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

DMA Fall 22

Note : This entire lab will be manually evaluated.

Name : 'Mary Guo' 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 15:56:35--  http://askoski.berkeley.edu/~zp/lab_4_training.csv
Resolving askoski.berkeley.edu (askoski.berkeley.edu)... 169.229.192.179
Connecting to askoski.berkeley.edu (askoski.berkeley.edu)|169.229.192.179|:80...
HTTP request sent, awaiting response... 200 OK
Length: 79177 (77K) [text/csv]
Saving to: ‘lab_4_training.csv’

lab_4_training.csv  100%[===================>]  77.32K   281KB/s    in 0.3s

2022-09-28 15:56:36 (281 KB/s) - ‘lab_4_training.csv’ saved [79177/79177]

--2022-09-28 15:56:36--  http://askoski.berkeley.edu/~zp/lab_4_test.csv
Resolving askoski.berkeley.edu (askoski.berkeley.edu)... 169.229.192.179
Connecting to askoski.berkeley.edu (askoski.berkeley.edu)|169.229.192.179|:80...
HTTP request sent, awaiting response... 200 OK
Length: 26519 (26K) [text/csv]
Saving to: ‘lab_4_test.csv’

lab_4_test.csv      100%[===================>]  25.90K  --.-KB/s    in 0.1s

2022-09-28 15:56:37 (190 KB/s) - ‘lab_4_test.csv’ saved [26519/26519]
```

	Unnamed: 0	gender	age	year	eyecolor	height	miles	brothers	sisters	com

```
df_test.head()
```

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

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).

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
df_train.groupby('gender').agg({'Unnamed: 0':'count'}).sort_values(by = 'Unnamed: 0',
```

```
0.5427852348993288
```

ANSWER: the majority class in the training set is female. The accuracy will be 0.5428

Question 1.b

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

```
# YOUR CODE HERE
df_new = df_test.groupby('gender').agg({'Unnamed: 0':'count'}).reset_index()
df_new[df_new['gender'] == 'female'].iloc[0,1]/len(df_test)
```

```
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?

```
# YOUR CODE HERE
```

```
X_train = df_train[['height']]
Y_train = df_train.replace({'male': 0, 'female': 1})[['gender']]
clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=100, activation='logistic',
                    solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train,Y_train)
y_pred_train=clf.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?

```
# YOUR CODE HERE
X_test = df_test[['height']]
Y_test = df_test.replace({'male': 0, 'female': 1})[['gender']]
y_pred_test=clf.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
X_train_log = np.log(df_train[['height']])
Y_train = df_train.replace({'male': 0, 'female': 1})[['gender']]
clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=100, activation='logistic',
                    solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_log,Y_train)
y_pred_train_log=clf.predict(X_train_log)
print(accuracy_score(Y_train,y_pred_train_log))

X_test_log = np.log(df_test[['height']])
```

```
Y_test = df_test.replace({'male': 0, 'female': 1})[['gender']]
y_pred_test_log =clf.predict(X_test_log)
print(accuracy_score(Y_test,y_pred_test_log))
```

```
0.8439597315436241
0.8542713567839196
```

ANSWER: The training set accruacy is 0.8439597315436241, the testing set accuracy is 0.8542713567839196

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.

```
# YOUR CODE HERE
df_train_binary = pd.get_dummies(df_train.replace({'male': 0, 'female': 1})[['gender',
df_test_binary = pd.get_dummies(df_test.replace({'male': 0, 'female': 1})[['gender', '
X_train_binary = df_train_binary.drop(columns= ['gender'])
Y_train_binary = df_train_binary[['gender']]

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                    solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_binary,Y_train_binary)

X_test_binary = df_test_binary.drop(columns= ['gender'])
Y_test_binary = df_test_binary[['gender']]

y_pred_test_binary =clf.predict(X_test_binary)
print(accuracy_score(Y_test_binary,y_pred_test_binary))
```

```
0.535175879396985
```

ANSWER: 0.535175879396985

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?

```
# YOUR CODE HERE
df_train_q4 = pd.get_dummies(df_train.replace({'male': 0, 'female': 1})[['height', 'ge
df_test_q4 = pd.get_dummies(df_test.replace({'male': 0, 'female': 1})[['height', 'genc
X_train_q4a = df_train_q4.drop(columns= ['gender'])
Y_train_q4a = df_train_q4[['gender']]

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                     solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_q4a,Y_train_q4a)

X_test_q4a = df_test_q4.drop(columns= ['gender'])
Y_test_q4a = df_test_q4[['gender']]

y_pred_test_q4a =clf.predict(X_test_q4a)
print(accuracy_score(Y_test_q4a,y_pred_test_q4a))
```

```
0.7839195979899497
```

ANSWER: 0.7839195979899497

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?

```
# YOUR CODE HERE

X_train_q4b = df_train_q4.drop(columns= ['gender'])
Y_train_q4b = df_train_q4[['gender']]

X_train_q4b['height_log'] = np.log(X_train_q4b['height'])
X_train_q4b = X_train_q4b.drop(columns = ['height'])

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                     solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_q4b,Y_train_q4b)
```

```
X_test_q4b = df_test_q4.drop(columns= ['gender'])
Y_test_q4b = df_test_q4[['gender']]

X_test_q4b['height_log'] = np.log(X_test_q4b['height'])
X_test_q4b = X_test_q4b.drop(columns = ['height'])

y_pred_test_q4b =clf.predict(X_test_q4b)
print(accuracy_score(Y_test_q4b,y_pred_test_q4b))
```

```
0.7361809045226131
```

ANSWER: 0.7361809045226131

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.

```
# YOUR CODE HERE

X_train_q4c = df_train_q4.drop(columns= ['gender'])
Y_train_q4c = df_train_q4[['gender']]

X_train_q4c['height_z'] = (X_train_q4c['height'] - np.mean(X_train_q4c['height']))/np.

X_train_q4c = X_train_q4c.drop(columns = ['height'])

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                     solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_q4c,Y_train_q4c)

X_test_q4c = df_test_q4.drop(columns= ['gender'])
Y_test_q4c = df_test_q4[['gender']]

X_test_q4c['height_z'] = (X_test_q4c['height'] - np.mean(X_test_q4c['height']))/np.st

X_test_q4c = X_test_q4c.drop(columns = ['height'])

y_pred_test_q4c =clf.predict(X_test_q4c)
print(accuracy_score(Y_test_q4c,y_pred_test_q4c))
```

```
0.8693467336683417
```

ANSWER: 0.8693467336683417

Question 5

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

```
# YOUR CODE HERE
# q5a
df_train_q5 = pd.get_dummies(df_train.replace({'male': 0, 'female': 1})[['exercisehour
df_test_q5 = pd.get_dummies(df_test.replace({'male': 0, 'female': 1})[['exercisehours'
X_train_q5a = df_train_q5.drop(columns= ['gender'])
Y_train_q5a = df_train_q5[['gender']]

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                     solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_q5a,Y_train_q5a)

X_test_q5a = df_test_q5.drop(columns= ['gender'])
Y_test_q5a = df_test_q5[['gender']]

y_pred_test_q5a =clf.predict(X_test_q5a)
print(accuracy_score(Y_test_q5a,y_pred_test_q5a))

#q5b
df_train_q5b = df_train_q5[~(df_train_q5['exercisehours'] == 0)]
df_test_q5b = df_test_q5[~(df_test_q5['exercisehours'] == 0)]

df_train_q5b['exercisehours_log'] = np.log(df_train_q5b['exercisehours'])
df_test_q5b['exercisehours_log'] = np.log(df_test_q5b['exercisehours'])



X_train_q5b = df_train_q5b.drop(columns= ['gender', 'exercisehours'])
Y_train_q5b = df_train_q5b[['gender']]


clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                     solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_q5b,Y_train_q5b)

X_test_q5b = df_test_q5b.drop(columns= ['gender', 'exercisehours'])
Y_test_q5b = df_test_q5b[['gender']]

y_pred_test_q5b =clf.predict(X_test_q5b)
print(accuracy_score(Y_test_q5b,y_pred_test_q5b))

#q5c
X_train_q5c = df_train_q5.drop(columns= ['gender'])
Y_train_q5c = df_train_q5[['gender']]
```

```
X_train_q5c['exercisehours_z'] = (X_train_q5c['exercisehours'] - np.mean(X_train_q5c['
X_train_q5c = X_train_q5c.drop(columns =['exercisehours'])

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                     solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_q5c,Y_train_q5c)

X_test_q5c = df_test_q5.drop(columns= ['gender'])
Y_test_q5c = df_test_q5[['gender']]

X_test_q5c['exercisehours_z'] = (X_test_q5c['exercisehours'] - np.mean(X_test_q5c['exe
X_test_q5c = X_test_q5c.drop(columns =['exercisehours'])

y_pred_test_q5c =clf.predict(X_test_q5c)
print(accuracy_score(Y_test_q5c,y_pred_test_q5c))
```

```
0.5653266331658291
0.5941422594142259
0.5603015075376885
```

ANSWER: 5a: 0.5653266331658291, 5b: 0.5941422594142259, 5c: 0.5603015075376885

Question 6

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

Question 6.a

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

```
# YOUR CODE HERE
df_train_q6 = pd.get_dummies(df_train.replace({'male': 0, 'female': 1})[['exercisehou
df_test_q6 = pd.get_dummies(df_test.replace({'male': 0, 'female': 1})[['exercisehours'

df_train_q6 = df_train_q6[~(df_train_q6['exercisehours'] == 0)]
df_test_q6 = df_test_q6[~(df_test_q6['exercisehours'] == 0)]

df_train_q6['exercisehours_log'] = np.log(df_train_q6['exercisehours'])
df_test_q6['exercisehours_log'] = np.log(df_test_q6['exercisehours'])

df_train_q6['height_z'] = (df_train_q6['height'] - np.mean(df_train_q6['height']))/np.
df_test_q6['height_z'] = (df_test_q6['height'] - np.mean(df_test_q6['height']))/np.st
```

```
X_train_q6 = df_train_q6.drop(columns= ['gender', 'exercisehours', 'height'])
Y_train_q6 = df_train_q6[['gender']]

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=50, activation='logistic',
                     solver='lbfgs', verbose=1,  random_state=43)
clf.fit(X_train_q6,Y_train_q6)

X_test_q6 = df_test_q6.drop(columns= ['gender', 'exercisehours', 'height'])
Y_test_q6 = df_test_q6[['gender']]

y_pred_test_q6 =clf.predict(X_test_q6)
print(accuracy_score(Y_test_q6,y_pred_test_q6))
```

```
0.8284518828451883
```

ANSWER: 0.8284518828451883

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

df_train_q6 = pd.get_dummies(df_train.replace({'male': 0, 'female': 1})[['exercisehour
df_test_q6 = pd.get_dummies(df_test.replace({'male': 0, 'female': 1})[['exercisehours'

df_train_q6 = df_train_q6[~(df_train_q6['exercisehours'] == 0)]
df_test_q6 = df_test_q6[~(df_test_q6['exercisehours'] == 0)]

df_train_q6['exercisehours_log'] = np.log(df_train_q6['exercisehours'])
df_test_q6['exercisehours_log'] = np.log(df_test_q6['exercisehours'])
```

```
df_train_q6['height_z'] = (df_train_q6['height'] - np.mean(df_train_q6['height']))/np.
df_test_q6['height_z'] = (df_test_q6['height'] - np.mean(df_test_q6['height']))/np.std

X_train_q6 = df_train_q6.drop(columns= ['gender', 'exercisehours', 'height'])
Y_train_q6 = df_train_q6[['gender']]

clf = MLPClassifier(hidden_layer_sizes=(10), max_iter=22, activation='logistic',
                    solver='lbfgs', verbose=1,  random_state=43) #10, 22 #15, 34
clf.fit(X_train_q6,Y_train_q6)

X_test_q6 = df_test_q6.drop(columns= ['gender', 'exercisehours', 'height'])
Y_test_q6 = df_test_q6[['gender']]

y_pred_test_q6 =clf.predict(X_test_q6)
print(accuracy_score(Y_test_q6,y_pred_test_q6))
```

```
0.8619246861924686
```

ANSWER: The final accuracy is 0.8619246861924686. I found out that when the hidden layer sizes and max_iter become super big like above 100, the accuracy actually decreases from question 6. Also I observed the parameters they choose in the paper is also small like 8. So I started with 8 hidden layer sizes and 16 max iter (double amount of the hidden layer sizes), and played around it. The final hidden_layer_sizes I chose is 10, and max_iter is 22. Also, I found out if I add two more hidden layers(second layer size is 6, the third layer size is 2), it will gives me the same accuracy 0.8619246861924686

✓ 0s completed at 9:39 AM