DSP 556 Assignment_2

Sheikh-Sedat Touray

Importing all the libraries

In [279]:
```
# Importing Libraries

import pandas as pd
import numpy as np
np.set_printoptions(formatter={'float_kind':"{:3.2f}".format})
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold, train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestClassifier
import sklearn.metrics as skm
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from math import sqrt
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LogisticRegression
```

The code below reads in the bike share dataset as a dataframe

In [280]:
```
## 1. Find cross validated rmse for a LM using 5 folds using the entir
e data
# 2. Reading the bikes csv as a dataframe
df = pd.read_csv("bikes.csv")
```

The code below allows us to inspect the first five rows and all the columns

In [281]:
```
# 3. Visually inspecting the data using the head method
df.head()
```

Out[281]:

	date	season	year	month	day_of_week	weekend	holiday	temp_actual	temp_feel	humidi
0	2011-01-01	winter	2011	Jan	Sat	True	no	57.399525	64.72625	80.583
1	2011-01-03	winter	2011	Jan	Mon	False	no	46.491663	49.04645	43.727
2	2011-01-04	winter	2011	Jan	Tue	False	no	46.760000	51.09098	59.043
3	2011-01-05	winter	2011	Jan	Wed	False	no	48.749427	52.63430	43.695
4	2011-01-07	winter	2011	Jan	Fri	False	no	46.503324	50.79551	49.869

In the code below we use the onehot encoder to map to numbers the categorical data for the features

In [282]:
```
#Initialize the onehot encoder and store it to the ohe variable
ohe = OneHotEncoder()
```

In [283]:
```
# Now we fit and transform the categorial variables with the onehot en
coder
ohe.fit_transform(df[['season','month','day_of_week','weekend','holida
y','weather_cat']]).toarray()
f1 = ohe.fit_transform(df[['season','month','day_of_week','weekend','h
oliday','weather_cat']]).toarray()
```

In [284]:
```
# I used this to inspect the categories of the encoded categorical dat
a
ohe.categories_
```

Out[284]:
```
[array(['fall', 'spring', 'summer', 'winter'], dtype=object),
 array(['Apr', 'Aug', 'Dec', 'Feb', 'Jan', 'Jul', 'Jun', 'Mar', 'May',
        'Nov', 'Oct', 'Sep'], dtype=object),
 array(['Fri', 'Mon', 'Sat', 'Sun', 'Thu', 'Tue', 'Wed'], dtype=objec
t),
 array([False,  True]),
 array(['no', 'yes'], dtype=object),
 array(['categ1', 'categ2', 'categ3'], dtype=object)]
```

In [285]:
```
# Store the categories to a variable
f1_labels = ohe.categories_
np.array(f1_labels,dtype = object).ravel()
```

Out[285]:
```
array([array(['fall', 'spring', 'summer', 'winter'], dtype=object),
       array(['Apr', 'Aug', 'Dec', 'Feb', 'Jan', 'Jul', 'Jun', 'Mar',
'May',
       'Nov', 'Oct', 'Sep'], dtype=object),
       array(['Fri', 'Mon', 'Sat', 'Sun', 'Thu', 'Tue', 'Wed'], dtype=
object),
       array([False,  True]), array(['no', 'yes'], dtype=object),
       array(['categ1', 'categ2', 'categ3'], dtype=object)], dtype=obj
ect)
```

In [286]:
```
#I ravel and concatenate them as single, array to keep them in one num
py array instead of multiple arrays
f1_labels = np.asarray(f1_labels, dtype = object).ravel()
f1_labels = np.concatenate(f1_labels)
```

In [287]:
```
# I store the numerical data to variable and named it f2
f2 = df.drop(['date','season','month','day_of_week','weekend','holida
y','weather_cat','rides'],axis=1)
f2.head()
```

Out[287]:

	year	temp_actual	temp_feel	humidity	windspeed
0	2011	57.399525	64.72625	80.5833	10.749882
1	2011	46.491663	49.04645	43.7273	16.636703
2	2011	46.760000	51.09098	59.0435	10.739832
3	2011	48.749427	52.63430	43.6957	12.522300
4	2011	46.503324	50.79551	49.8696	11.304642

In [288]:
```
# I created a data frame out of the encoded categorical data and named
it f1 and inspected the first 5 rows
f1=pd.DataFrame(f1,columns = f1_labels)
f1.head()
```

Out[288]:

	fall	spring	summer	winter	Apr	Aug	Dec	Feb	Jan	Jul	...	Thu	Tue	Wed	False	True
0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	1.0
1	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	1.0	0.0	1.0	0.0
3	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	1.0	1.0	0.0
4	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	1.0	0.0

5 rows × 30 columns

1. RMSE of CV on Entire Dataset

In [289]:
```
#get data and concatenating f1 and f2 will give me all my predictors
features  = pd.concat([f1,f2],axis=1)

#response varaible is ride
target = df['rides']

# start making the linear regession model
lm = LinearRegression()

# The rmse of my CV and we used the absolutue function to deal with ne
gatives
score = cross_val_score(lm, features, target, cv=5, scoring = 'neg_mea
n_absolute_error')
print(" 5-Fold CV RMSE: {:.2f}".format(np.mean(np.abs(score))))
```

```
 5-Fold CV RMSE: 531.99
```

RMSE for Training Vs Test not necessary but it is nice to see and compare it with the CV RMSE

In [290]:
```
# SPlit the training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target,
train_size=0.8, test_size=0.2, random_state=3)
```

In [291]:
```
# fit the linear model on the training set
lm = lm.fit(X_train,y_train)
```

In [292]:
```
# Predit unseen data
y_pred = lm.predict(X_test)
#test the model mse
mse = skm.mean_squared_error(y_test,y_pred)
# we take the square root of the MSE to get the RMSE
rmse = print("MSE Score for Test data: {:.2f}".format(sqrt(mse)))
```

```
MSE Score for Test data: 566.09
```

Above is the RMSE for training vs test data

2. Find the Cross Validation Scores First using the Default score and then AUC

In [293]:
```
# 2. Reading the default csv as a dataframe
df2 = pd.read_csv("default.csv")
```

In [294]:
```
# 2i. Visually inspecting the data using the head method
df2.head()
```

Out[294]:

	Unnamed: 0	default	student	balance	income
0	1	No	No	729.526495	44361.625074
1	2	No	Yes	817.180407	12106.134700
2	3	No	No	1073.549164	31767.138947
3	4	No	No	529.250605	35704.493935
4	5	No	No	785.655883	38463.495879

In [295]:
```
# Splitting the dataset into features and target variables
x2 = df2[['balance','income']]

#Below is default the target
y = df2['default']
```

In [296]:
```
# Encode the output to avoid errors
lb = LabelEncoder()
y=lb.fit_transform(y)
y
```

Out[296]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [297]:
```
# USe one hot encoder for categorical predictors and store them in a v
ariable
ohe.fit_transform(df2[['student']]).toarray()
x1 = ohe.fit_transform(df2[['student']]).toarray()
```

In [298]:
```
# Visualize the labels
ohe.categories_
```

Out[298]:
```
[array(['No', 'Yes'], dtype=object)]
```

In [299]:
```
# store the labels and turn them into a single numpy array
x1_labels = ohe.categories_
np.array(x1_labels)
```

Out[299]:
```
array([['No', 'Yes']], dtype=object)
```

In [300]:
```
# I turned the array into a Data frame and visualized the first 5 rows
x1=pd.DataFrame(x1,columns = x1_labels)
x1.head()
```

Out[300]:

	No	Yes
0	1.0	0.0
1	0.0	1.0
2	1.0	0.0
3	1.0	0.0
4	1.0	0.0

In [301]:
```
# concatenate the numerical and encoded categorical data into one data frame
x= pd.concat([x1,x2],axis=1)
```

In [302]:
```
# Initialize the model with k=5
klm = KNeighborsClassifier(n_neighbors=5)
#fit the model on the entire data set
model = klm.fit(x,y)
```

The default score is the Accuracy score which is printed below

In [303]:
```
# do the 5-fold cross validation
scored = cross_val_score(model, x, y, cv=5)
print("5-Fold DefaultScore: {}".format(scores))
print('\n')
print("5-Fold mean DefaultScore: {:3.2f}".format(np.mean(np.abs(score
d))))
```

```
5-Fold DefaultScore: [0.97 0.97 0.97 0.97 0.96]


5-Fold mean DefaultScore: 0.97
```

Now we use the Area under the Curve, AUC score in the CV

In [304]:
```
# do the 5-fold cross validation
scores = cross_val_score(model, x, y, cv=5, scoring= 'roc_auc')
print("5-Fold ROC_AUC: {}".format(scores))
print('\n')
# Print a mean score for all the folds
print("5-Fold mean ROC_AUC: {:3.2f}".format(np.mean(np.abs(scores))))
```

```
5-Fold ROC_AUC: [0.79 0.79 0.80 0.80 0.77]


5-Fold mean ROC_AUC: 0.79
```

Repeating number 2 on the train vs Test not necessary for this assignment but it was not to see difference

In [305]:
```
# SPlit the training and testing sets
X_train1, X_test1, y_train1, y_test1 = train_test_split(x, y, train_si
ze=0.8, test_size=0.2, random_state=3)

# USe the scalar function to nor
sc = StandardScaler()
X_train1 = sc.fit_transform(X_train1)
X_test1 = sc.fit_transform(X_test1)
```

In [306]:
```
# Initializing the model
klm = KNeighborsClassifier(n_neighbors=5)
#fitting model on train set
model = klm.fit(X_train1,y_train1)

# do the 5-fold cross validation on train set
scores = cross_val_score(model, X_train1,y_train1, cv=5)
print("Fold Accuracies: {}".format(scores))
```

```
Fold Accuracies: [0.97 0.97 0.97 0.97 0.96]
```

In [307]:
```
#Predicting on unseen data
pred_y = model.predict(X_test1)
# AUC on test
scoring = roc_auc_score(y_test1,pred_y)
print("ROC and AUC Score: {:3.2f}".format(scoring))
```

```
ROC and AUC Score: 0.72
```

3. Using Grid Search to choose best parameters and get Accuracies

In [308]:
```
# KNN
modeln = KNeighborsClassifier()

# do the 10-fold cross validation and shuffle the data
cv = KFold(n_splits=10,  shuffle = True)

# grid search
param_grid = {'n_neighbors': list(range(1,31))}
grid = GridSearchCV(modeln, param_grid, cv=cv)

# performing grid search
grid.fit(X_train1, y_train1)
print("Grid Search: best parameters: {}".format(grid.best_params_))
```

```
Grid Search: best parameters: {'n_neighbors': 17}
```

In [309]:
```
# accuracy of best model with confidence interval
y_pred = grid.best_estimator_.predict(X_test1)
Auc = roc_auc_score(y_test1, y_pred)
#lb,ub = classification_confint(Auc,X_test.shape[0])
print("Accuracy: {:3.2f} ".format(Auc))
```

```
Accuracy: 0.71
```

In [310]:
```
# best tuning is k=9
modelb = KNeighborsClassifier(n_neighbors=9)
modelb = modelb.fit(X_train1,y_train1)

#make a prediction
bestpred = modelb.predict(X_test1)
#Storing AUC for the prediction on test and printing score
Aucbest = roc_auc_score(y_test1, bestpred)
print("Accuracy: {:3.2f} ".format(Aucbest))
```

```
Accuracy: 0.72
```

From the accuracy above we can see that K = 9 is giving best accuracy score

4. Using the ischemic data to find Accuracy, AUC and Confusion Matrix

In [311]:
```
# 4. Using the Ischemic data set fit a random forest
df3 = pd.read_csv("ischemic.csv")
```

In [312]:
```
# looking at the first 5 rows
df3.head()
```

Out[312]:

	stroke	nascet_scale	calc_vol	calc_vol_prop	matx_vol	matx_vol_prop	lrnc_vol	lrnc
0	no	0	235.252599	0.070443	3156.834690	0.759958	224.871710	
1	no	0	31.433595	0.016165	3032.860796	0.813306	368.560663	
2	no	0	113.404823	0.038081	3835.220140	0.782526	321.158928	
3	yes	0	780.823789	0.213432	3518.876937	0.761089	140.517346	
4	no	0	84.055774	0.041384	2990.273268	0.749869	293.269922	

5 rows × 29 columns

In [313]:
```
#Splitting the data set into predictors and a response variable
feat = df3.drop(['stroke'],axis=1)
targ = df3['stroke']
```

In [314]:
```
#map the response variable to numbers by encoding it
targ=lb.fit_transform(targ)
targ
```

Out[314]:
```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
0,
       1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1,
0,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0,
1,
       1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1,
1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0,
0,
       1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1])
```

In [315]:
```
# SPlit the training and testing sets
X_traint, X_testt, y_traint, y_testt = train_test_split(feat, targ, tr
ain_size=0.8, test_size=0.2, random_state=3)
```

In [316]:
```
# Random Forest Model
forest = RandomForestClassifier()
forest = forest.fit(X_traint,y_traint)
```

In [317]:
```
#Predicting with the randomforest and printing out the scores
pred_y1 = forest.predict(X_testt)
print("ROC and AUC Score: {:3.2f}".format(roc_auc_score(y_testt,pred_y
1)))
print("ACCURACY Score: {:3.2f}".format(accuracy_score(y_testt,pred_y
1)))
```

```
ROC and AUC Score: 0.65
ACCURACY Score: 0.65
```

In [321]:
```
#setting threshold
mythreshold = 0.7
#comparing prediction with threshold
pred_y2 = (pred_y1 >= mythreshold).astype(int)
#creating confusion matrix
cm=confusion_matrix(y_testt,pred_y2)
#Defining the positions for the true positives and negatives and the f
alse positives and negatives
TP = cm[0][0]
TN = cm[0][1]
FN = cm[1][0]
FP = cm[0][1]

#Printing the confusion matrix
print("Confusion Matrix:\n{}".format(cm))

#print a space below
print()

# computing and printing sensitivity and Specificity values
sensitivity = TP/(TP+FN)
print("sensitivity:{:3.2f}".format(sensitivity))
Specificity = TN/(TN+FP)
print("Specificity:{:3.2f}".format(Specificity))
```

```
Confusion Matrix:
[[8 4]
 [5 9]]

sensitivity:0.62
Specificity:0.50
```

In [322]:
```
# USe the scalar function to nor
sc = StandardScaler()
X_traint = sc.fit_transform(X_traint)
X_testt = sc.fit_transform(X_testt)
```

In [323]:
```
# I am going to create a function fot the models I am going to use
def modelx(X_traint,y_traint):

    #logistic regression
    log = LogisticRegression(random_state=0)
    log.fit(X_traint,y_traint)

    #Decision Tree
    tree = DecisionTreeClassifier(random_state=0)
    tree.fit(X_traint,y_traint)

    #MLP Classifier
    mlp = MLPClassifier(random_state=0)
    mlp.fit(X_traint,y_traint)

    # Support Vector Machine
    svc = SVC(random_state=0)
    svc.fit(X_traint,y_traint)

    #returning the models
    return log, tree, mlp, svc
```

In [324]:
```
#creating a new variable to use in my for loop
modelx = modelx(X_traint,y_traint)
```

```
/usr/local/lib64/python3.6/site-packages/sklearn/neural_network/_multi
layer_perceptron.py:617: ConvergenceWarning: Stochastic Optimizer: Max
imum iterations (200) reached and the optimization hasn't converged ye
t.
  % self.max_iter, ConvergenceWarning)
```

In [325]:
```
# Testing the accuracy of the models on unseen data
for i in range(len(modelx)):
    print('modelx', i)
    scorex = roc_auc_score(y_testt, modelx[i].predict(X_testt))
    print("AUC Scores: {:3.2f}".format(scorex))
```

```
modelx 0
AUC Scores: 0.81
modelx 1
AUC Scores: 0.72
modelx 2
AUC Scores: 0.68
modelx 3
AUC Scores: 0.80
```

Above we can see that the logistic regression and the support vector machine give the higest AUC values

In [ ]:
```

```