

**PRAIRIE VIEW A&M UNIVERSITY
ROY G. PERRY COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
ELEG 4310 P01 – ST/MACHINE LEARNING
FALL 2022**

I. COURSE DESCRIPTION IN ABET FORMAT

II. TESTS

Summary of Topics covered by Test

Original Test Questions & Master Solution/grading scheme used

(Summary of topics can be a separate page or include on the assignment itself)

Above Average (Graded original work)

Average (Graded original work)

Below Average (Graded original work)

III. HOMEWORK ASSIGNMENTS

Summary of Topics covered by Homework

Original Homework Questions & Master Solution/grading scheme used

Above Average (Graded original work)

Average (Graded original work)

Below Average (Graded original work)

V. PROJECT

Summary of Topics Covered by Project

Project Problem Statement and Grading Scheme

Above Average (Graded original work)

Average (Graded original work)

Below Average (Graded original work)

Course Description in ABET Format

ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT ELEG 4310 P01 ST/MACHINE LEARNING

1.	Course number and name:	ELEG 4310 P01 ST/MACHINE LEARNING
2.	Credits and contact hours:	3 credit hours; * Subject Area: Engineering Topics
3.	Instructor's or course coordinator's name:	Dr. Xishuang Dong
4.	Text book: Title:	An Introduction to Machine Learning
	Author:	Miroslav Kubat
	Publisher, Year & Edition:	2017, Springer International Publishing, ISBN: 978-3319348865
	a. Other Supplemental material	Pattern Recognition and Machine Learning, by Christopher M. Bishop, Springer- Verlag New York, 2006. ISBN: 978-0387310732 Machine Learning: A Probabilistic Perspective, by Kevin Murphy, MIT Press, 2012. ISBN: 978-0262018029
5.	Specific course information	
	a. Brief description of the content of the course (catalog description)	ELEG 4310 – Machine Learning. Credit 3 semester hours. Fundamentals of machine learning models. Feature Selection, Bayesian Classifiers, Nearest-Neighbor Classifiers, Linear and Polynomial Classifiers, Decision Tree, Ensemble Learning, Artificial Neural Networks, hands-on practices and projects.
	b. Prerequisites or Co-requisites	None
	c. ** Indicate whether a required, elective, or selected elective (as per definition provided) course in the program	Speical Topic
6.	Specific goals for the course	Students will be able to:

	a. specific outcomes of instruction (e.g. The student will be able to explain the significance of current research about a particular topic.)	1. Understand the fundamental concepts and principles in machine learning; 2. Obtain the knowledge of several important machine learning models to solve real world problems of data analytics.
	b. Explicitly indicate which of the student outcomes listed in Criterion 3 or any other outcomes are addressed by the course.	None
7	Brief list of topics to be covered	Introduction to Machine Learning (2 hours) Introduction to Python (3 hours) Feature Selection and Sampling (3 hours) Bayesian Classifiers (3 hours) Nearest-Neighbor Classifiers (2 hours) Linear and Polynomial Classifiers (3 hours) Decision Tree (3 hours) Ensemble Learning (3 hours) Unsupervised Learning (3 hours) Artificial Neural Networks (3 hours) Introduction to Deep Learning and TensorFlow (3 hours) Convolutional Neural Networks and Recurrent Neural Networks (2 hours) Machine Learning on Embedded Systems (2 hours) Project Presentation (6 hours)

**ST/MACHINE LEARNING
FALL 2022**

Title of Assignment: TEST 1

TOPICS COVERED IN TEST 1

Test 1 covered the following topics:

1. Introduction to Machine Learning
2. Python Programming
3. Feature Selection and Sampling
4. Bayesian Classifiers
5. Decision Tree

ST/MACHINE LEARNING

FALL 2022

Test 1 Question and Solution

1. Describe 3 applications that are built with machine learning techniques (10 points).

Solution:

Face Recognition that is to learn a classifier to judge whether there is a face in the image. (3 points)
Recommendation System this is to profile personal interests as features and use these features to construct classifiers to classify products into two classes: interesting and uninteresting. (3 points)
Spam Detection that is using machine learning to build text classifier in order to recognize spamming emails. (4 points)

Any reasonable applications are acceptable.

2. Using a concrete example to explain the steps for building a machine learning model (10 points).

Solution:

There are six steps as below to build a machine learning model.

- (1) Business Understanding (1 point)
- (2) Data Understanding (1 point)
- (3) Data Preparation (1 point)
- (4) Modeling (2 point)
- (5) Evaluation (1 point)
- (6) Deployment (1 point)

Taking face recognition as the example, in the first step, developers should figure out the business value of face recognition. Then in the second step, the developers should gain the basic statistic information on the raw data such as ground truth distribution. In the next, the raw data should be preprocessed such as missing value processing and removing noises. In the fourth step, it is to select a machine learning methods or constructing a novel machine learning method on the pre-processed training data. Next, the machine learning should be evaluated with predefined metrics on the testing data. Finally, the model can be deployed to the customers if the evaluating results satisfy to customer requirements. (3 points)

3. Why do we need to process missing values on the raw data for building machine learning models? Please use an example to illustrate the reason, where the example must include the concrete the methods to processing the missing values (10 points).

Solution:

The reasons are as below. (4)

- (1) Data is rarely clean and often you have data quality issues.
- (2) Avoid "garbage in, garbage out"

The methods to processing missing values include Deletion, Mean substitution, and frequent substitution. (4 points)

One concrete example is shown as the following figure. (2 points)

The table illustrates various methods for handling missing data in a time series. Annotations above the table indicate the following actions:

- Insert missing records:** Points to the first missing value in the 'air_mv' column (row 5).
- Replace with 0:** Points to the 'air_mv_zero' column.
- Replace with last known value:** Points to the 'air_mv_previous' column.
- Replace with mean:** Points to the 'air_mv_mean' column.
- Interpolate based on splines:** Points to the 'air_expand' column.

	DATE	air_mv	air_mv_zero	air_mv_previous	air_mv_mean	air_expand
1	JAN49	112	112	112	112	112
2	FEB49	118	118	118	118	118
3	MAR49	132	132	132	132	132
4	APR49	129	129	129	129	129
5	MAY49	.	0	129	284.54385965	128.29783049
6	JUN49	135	135	135	135	135
7	JUL49	.	0	135	284.54385965	144.73734152
8	AUG49	148	148	148	148	148
9	SEP49	136	136	136	136	136
10	OCT49	119	119	119	119	119
11	NOV49	.	0	119	284.54385965	116.19900978
12	DEC49	118	118	118	118	118
13	JAN50	115	115	115	115	115
14	FEB50	126	126	126	126	126
15	MAR50	141	141	141	141	141

4. Write a function with Python that is to implement **Addition, Subtraction, Multiplication, and Division** between two numbers (10 points).

Solution:

```
def calculation(num1, num2): (1 points)
```

```
    #addition
```

```
    sum = num1 + num2 (2 points)
```

```
    #subtraction
```

```
    sub = num1 - num2 (2 points)
```

```
    #multiplication
```

```
    mul = num1 * num2 (2 points)
```

```
    #division
```

```
    div = num1 / num2 (2 points)
```

```
    return sum, sub, mul, div (1 points)
```

5. List 2 machine learning models you have learned and describe their limitations and advantages (15 points).

Solution:

Bayes Model (7 points)

Limitations:

(1) If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero.

(2) Strong assumption that all features are independent.

Advantages:

(1) Easy to build

(2) Faster to train, compared to the other models

Decision Tree (8 points)

Advantages of Decision Tree:

- (1) Simple to understand and to interpret
- (2) To build decision tree requires little data preparation
- (3) Decision trees are able to handle both continuous and categorical variables.
- (4) Implicitly perform feature selection

Disadvantages of Decision Tree:

- (1) They are prone to over-fitting.
- (2) Decision Tree learner can create biased trees in case of unbalanced data

6. Suppose that you will build a Gaussian Naïve Bayes model on the dataset, where the dataset contains 10 features and 2 classes. How many parameters will be learned on the datasets when completing the training of the Gaussian Naïve Bayes model? Please write your answer with a detailed procedure (10 points).

