Assignment 3

Title : Perform the categorization of dataset.

Theory :

Classification is a large domain in the field of statistics and machine learning.

1. Binary classification, where we wish to group an outcome into one of two group.
2. Multi-classification, where we wish to group an outcome into one of multiple (more than two) groups.

In this post, the main focus will be on using a variety of classification algorithm across both of these domains, less emphasis will be placed on the theory behind them.

We can use libraries in Python such as scikit-learn for machine learning models, and Pandas to import data as data frames.

These can easily be installed & imported into Python with pip:

```
$ python 3 - m pip install sklearn
$ python 3 - m pip install pandas
import sklearn as sk
import pandas as pd
```

Binary Classification

For binary classification we are interested in classifying data into one of two binary group - these are usually represented as 0's & 1's in our data.

We will look at data regarding coronary heart disease (CHD) in South Africa.

The goal is to use different variables such as tobacco usage, family history, Idl cholestrol levels, alcohol usage, obesity & more.

A full description of this dataset is available in the "Data" section of the Elements of Statistical learning website.

The code below reads the data into a pandas data frame & then separates the data frame into a y vector of the response and an x matrix of explanatory variables:

Logistic Regression:-
It is a type of Generalized linear Model (GLM) that uses a logistic function to model variable based on any kind of independent variables.
To fit a binary logistics regression with sklearn, we use the logistic Regression module with multi-class set to "over" & fit x & y.
We can then use the predict method to predict probabilities of new data, as well as the score method to get the mean prediction accuracy:

Support vector Machines:
Support vector Machines (SVMs) are a type of classification algorithm that are more flexible - they can do linear classification, but can use other non-linear kernis functions. The following ex uses a linear classifier to fit a hyperplane that separates the data into two classes:

Random Forests
Random Forests are an ensemble learning method that fit multiple Decision Trees on subsets of the data and average that results. We can again fit them using sklearn, and use them to predict outcomes, as well as get mean prediction accuracy:

```
import sklearn as sk
From sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
RF.Fit(x, y)
RF.predict(x.iloc[460:, :])
round(RF.score(x, y), 4)
  0.7338
```

Neural Network

Neural Network are a machine learning algorithm that involves fitting many hidden layers used to represent neurons that are connected with synaptic activation functions. These essentially use a very simplified model of the brain to model & predict data.

We use sklearn for consistency in this post, however libraries such as TensorFlow and Keras are more suited to fitting & ~~com~~ customizing neural network, of which there are a few varieties used for different purpose:

Multi-Class Classification

While binary classification ~~along~~ alone is incrediably useful, there are times when we would like to model and predict data that has more than two classes. Many of the same algorithms can be used with slight modifications.
Additionally, it is common to split data into training & test sets. This means we use a certain portion of the data to fit the model (the training set) and save the remaining portion of it to evaluate to the predictive accuracy of the fitted model (the test set).
There's no official rule to follow when deciding on a split proportion, though in most cases you'd want about 70% to be dedicated for the training set and around 30% for the test set.
To explore both multi-class classifications, as well as training/test data, we will look at another dataset from the Elements of Statistical Learning website. This is data used to determine which one of eleven vowel sounds were spoken:

We will now fit models & test them as is normally done in statistics/machine learning: by training them on the training set & evaluating them on the test set.

Conclusion:

To summarize, we began by exploring simplest Form of classification: binary. We then moved further into multi-class classification, when the response variable can take any no. of states.

We also saw how to fit & evaluate models with training & test set, Furthermore, we could explore additional ways to define model fitting among various algorithms.

In [1]:
```
import sklearn as sk
import pandas as pd
```

Binary Classification

In [2]:
```
import pandas as pd
import os


heart = pd.read_csv((r"C:\Users\prati\Desktop\data.csv"), sep=',', header=0)
heart.head()

y = heart.iloc[:,9]
X = heart.iloc[:,:9]
heart
```

Out[2]:

	sbp	tobacco	ldl	adiposity	famhist	typea	obesity	alcohol	age	chd
0	160	12.00	5.73	23.11	1	49	25.30	97.20	52	1
1	144	0.01	4.41	28.61	0	55	28.87	2.06	63	1
2	118	0.08	3.48	32.28	1	52	29.14	3.81	46	0
3	170	7.50	6.41	38.03	1	51	31.99	24.26	58	1
4	134	13.60	3.50	27.78	1	60	25.99	57.34	49	1
...	...	...	...	...	...	...	...	...	...	...
457	214	0.40	5.98	31.72	0	64	28.45	0.00	58	0
458	182	4.20	4.41	32.10	0	52	28.61	18.72	52	1
459	108	3.00	1.59	15.23	0	40	20.09	26.64	55	0
460	118	5.40	11.61	30.79	0	64	27.35	23.97	40	0
461	132	0.00	4.82	33.41	1	62	14.70	0.00	46	1

