DMA 2023

Make sure you fill in any place that says `YOUR CODE HERE` or `YOUR ANSWER HERE`, as well as your name below:

```
NAME = "Jeonghyun Lee"
```

Lab 4: Neural Networks

Please read the following instructions very carefully

Working on the assignment / FAQs

- **Always use the seed/random_state as *42* wherever applicable** (This is to ensure repeatability in answers, across questions, students and coding environments).
- All questions will be graded manually.
- Most questions have two cells:
  - A code cell for your work/code
  - A text cell for giving your final answer
- The points each question carries are indicated.
- Most assignments have bonus questions for extra credit, do try them out!
- **Submitting the assignment** : Download the '.ipynb' and '.pdf' files from Colab and upload them to Gradescope. Do not delete any outputs from cells before submitting. Make sure to assign pages to questions when uploading your PDF to Gradescope.
- That's about it. Happy coding!

About the dataset

This assignment uses a dataset obtained from the JSE Data Archive that contains biological and self-reported activity traits of a sample of college students at a single university uploaded in 2013. Background Information on the dataset:
http://jse.amstat.org/v21n2/froelich/eyecolorgender.txt

For this lab, the dataset has already been split into a training set `df_train` and a test set `df_test`.

```
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()
```

```
--2023-09-27 11:20:14--  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|:
HTTP request sent, awaiting response... 200 OK
Length: 79177 (77K) [text/csv]
Saving to: 'lab_4_training.csv.1'

lab_4_training.csv. 100%[===================>]  77.32K  --.-KB/s    in 0.06

2023-09-27 11:20:15 (1.34 MB/s) - 'lab_4_training.csv.1' saved [79177/79177

--2023-09-27 11:20:15--  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|:
HTTP request sent, awaiting response... 200 OK
Length: 26519 (26K) [text/csv]
Saving to: 'lab_4_test.csv.1'

lab_4_test.csv.1    100%[===================>]  25.90K  --.-KB/s    in 0.02

2023-09-27 11:20:15 (1.45 MB/s) - 'lab_4_test.csv.1' saved [26519/26519]
```

	Unnamed: 0	gender	age	year	eyecolor	height	miles	brothers	sisters	c
0	577	male	20	third	hazel	72.0	180.0	0	0	
1	677	male	19	second	hazel	72.0	120.0	1	1	
2	1738	male	20	second	brown	63.0	55.0	1	2	
3	1355	male	20	third	green	78.0	200.0	0	0	
4	891	female	19	second	green	67.0	280.0	2	0	

```
df_test.head()
```

	Unnamed: 0	gender	age	year	eyecolor	height	miles	brothers	sisters	c
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 (1 point)

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
majority_class = df_train['gender'].value_counts().idxmax()


true_positives = (df_train['gender'] == majority_class).sum()
total_samples = df_train.shape[0]
accuracy = true_positives / total_samples

print(accuracy)
```

```
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?

```
# YOUR CODE HERE
true_positives_test = (df_test['gender'] == majority_class).sum()
total_samples_test = df_test.shape[0]
accuracy_test = true_positives_test / total_samples_test

print(accuracy_test)
```

```
0.5226130653266332
```

Answer: 0.5226130653266332

Question 2 (1.5 points)

Get started with Neural Networks.

Choose a NN implementation (we recommend Sklearn MLPclassifier) 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.

My NN implementation of choice: My NN implementation of choice: I choose the MLPClassifier from Scikit-learn (Sklearn). This implementation allows for the modification of the number of hidden layers and hidden nodes per layer. In MLPClassifier, the logistic sigmoid function can be specified using the activation parameter set to 'logistic'. Additionally, the Levenberg-Marquardt backpropagation (lbfgs) can be used by setting the solver parameter to 'lbfgs'.

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['gender'] == 'female'

clf = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solver='lbf
clf.fit(X_train, y_train)

y_pred = clf.predict(X_train)
accuracy_nn = accuracy_score(y_train, y_pred)

print(accuracy_nn)
```

```
0.8439597315436241
```

Answer: 0.8439597315436241

Question 2.b (0.5 points)

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['gender'] == 'female'

y_pred_test = clf.predict(X_test)

accuracy_test_nn = accuracy_score(y_test, y_pred_test)

print(accuracy_test_nn)
```

```
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
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


clf_scaled = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', solv
clf_scaled.fit(X_train_scaled, y_train)


y_pred_scaled = clf_scaled.predict(X_train_scaled)
accuracy_nn_scaled = accuracy_score(y_train, y_pred_scaled)


y_pred_test_scaled = clf_scaled.predict(X_test_scaled)
accuracy_test_nn_scaled = accuracy_score(y_test, y_pred_test_scaled)

print(accuracy_nn_scaled, accuracy_test_nn_scaled)
```

```
    0.8439597315436241 0.8542713567839196
```

Answer (accuracy on training set): 0.8439597315436241

Answer (accuracy on test set): 0.8542713567839196

Question 3 (1 point)

Many of the remaining features in the dataset 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 one-hot transformed features, train and predict the class of the test set. What was your accuracy using a Neural Network with a single 10 node hidden layer?

```
# YOUR CODE HERE

df_train_encoded = pd.get_dummies(df_train[['year', 'eyecolor', 'exercise']])
df_test_encoded = pd.get_dummies(df_test[['year', 'eyecolor', 'exercise']])

X_train= df_train_encoded
X_test = df_test_encoded
y_train = df_train['gender'] == 'female'
y_test = df_test['gender'] == 'female'

clf_encoded = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', sol
clf_encoded.fit(X_train, y_train)

y_pred_test_encoded = clf_encoded.predict(X_test)
accuracy_test_nn_encoded = accuracy_score(y_test, y_pred_test_encoded)

print(accuracy_test_nn_encoded)
```

```
0.5452261306532663
```

Answer: 0.5452261306532663

Question 4 (3 points)

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
dummy_train_eyecolor = pd.get_dummies(df_train['eyecolor'])
dummy_test_eyecolor = pd.get_dummies(df_test['eyecolor'])


X_train_eyecolor = pd.concat([df_train['height'], dummy_train_eyecolor], axis=1)
X_test_eyecolor = pd.concat([df_test['height'], dummy_test_eyecolor], axis=1)

clf_eyecolor = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', so
clf_eyecolor.fit(X_train_eyecolor, y_train)

y_pred_test_eyecolor = clf_eyecolor.predict(X_test_eyecolor)
accuracy_test_nn_eyecolor = accuracy_score(y_test, y_pred_test_eyecolor)

print(accuracy_test_nn_eyecolor)
```

```
0.8618090452261307
```

Answer: 0.8618090452261307

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
df_train['log_height'] = np.log(df_train['height'])
df_test['log_height'] = np.log(df_test['height'])

dummy_train_eyecolor = pd.get_dummies(df_train['eyecolor'])
dummy_test_eyecolor = pd.get_dummies(df_test['eyecolor'])


X_train_log_eyecolor = pd.concat([df_train['log_height'], dummy_train_eyecolor],
X_test_log_eyecolor = pd.concat([df_test['log_height'], dummy_test_eyecolor], ax

clf_log_eyecolor = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic'
clf_log_eyecolor.fit(X_train_log_eyecolor, y_train)

y_pred_test_log_eyecolor = clf_log_eyecolor.predict(X_test_log_eyecolor)
accuracy_test_nn_log_eyecolor = accuracy_score(y_test, y_pred_test_log_eyecolor)

print(accuracy_test_nn_log_eyecolor)
```

```
0.8618090452261307
```

Answer: 0.8618090452261307

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
mean_height_train = df_train['height'].mean()
std_height_train = df_train['height'].std()

df_train['zscore_height'] = (df_train['height'] - mean_height_train) / std_heigh
df_test['zscore_height'] = (df_test['height'] - mean_height_train) / std_height_


dummy_train_eyecolor = pd.get_dummies(df_train['eyecolor'])
dummy_test_eyecolor = pd.get_dummies(df_test['eyecolor'])

X_train_zscore_eyecolor = pd.concat([df_train['zscore_height'], dummy_train_eyec
X_test_zscore_eyecolor = pd.concat([df_test['zscore_height'], dummy_test_eyecolo

clf_zscore_eyecolor = MLPClassifier(hidden_layer_sizes=(10,), activation='logist
clf_zscore_eyecolor.fit(X_train_zscore_eyecolor, y_train)

y_pred_test_zscore_eyecolor = clf_zscore_eyecolor.predict(X_test_zscore_eyecolor
accuracy_test_nn_zscore_eyecolor = accuracy_score(y_test, y_pred_test_zscore_eye

print(accuracy_test_nn_zscore_eyecolor)
```

```
0.8668341708542714
```

Answer: 0.8668341708542714

Question 5 (1.5 points)

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

Question 5.a

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

```
# YOUR CODE HERE
dummy_train_eyecolor = pd.get_dummies(df_train['eyecolor'])
dummy_test_eyecolor = pd.get_dummies(df_test['eyecolor'])

X_train_playgames_eyecolor = pd.concat([df_train['playgames'], dummy_train_eyeco
X_test_playgames_eyecolor = pd.concat([df_test['playgames'], dummy_test_eyecolor

clf_playgames_eyecolor = MLPClassifier(hidden_layer_sizes=(10,), activation='log
clf_playgames_eyecolor.fit(X_train_playgames_eyecolor, y_train)

y_pred_test_playgames_eyecolor = clf_playgames_eyecolor.predict(X_test_playgames
accuracy_test_nn_playgames_eyecolor = accuracy_score(y_test, y_pred_test_playgam

print(accuracy_test_nn_playgames_eyecolor)
```

```
0.678391959798995
```

Answer: 0.678391959798995

Question 5.b

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

Note: You can drop rows that have 0 in the `playgames` column, in order to avoid -inf values when applying log.

```
# YOUR CODE HERE
df_train = df_train[df_train['playgames'] != 0]
df_test = df_test[df_test['playgames'] != 0]

y_train = df_train['gender'] == 'female'
y_test = df_test['gender'] == 'female'

df_train['log_playgames'] = np.log(df_train['playgames'])
df_test['log_playgames'] = np.log(df_test['playgames'])

dummy_train_eyecolor = pd.get_dummies(df_train['eyecolor'])
dummy_test_eyecolor = pd.get_dummies(df_test['eyecolor'])

X_train_log_playgames_eyecolor = pd.concat([df_train['log_playgames'], dummy_tra
X_test_log_playgames_eyecolor = pd.concat([df_test['log_playgames'], dummy_test_

clf_log_playgames_eyecolor = MLPClassifier(hidden_layer_sizes=(10,), activation=
clf_log_playgames_eyecolor.fit(X_train_log_playgames_eyecolor, y_train)

y_pred_test_log_playgames_eyecolor = clf_log_playgames_eyecolor.predict(X_test_l
accuracy_test_nn_log_playgames_eyecolor = accuracy_score(y_test, y_pred_test_log

print(accuracy_test_nn_log_playgames_eyecolor)
```

```
0.6300813008130082
```

Answer: 0.6300813008130082

Question 5.c

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

```
# YOUR CODE HERE
mean_playgames_train = df_train['playgames'].mean()
std_playgames_train = df_train['playgames'].std()

df_train['zscore_playgames'] = (df_train['playgames'] - mean_playgames_train) /
df_test['zscore_playgames'] = (df_test['playgames'] - mean_playgames_train) / st

dummy_train_eyecolor = pd.get_dummies(df_train['eyecolor'])
dummy_test_eyecolor = pd.get_dummies(df_test['eyecolor'])

X_train_zscore_playgames_eyecolor = pd.concat([df_train['zscore_playgames'], dum
X_test_zscore_playgames_eyecolor = pd.concat([df_test['zscore_playgames'], dummy

clf_zscore_playgames_eyecolor = MLPClassifier(hidden_layer_sizes=(10,), activati
clf_zscore_playgames_eyecolor.fit(X_train_zscore_playgames_eyecolor, y_train)

y_pred_test_zscore_playgames_eyecolor = clf_zscore_playgames_eyecolor.predict(X_
accuracy_test_nn_zscore_playgames_eyecolor = accuracy_score(y_test, y_pred_test_

print(accuracy_test_nn_zscore_playgames_eyecolor)
```

```
0.6463414634146342
```

Answer: 0.6463414634146342

Question 6 (2 points)

Combine the features from question 3, 4, and 5 (`year`, `eyecolor`, `exercise`, `height`, `playgames`). 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

dummy_train = pd.get_dummies(df_train[['year', 'eyecolor', 'exercise']])
dummy_test = pd.get_dummies(df_test[['year', 'eyecolor', 'exercise']])

df_train['zscore_playgames'] = (df_train['playgames'] - mean_playgames_train) /
df_test['zscore_playgames'] = (df_test['playgames'] - mean_playgames_train) / st

df_train['log_height'] = np.log(df_train['height'])
df_test['log_height'] = np.log(df_test['height'])

X_train_combined = pd.concat([df_train['log_height'], df_train['zscore_playgames
X_test_combined = pd.concat([df_test['log_height'], df_test['zscore_playgames'],

clf_combined = MLPClassifier(hidden_layer_sizes=(10,), activation='logistic', so
clf_combined.fit(X_train_combined, y_train)

y_pred_test_combined = clf_combined.predict(X_test_combined)
accuracy_test_nn_combined = accuracy_score(y_test, y_pred_test_combined)

print(accuracy_test_nn_combined)
```

```
0.7967479674796748
```

Answer: 0.7967479674796748

Question 7- Bonus (1 point)

Can you improve your test set prediction accuracy by 3% or more? See how close to that milestone of improvement you can get by modifying the hyperparameters of Neural Networks (the number of hidden layers, number of hidden nodes in each layer, the learning rate, the type of activation function etc.).

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 more generally to any ML technique with tuning parameters.

Please give your new prediction accuracy on the test set, and write a paragraph in a text cell below with an explanation of your approach and evaluation metrics.

```
# YOUR CODE HERE

param_grid = {
    'hidden_layer_sizes': [(10,), (20,), (30,), (10, 10), (20, 20)],
    'activation': ['logistic', 'relu', 'tanh'],
    'solver': ['lbfgs', 'adam'],
    'learning_rate_init': [0.001, 0.01, 0.1],
    'max_iter': [100, 200, 300]
}


grid_search = GridSearchCV(MLPClassifier(random_state=42), param_grid, cv=5, sco


grid_search.fit(X_train_combined, y_train)


best_params = grid_search.best_params_


clf_best = MLPClassifier(**best_params, random_state=42)
clf_best.fit(X_train_combined, y_train)


y_pred_test_best = clf_best.predict(X_test_combined)
accuracy_test_nn_best = accuracy_score(y_test, y_pred_test_best)

print(accuracy_test_nn_best)
```

```
Fitting 5 folds for each of 270 candidates, totalling 1350 fits
0.8211382113821138
```

Answer: 0.8211382113821138

Explanation: To improve the accuracy of the neural network model, I utilized grid search with cross-validation. This method systematically works through multiple combinations of hyperparameter tunes, cross-validating as it goes to determine which tune gives the best performance. The hyperparameters we tuned include the number of hidden layers, number of hidden nodes in each layer, the learning rate, the type of activation function, and the solver. By optimizing these hyperparameters, we were able to achieve a better model performance on the test set compared to the initial model.