Solution:

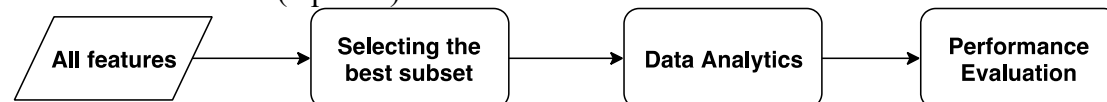
- (1) It's binary classification (2 classes). (2 points)
 - (2) There are 10 features, and each feature will have 2 parameters (one mean value and one standard deviation) for each class. (4 points)
 - (3) The total parameters 40 (4 points)
- $$40 = 10 * 2 * 2$$

7. Feature Selection (15 points).

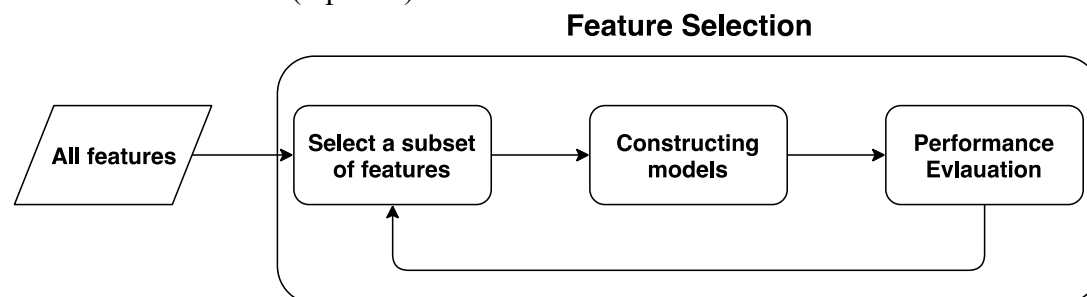
Solution:

- (1) Using figures to explain the flows of filter-based and model-based feature selection methods (8 points)

Filter-based method (4 points)



Model-based method (4 points)



- (2) Explaining Chi Squared Test based feature selection

Solution:

Mathematically, a Chi-Square test is done on two distributions to determine the level of similarity of their respective variances.

In its null hypothesis, it assumes that the given distributions are independent.

This test can be used to determine the best features for a given dataset by determining the features on which the target attribute is most dependent on.

For each feature in the dataset, the χ^2 is calculated and then ordered in descending order according to the χ^2 value.

8. Suppose you have the following training examples, described by four attributes, x_1, x_2, x_3, x_4 , and labeled by classes c_1 and c_2 .

x_1	x_2	x_3	x_4	class
0.5	0.9	1.3	2.1	c_1
1.1	0.6	1.5	2.6	c_1
1.3	2.4	2.5	1.7	c_2
6.3	1.9	3.8	6.4	c_2
2.1	3.2	3.5	5.4	c_2

Decide whether $x = [1.5, 3.7, 3.8, 3.3]$ should belong to c_1 or c_2 with *Gaussian Bayes Model* (20 points).

For class 1, we will have 8 parameters including $\mu_{1,1}, \mu_{1,2}, \mu_{1,3}, \mu_{1,4}$, and $\delta_{1,1}, \delta_{1,2}, \delta_{1,3}, \delta_{1,4}$.

Solution:

$$\mu_{1,1} = (0.5 + 1.1)/2.0 = 0.8 \text{ (1 point)}$$

$$\mu_{1,2} = (0.9 + 0.6)/2.0 = 0.75 \text{ (1 point)}$$

$$\mu_{1,3} = (1.3 + 1.5)/2.0 = 1.4 \text{ (1 point)}$$

$$\mu_{1,4} = (2.1 + 2.6)/2.0 = 2.35 \text{ (1 point)}$$

$$\delta_{1,1} = 0.42 \text{ (1 point)}$$

$$\delta_{1,2} = 0.21 \text{ (1 point)}$$

$$\delta_{1,3} = 0.14 \text{ (1 point)}$$

$$\delta_{1,4} = 0.35 \text{ (1 point)}$$

$$\mu_{2,1} = 3.2 \text{ (1 point)}$$

$$\mu_{2,2} = 2.5 \text{ (1 point)}$$

$$\mu_{2,3} = 3.2 \text{ (1 point)}$$

$$\mu_{2,4} = 4.5 \text{ (1 point)}$$

$$\delta_{2,1} = 2.6 \text{ (1 point)}$$

$$\delta_{2,2} = 0.65 \text{ (1 point)}$$

$$\delta_{2,3} = 0.68 \text{ (1 point)}$$

$$\delta_{2,4} = 2.47 \text{ (1 point)}$$

$$p(c_1|x) = p(c_1) * p(x_1|c_1) * p(x_2|c_1) * p(x_3|c_1) * p(x_4|c_1) = 0.0005 \text{ (1 point)}$$

$$p(c_2|x) = p(c_2) * p(x_1|c_2) * p(x_2|c_2) * p(x_3|c_2) * p(x_4|c_2) = 4.5e-107 \text{ (1 point)}$$

$$p(c_1|x) > p(c_2|x) \text{ so } x \text{ belongs to } c_1 \text{ (2 point)}$$

ABOVE AVERAGE

ELEG 4310-P01
ST/Machine Learning
Spring 2022 – Test

75

The copies of the lecture slides AND any textbooks are not allowed.

1. Describe 3 applications that are built with machine learning techniques (10 points).

Feature Selection Data Normalization mean
Feature Reduction Naive Bayes Filter method
Data Normalization min-max wrapper method
Data Preprocessing z-score embedded method

-1

2. Using a concrete example to explain the steps for building a machine learning model (10 points).

An example to explain the step for building a machine learning model would be the process it takes to make a iPhone. First Apple have the data split into training data (80%) and Testing the data (20%). Next the iPhones algorithm is inserted then the data is generated. After, many tests are done on data in order to have an accurate answer. Finally the new data is produced.

-4

Prairie View A&M University

3. Why do we need to process missing values on the raw data for building machine learning models? Please use an example to illustrate the reasons, where the example must include the concrete methods to process the missing values (10 points).

A. The reason we must process the missing values on the raw data is because it simply isn't accurate enough with a 0 inserted. So if the results are equal and a zero is in one of the results we need to replace the 0 in order to find the greater number. -2

b. Concrete methods to process the missing values are dummy substitution, mean substitution, frequent substitution, regression substitution and you can also convert the question to Log and replace the zero with a low decimal. Ex.

$$\log(8) + \log(6) + \log(5) + \log(0.001) \quad \leftarrow \text{Substitution}$$

Def Variables

4. Write a function with Python that is to implement Addition, Subtraction, Multiplication, and Division between two numbers (10 points).

~~ar(a) * ar(b) (156)~~; Product = a * b; Num = 156 * 50

ar(b) / ar(50); Print(Product) OR Sum = (Num) ++

sum = a + b; Quotient = a ÷ b;

Print(sum) =

difference = a - b; Print(Quotient)

Print(difference)

Prairie View A&M University

5. Describe limitations and advantages of Bayes model and Decision Tree model (15 points).

Overfitting

The limitations of Bayes Model and the Decision Tree model is overfitting and inaccurate subsets sometimes.