462 rows × 10 columns

Logistic Regression

In [3]:
```
import sklearn as sk
from sklearn.linear_model import LogisticRegression
import pandas as pd
import os

heart = pd.read_csv((r"C:\Users\prati\Desktop\data.csv"), sep=',', header=0)
heart.head()

y = heart.iloc[:,9]
X = heart.iloc[:,:9]

LR = LogisticRegression(random_state=0, solver='lbfgs', multi_class='ovr').fit(X, y)
LR.predict(X.iloc[460:,:])
round(LR.score(X,y), 4)
```

```
C:\Users\prati\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Conver
```

```
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

Out[3]: 0.7359

Support Vector Machines

In [4]:
```
import sklearn as sk
from sklearn import svm
import pandas as pd
import os


heart = pd.read_csv((r"C:\Users\prati\Desktop\data.csv"), sep=',', header=0)

y = heart.iloc[:,9]
X = heart.iloc[:,:9]

SVM = svm.LinearSVC()
SVM.fit(X, y)
SVM.predict(X.iloc[460:,:])
round(SVM.score(X,y), 4)
```

```
C:\Users\prati\anaconda3\lib\site-packages\sklearn\svm\_base.py:976: ConvergenceWarning:
Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

Out[4]: 0.3485

Random Forests

In [5]:
```
import sklearn as sk
from sklearn.ensemble import RandomForestClassifier

RF = RandomForestClassifier(n_estimators=100, max_depth=2, random_state=0)
RF.fit(X, y)
RF.predict(X.iloc[460:,:])
round(RF.score(X,y), 4)
```

Out[5]: 0.7338

Neural Networks

In [6]:
```
import sklearn as sk
from sklearn.neural_network import MLPClassifier

NN = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=
NN.fit(X, y)
NN.predict(X.iloc[460:,:])
round(NN.score(X,y), 4)
```

Out[6]: 0.6537

Multi-Class Classification

In [7]:
```
import pandas as pd
import os
vowel_train = pd.read_csv((r"C:\Users\prati\Desktop\voweltrain.csv"), sep=',', header=0
vowel_test = pd.read_csv((r"C:\Users\prati\Desktop\voweltest.csv"), sep=',', header=0)

vowel_train.head()

y_tr = vowel_train.iloc[:,0]
X_tr = vowel_train.iloc[:,1:]

y_test = vowel_test.iloc[:,0]
X_test = vowel_test.iloc[:,1:]
vowel_train
```

Out[7]:

	y	x.1	x.2	x.3	x.4	x.5	x.6	x.7	x.8	x.9	x.10
0	1	-3.639	0.418	-0.670	1.779	-0.168	1.627	-0.388	0.529	-0.874	-0.814
1	2	-3.327	0.496	-0.694	1.365	-0.265	1.933	-0.363	0.510	-0.621	-0.488
2	3	-2.120	0.894	-1.576	0.147	-0.707	1.559	-0.579	0.676	-0.809	-0.049
3	4	-2.287	1.809	-1.498	1.012	-1.053	1.060	-0.567	0.235	-0.091	-0.795
4	5	-2.598	1.938	-0.846	1.062	-1.633	0.764	0.394	-0.150	0.277	-0.396
...	...	...	...	...	...	...	...	...	...	...	...
523	7	-4.065	2.876	-0.856	-0.221	-0.533	0.232	0.855	0.633	-1.452	0.272
524	8	-4.513	4.265	-1.477	-1.090	0.215	0.829	0.342	0.693	-0.601	-0.056
525	9	-4.651	4.246	-0.823	-0.831	0.666	0.546	-0.300	0.094	-1.343	0.185
526	10	-5.034	4.993	-1.633	-0.285	0.398	0.181	-0.211	-0.508	-0.283	0.304
527	11	-4.261	1.827	-0.482	-0.194	0.731	0.354	-0.478	0.050	-0.112	0.321

528 rows × 11 columns

In [8]:
```
import pandas as pd
import sklearn as sk
from sklearn.linear_model import LogisticRegression
from sklearn import svm
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier

vowel_train = pd.read_csv((r"C:\Users\prati\Desktop\voweltrain.csv"), sep=',', header=0
vowel_test = pd.read_csv((r"C:\Users\prati\Desktop\voweltest.csv"), sep=',', header=0)

y_tr = vowel_train.iloc[:,0]
X_tr = vowel_train.iloc[:,1:]

y_test = vowel_test.iloc[:,0]
X_test = vowel_test.iloc[:,1:]

LR = LogisticRegression(random_state=0, solver='lbfgs', multi_class='multinomial').fit(
LR.predict(X_test)
round(LR.score(X_test,y_test), 4)

SVM = svm.SVC(decision_function_shape="ovo").fit(X_tr, y_tr)
```

```
SVM.predict(X_test)
round(SVM.score(X_test, y_test), 4)

RF = RandomForestClassifier(n_estimators=1000, max_depth=10, random_state=0).fit(X_tr,
RF.predict(X_test)
round(RF.score(X_test, y_test), 4)

NN = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(150, 10), random_sta
NN.predict(X_test)
round(NN.score(X_test, y_test), 4)
```

```
C:\Users\prati\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Conver
genceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
C:\Users\prati\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_perceptro
n.py:471: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
  self.n_iter_ = _check_optimize_result("lbfgs", opt_res, self.max_iter)
```

Out[8]: 0.5281

In [ ]: