test_loghe = pd.concat([df3testhot[cols], log_dftest["height"]], axis = 1)
classify4b = MLPClassifier(hidden_layer_sizes = (10,),
                           activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify4b.fit(train_loghe, bin_dftrain["gender"])
print("training:", classify4b.score(train_he, log_dftrain["gender"]),
      " testing:", classify4b.score(test_he, log_dftest["gender"]))
```

```
training: 0.45721476510067116  testing: 0.47738693467336685
```

ANSWER: training: 0.457214765100671116 testing: 0.47738693467336685

Question 4.c

What is the accuracy on the test set using the Z-score of `height` values and `eyecolor` as a one-hot?

Z-score is a normalization function. It is the value of a feature minus the average value for that feature (in the training set), divided by the standard deviation of that feature (in the training set). Remember that, whenever applying a function to a feature in the training set, it also has to be applied to that same feature in the test set.

```
from scipy.stats import zscore
train_z_he = pd.concat([df3trainhot[cols], pd.Series(zscore(bin_dftrain["height"]))], axis = 1)
test_z_he = pd.concat([df3testhot[cols], pd.Series(zscore(bin_dftest["height"]))], axis = 1)
classify4c = MLPClassifier(hidden_layer_sizes = (10,),
                           activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify4c.fit(train_z_he, bin_dftrain["gender"])
print("training accuracy:", classify4c.score(train_z_he, bin_dftrain["gender"]),
      " testing accuracy:", classify4c.score(test_z_he, bin_dftest["gender"]))
```

```
    training accuracy: 0.8456375838926175  testing accuracy: 0.8618090452261307
```

ANSWER: training accuracy: 0.8456375838926175 testing accuracy: 0.8618090452261307

Question 5

Repeat question 4 for `exercisehours` & `eyecolor`.

```
#5a
cols= ["eyecolor_blue", "eyecolor_brown", "eyecolor_green", "eyecolor_hazel", "eyecolor_other"]
train_ee = pd.concat([df3trainhot[cols], bin_dftrain["exercisehours"]], axis = 1)
test_ee = pd.concat([df3testhot[cols], bin_dftest["exercisehours"]], axis = 1)
classify5 = MLPClassifier(hidden_layer_sizes = (10,),
                          activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify5.fit(train_ee, bin_dftrain["gender"])
```

```
print("training:", classify5.score(train_ee, bin_dftrain["gender"]),
      " testing:", classify5.score(test_ee, bin_dftest["gender"]))
#5b
train_logee = pd.concat([df3trainhot[cols], np.log(df_train["exercisehours"].replace({0: 0.000000001}))], axis = 1
test_logee = pd.concat([df3testhot[cols], np.log(df_test["exercisehours"].replace({0: 0.000000001}))], axis = 1)
classify5b = MLPClassifier(hidden_layer_sizes = (10,),
                           activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify5b.fit(train_logee, bin_dftrain["gender"])
print("training:", classify5b.score(train_logee, log_dftrain["gender"]),
      " testing:", classify5b.score(test_logee, log_dftest["gender"]))
#5c

train_z_ee = pd.concat([df3trainhot[cols], pd.Series(zscore(bin_dftrain["exercisehours"]))], axis = 1)
test_z_ee = pd.concat([df3testhot[cols], pd.Series(zscore(bin_dftest["exercisehours"]))], axis = 1)
classify5c = MLPClassifier(hidden_layer_sizes = (10,),
                           activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify5c.fit(train_z_ee, bin_dftrain["gender"])
print("training:", classify5c.score(train_z_ee, bin_dftrain["gender"]),
      " testing:", classify5c.score(test_z_ee, bin_dftest["gender"]))
```

```
training: 0.5796979865771812  testing: 0.5728643216080402
training: 0.5830536912751678  testing: 0.5703517587939698
training: 0.5922818791946308  testing: 0.5603015075376885
```

ANSWER:

(no preprocessing exercisehours and encoding eyecolor) training: 0.5796979865771812 testing: 0.5728643216080402 (log exercisehours and encoding eyecolor) training: 0.5830536912751678 testing: 0.5703517587939698 (zscore exercisehours and encoding eyecolor)training: 0.5922818791946308 testing: 0.5603015075376885

Question 6

Combine the features from question 3, 4, and 5 (`year`, `eyecolor`, `exercise`, `height`, `exercisehours`). For numeric features use

Question 6.a

What was the NN accuracy on the test set using the single 10 node hidden layer?

```
trainNN = pd.concat([df3trainhot, pd.Series(zscore(bin_dftrain["height"])),
                     pd.Series(zscore(bin_dftrain["exercisehours"]))], axis = 1)
testNN = pd.concat([df3testhot, pd.Series(zscore(bin_dftest["height"])),
                    pd.Series(zscore(bin_dftest["exercisehours"]))], axis = 1)
classify6 = MLPClassifier(hidden_layer_sizes = (10, ),
                          activation = "logistic", solver = 'lbfgs', max_iter = 50, random_state = 42)
classify6.fit(trainNN, bin_dftrain["gender"])
print("testing accuracy:", classify6.score(testNN, bin_dftest["gender"]))
```

```
testing accuracy: 0.8391959798994975
```

ANSWER: NN testing accuracy: 0.8391959798994975

Question 7- Bonus (10%)

Can you improve your test set prediction accuracy by 5% or more?

See how close to that milestone of improvement you can get by modifying the tuning parameters of Neural Networks (the number of hidden layers, number of hidden nodes in each layer, the learning rate aka mu). A great guide to tuning parameters is explained in this guide: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

While the guide is specific to SVM and in particular the C and gamma parameters of the RBF kernel, the method applies to generally to any ML technique with tuning parameters.

Please also write a paragraph in a markdown cell below with an explanation of your approach and evaluation metrics.

```
# YOUR CODE HERE
```

ANSWER: