Lab 4: SVM + 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://people.ischool.berkeley.edu/~zp/course_datasets/lab_4_training.csv
!wget http://people.ischool.berkeley.edu/~zp/course_datasets/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()
```

```
--2019-02-28 20:32:41--  http://people.ischool.berkeley.edu/~zp/course_datasets
Resolving people.ischool.berkeley.edu (people.ischool.berkeley.edu)... 128.32.7
Connecting to people.ischool.berkeley.edu (people.ischool.berkeley.edu)|128.32.
HTTP request sent, awaiting response... 200 OK
Length: 105581 (103K) [text/csv]
Saving to: ‘lab_4_training.csv.2’
```

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
from sklearn.metrics import confusion_matrix
df_train.groupby("gender").size()
tn, fp, fn, tp = confusion_matrix(df_train["gender"].tolist(),["female" for i in np.ar
tn, fp, fn, tp
855.0/(855.0+735.0)
```

```
0.5377358490566038
```

ANSWER: The majority class is the female, if we always predicted this class in the training set, we will get an accuracy of 0.5377358490566038

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_test.groupby("gender").size()
208.0/(208.0+190.0)
```

```
0.5226130653266332
```

ANSWER: In the test set, we will get accuracy of 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/logistc) function as the transfer function (another word for activation function) for the output node and use Levenberg-Marquardt backpropagation (lbfgs). It is possible to specify logsig or 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 using the height feature from the training set. What was the accuracy of this prediction?

```
# YOUR CODE HERE
import sklearn
from sklearn.tree import DecisionTreeClassifier
from sklearn.feature_extraction import DictVectorizer #to turn categorial variables ir
from sklearn import preprocessing #to transform the feature labels
from sklearn.neural_network import MLPClassifier

df_train
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
gender.transform(df_train["gender"])
df_train["gender_trans"]=gender.transform(df_train["gender"])

X = df_train["height"].reshape(-1, 1)
y = df_train["gender_trans"]
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(X,y)
clf.predict(df_train["height"].reshape(-1, 1) )
clf.score(df_train["height"].reshape(-1, 1),df_train["gender_trans"])
```

```
0.8465408805031447
```

ANSWER: Accuracy: 0.8465408805031447

Question 2.b

Take the trained model from question 2.b 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
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
gender.transform(df_test["gender"])
df_test["gender_trans"]=gender.transform(df_test["gender"])
clf.predict(df_test["height"].reshape(-1, 1) )
clf.score(df_test["height"].reshape(-1, 1),df_test["gender_trans"])
```

```
0.8542713567839196
```

ANSWER: The accuracy of the test set is 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.c with the log version and the normalized and centered version of this feature

```
# YOUR CODE HERE
#Log
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
df_train["gender_trans"]=gender.transform(df_train["gender"])
df_test["gender_trans"]=gender.transform(df_test["gender"])


clf.fit(np.log(df_train["height"].reshape(-1, 1)), df_train["gender_trans"])

print(clf.score(np.log(df_train["height"].reshape(-1, 1)),df_train["gender_trans"]))
print(clf.score(np.log(df_test["height"].reshape(-1, 1)),df_test["gender_trans"]))
```

```
0.8465408805031447
0.8542713567839196
```

```
#standard normalization
scaler = preprocessing.StandardScaler()
scaler.fit(df_train["height"].reshape(-1, 1))
X = scaler.transform(df_train["height"].reshape(-1, 1))
y = df_train["gender_trans"]

T = preprocessing.MinMaxScaler()
T.fit(X,y)
X2=T.transform(X)

clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(X2,y)

print(clf.score(X2, df_train["gender_trans"]))
print(clf.score(T.transform(scaler.transform(df_test["height"].reshape(-1, 1))),df_tes
#train_score
```

```
0.8465408805031447
0.8542713567839196
```

ANSWER: Both the log transformation and normalization gives the training data has accuracy of 0.8465408805031447 and the log transformation of test data has accuracy of 0.8542713567839196.

Question 3

The rest of features in this dataset barring a few are categorical. Neither 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.

```
# df_train["gender"]=df_train["gender_trans"]
# df_train["eyecolor"].unique()



# df_train["exercise"].unique()



# eyecolors = preprocessing.LabelEncoder()
# eyecolors.fit(['green', 'other', 'hazel', 'brown', 'blue'])
# #df_train["eyecolor"]=eyecolors.transform(df_train["eyecolor"])
# year = preprocessing.LabelEncoder()
# year.fit(['second', 'third', 'fourth', 'other', 'first', 'first"'])
# #df_train["year"]=year.transform(df_train["year"])
# exercise=preprocessing.LabelEncoder()
# df_train["exercise"].unique()
# exercise.fit(['Yes', 'No'])
# df_train["exercise"]=exercise.transform(df_train["exercise"])
# df_train

pd.concat([pd.get_dummies(df_train["eyecolor"]), pd.get_dummies(df_train["year"]),pd.g
pd.concat([pd.get_dummies(df_test["eyecolor"]), pd.get_dummies(df_test["year"]),pd.get
```

13	0	0	1	0	0	0	0	1	0	0
14	0	1	0	0	0	0	0	0	0	1
15	1	0	0	0	0	0	0	0	0	0
16	1	0	0	0	0	0	0	1	0	0
17	0	0	0	1	0	0	0	0	0	1
18	0	0	0	1	0	0	0	1	0	0
19	1	0	0	0	0	0	0	0	1	0
20	0	1	0	0	0	1	0	0	0	0
21	0	0	1	0	0	0	0	0	0	0
22	0	1	0	0	0	0	0	0	0	0
23	1	0	0	0	0	0	0	0	0	1
24	1	0	0	0	0	0	0	0	0	1
25	0	0	0	1	0	1	0	0	0	0

Question 3.a

What was your accuracy using Neural Network with a single 10 node hidden layer? During training, use a maximum number of iterations of 50. (Expected training time: ~15 mins)

```
# YOUR CODE HERE
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
feature=pd.concat([pd.get_dummies(df_train["eyecolor"]), pd.get_dummies(df_train["year

clf.fit(feature,df_train["gender_trans"])
print(clf.score(feature,df_train["gender_trans"]))


gender.transform(df_test["gender"])
df_test["gender_trans"]=gender.transform(df_test["gender"])
print(clf.score(pd.concat([pd.get_dummies(df_test["eyecolor"]), pd.get_dummies(df_test
```

```
0.5691823899371069
0.6105527638190955
```

ANSWER: Training data set has accuracy of 0.5691823899371069 and test data set has accuracy 0.6105527638190955

Question 4

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

380	1	0	0	0	0	0	0	0	0	1

Question 4.a

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

```
# YOUR CODE HERE
eyecolors = preprocessing.LabelEncoder()
eyecolors.fit(['green', 'other', 'hazel', 'brown', 'blue'])
df_train["eyecolor"]=eyecolors.transform(df_train["eyecolor"])
df_test["eyecolor"]=eyecolors.transform(df_test["eyecolor"])



df_train[["height","eyecolor"]]
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(df_train[["height","eyecolor"]],df_train["gender_trans"])
clf.score(df_test[["height","eyecolor"]],df_test["gender_trans"])
```

```
0.8542713567839196
```

ANSWER: Accuracy on test data 0.8542713567839196

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 eye color as a one-hot?

```
# YOUR CODE HERE

p=df_train[["eyecolor","height"]]
p["height"]=np.log(p["height"])
q=df_test[["eyecolor","height"]]
q["height"]=np.log(q["height"])
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(p,df_train["gender_trans"])
clf.score(q,df_test["gender_trans"])
```

```
0.8592964824120602
```

ANSWER: The test accuracy is 0.8592964824120602

Question 4.c

What is the accuracy on the test set using the Z-score of height values and eye color 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

scaler = preprocessing.StandardScaler()
scaler.fit(df_train["height"].reshape(-1, 1))
X = scaler.transform(df_train["height"].reshape(-1, 1))
zscore=df_train[["eyecolor"]]
zscore["z"]=X
zscore2=df_test[["eyecolor"]]
```

```
zscore2["z"]=scaler.transform(df_test["height"].reshape(-1, 1))
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(zscore,df_train["gender_trans"])
clf.score(zscore2,df_test["gender_trans"])
```

```
0.8693467336683417
```

ANSWER: The test set has accuracy of 0.8693467336683417

Question 5

Repeat question 5 for exercise hours + eye color

```
# YOUR CODE HERE
#Repeat 4a

df_train[["exercisehours","eyecolor"]]
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(df_train[["exercisehours","eyecolor"]],df_train["gender_trans"])
print(clf.score(df_test[["exercisehours","eyecolor"]],df_test["gender_trans"]))

 #repeat 4b

p=df_train[["eyecolor","exercisehours"]]
p=p[p['exercisehours']!=0.0]
p["exercisehours"]=np.log(p["exercisehours"])

q=df_test[["eyecolor","exercisehours"]]
q=q[q['exercisehours']!=0.0]
q["exercisehours"]=np.log(q["exercisehours"])
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(p,df_train[df_train['exercisehours']!=0.0]["gender_trans"])
print(clf.score(q,df_test[df_train['exercisehours']!=0.0]["gender_trans"]))

#repeat 4c
scaler = preprocessing.StandardScaler()
scaler.fit(df_train["exercisehours"].reshape(-1, 1))
X = scaler.transform(df_train["exercisehours"].reshape(-1, 1))
zscore=df_train[["eyecolor"]]
zscore["z"]=X
zscore2=df_test[["eyecolor"]]
zscore2["z"]=scaler.transform(df_test["exercisehours"].reshape(-1, 1))
clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=1,
clf.fit(zscore,df_train["gender_trans"])
print(clf.score(zscore2,df_test["gender_trans"]))
```

```
0.5728643216080402
0.606694560669456
0.5628140703517588
```

ANSWER:

Test sets accuracy is

0.5728643216080402

0.606694560669456

0.5628140703517588

Question 6

Combine the features from question 4, 5, and exercise hours from question 6 (using the best normalization feature set form questions 5 and 6)

Question 6.a

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

```
# YOUR CODE HERE
#Height, eyecolor, exercise hours
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
Q6=df_train[df_train['exercisehours']!=0.0]
Q6["gender_trans"]=gender.transform(Q6["gender"])
a=Q6[["eyecolor","exercisehours","height","gender_trans"]]

a["height"]=np.log(a["height"])
a["exercisehours"]=np.log(a["exercisehours"])

Q7=df_test[df_test['exercisehours']!=0.0]
b=Q7[["eyecolor","exercisehours","height"]]
b["height"]=np.log(b["height"])
b["exercisehours"]=np.log(b["exercisehours"])

b["gender_trans"]=gender.transform(Q7["gender"])



clf = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(10), random_state=
clf.fit(a[["eyecolor","exercisehours","height"]],a["gender_trans"])
print(clf.score(b[["eyecolor","exercisehours","height"]], b["gender_trans"]))
```

```
0.8451882845188284
```

ANSWER: The accuracy of the test data is 0.8451882845188284

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
# clf = (solver='lbfgs', alpha=1e-5, hidden_layer_sizes=10, random_state=10,activation
# clf.fit(a[["height","eyecolor","exercisehours"]],a["genders"])
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import RandomizedSearchCV
from scipy.stats import uniform as sp_rand
from sklearn.model_selection import GridSearchCV
def svc_param_selection(X, y, nfolds):
    Cs = [0.001, 0.01, 0.1, 1, 10]
    gammas = [0.001, 0.01, 0.1, 1,3]
    param_grid = {'C': Cs, 'gamma' : gammas}
    grid_search = GridSearchCV(svm.SVC(kernel='rbf'), param_grid, cv=nfolds)
    grid_search.fit(X, y)
    grid_search.best_params_
    return grid_search.best_params_


svc_param_selection(a,a["gender_trans"],5)
p=SVC(gamma=50, C=10000000,random_state=50)
p.fit(a[["eyecolor","exercisehours","height"]],a["gender_trans"])
a=p.score(a[["eyecolor","exercisehours","height"]],a["gender_trans"])
print(a*100)
```

```
89.88877654196158
```

ANSWER: I've tried a lot of method and approaches. I first try to use xgboost to find the best features that can predict the gender. Then I use GridSearchCV to find the best hyperparameter for the transformed features I had. I noticed that increasing in gamma which is the kernel coefficient and Penalty parameter would increase the accuracy of the prediction even though the run time would also increase. Since we are only using 3 features, it won't take too long and the prediciton accuracy is now 89.888% which increases by slightly more than 5%

Question 8 (Bonus: 20%)

Get started with Support Vector Machines.

Chose a SVM implementation and specify which you choose. Be sure the implementation allows you to choose between linear and RBF kernels.

Question 8.a

Use the same dataset from 2.a using the linear kernel to find training set prediction accuracy.

```
# YOUR CODE HERE
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
df_train["gender_trans"]=gender.transform(df_train["gender"])

X = df_train["height"].reshape(-1, 1)
y = df_train["gender_trans"]

clf = SVC(10**(3), kernel = "linear")
clf.fit(X, y)
clf.score(df_train["height"].reshape(-1, 1),df_train["gender_trans"])
```

```
0.8465408805031447
```

ANSWER: Accuracy for training set is 0.8465408805031447

Question 8.b

Use the same dataset from 2.a using the linear kernel to find test set prediction accuracy

```
# YOUR CODE HERE
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
#df_test["gender_trans"]=gender.transform(df_train["gender"])

X = df_test["height"].reshape(-1, 1)
y = df_test["gender_trans"]

clf.score(X,y)
```

```
0.8542713567839196
```

ANSWER: The accuracy of the test sets is 0.8542713567839196

Question 8.c

Use the same dataset from 2.a using the RBF kernel to find training set prediction accuracy

```
# YOUR CODE HERE
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
df_train["gender_trans"]=gender.transform(df_train["gender"])

X = df_train["height"].reshape(-1, 1)
y = df_train["gender_trans"]

clf = SVC(10**(3), kernel = "rbf")
clf.fit(X, y)
clf.score(df_train["height"].reshape(-1, 1),df_train["gender_trans"])
```

```
0.8465408805031447
```

ANSWER: The accuracy of the test data set is 0.8465408805031447

Question 8.d

Use the same dataset from 2.a using the RBF kernel to find test set prediction accuracy

```
# YOUR CODE HERE
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
#df_test["gender_trans"]=gender.transform(df_train["gender"])

X = df_test["height"].reshape(-1, 1)
y = df_test["gender_trans"]

clf.score(X,y)
```

```
0.8542713567839196
```

ANSWER: The accuracy of the test data is 0.8542713567839196

Question 8.e

Use the same dataset from 2.c (log) using the RBF to find test set prediction accuracy

```
# YOUR CODE HERE
gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
df_train["gender_trans"]=gender.transform(df_train["gender"])

X = np.log(df_train["height"].reshape(-1, 1) )
y = df_train["gender_trans"]

clf = SVC( kernel = "rbf")
clf.fit(X, y)

X2 = np.log(df_test["height"].reshape(-1, 1) )
y2 = df_test["gender_trans"]

clf.score(X2,y2)
```

```
0.8542713567839196
```

ANSWER: The test set has accuracy of 0.8542713567839196

Question 8.f

Z-score is a normalization technique. 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. Repeat question 3.e using Z-score and note if there is any difference in accuracy and comment on why there is a change or no change in accuracy

```
# YOUR CODE HERE
scaler = preprocessing.StandardScaler()
scaler.fit(df_train["height"].reshape(-1, 1))
X = scaler.transform(df_train["height"].reshape(-1, 1))

gender = preprocessing.LabelEncoder()
gender.fit(["male","female"])
df_train["gender_trans"]=gender.transform(df_train["gender"])

y = df_train["gender_trans"]

clf2 = SVC( kernel = "rbf")
clf2.fit(X, y)

X2 = scaler.transform(df_test["height"].reshape(-1, 1))
y2 = df_test["gender_trans"]

clf2.score(X2,y2)
```

```
0.8542713567839196
```

ANSWER: The test has accuracy of 0.8542713567839196. They have the same prediction. Because SVM will nonlinearly maps samples into a higher dimensional space and can handle the relation between class that are nonlinear. This kernel nonlinearly maps samples into a higher dimensional space so it, unlike the linear kernel, can handle the case when the relation between class labels and attributes is nonlinear. Log transformation also map the data onto a different scale but they both don't change the distribution. Therefore, the prediction remained the same.