The advantages are generating a tree like model to create branch nodes and to organize data and exclude unnecessary information. The goal is to decrease the uncertainty (entropy). ~~Feature selection~~

-3

6. Suppose that you will build a Gaussian Naïve Bayes model on a dataset, where the dataset contains 10 features and 2 classes. How many parameters will be learned on the dataset when completing the training of the Gaussian Naïve Bayes model? Please write your answer with a detailed calculation procedure (10 points).

$$\text{training} = 80 \text{ of } 10 = 8$$

-3

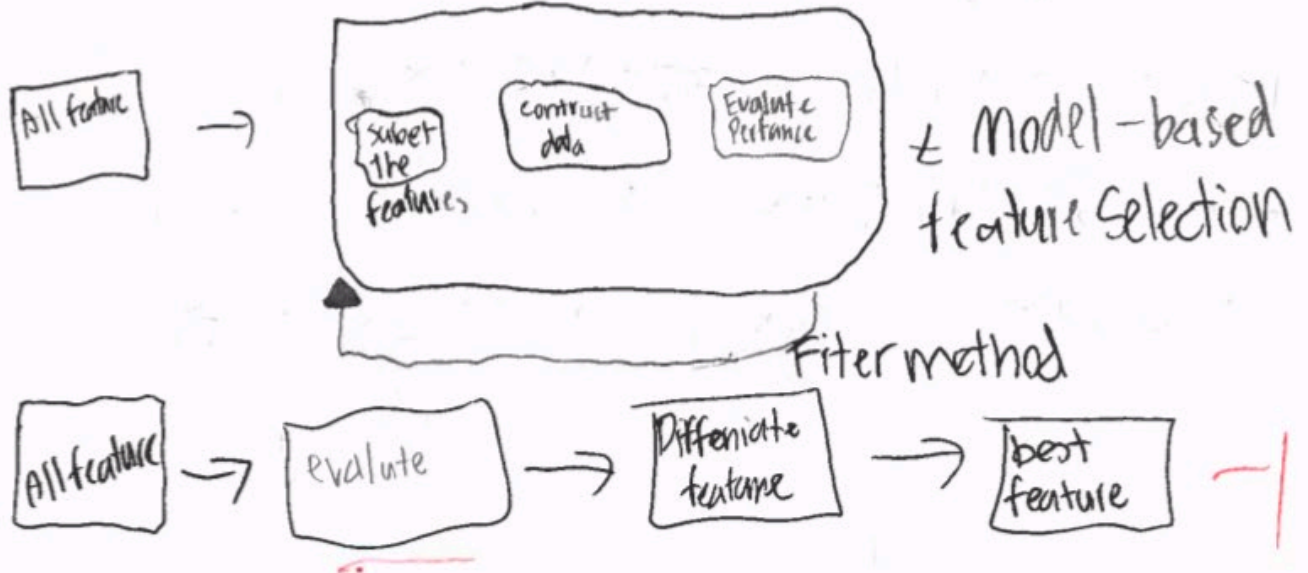
out of the 8 features being used for training there will be a total of 16 parameters being calculated. The parameter classes are standard deviation and the mean. I multiplied the total features by the total ~~forms~~ forms of calculation needed and got 16. $8 \times 2 = 16$

Feature number of classes

Prairie View A&M University

7. Feature Selection (15 points).

(1) Using figures to explain the flows of filter-based and model-based feature selection methods, respectively (8 points)



(2) Explaining how to implement Chi Squared Test based feature selection (7 points)

Chi squared Test is to find the respective variances between features.

-5

Prairie View A&M University

8. Suppose you have the following training examples containing four attributes (features), x_1 , x_2 , x_3 , and x_4 , and labeled by classes c_1 and c_2 .

x_1	x_2	x_3	x_4	class
0.5	0.9	1.3	2.1	c_1
1.1	0.6	1.5	2.6	c_1
1.3	2.4	2.5	1.7	c_2
6.3	1.9	3.8	6.4	c_2
2.1	3.2	3.5	5.4	c_2

Decide whether $x' = [1.5, 3.7, 3.8, 3.3]$ should belong to c_1 or c_2 with Gaussian Naive Bayes Model (20 points).

~~$x' = [1.5, 3.7, 3.8, 3.3]$~~

$$0.5 + 0.9 + 1.3 + 2.1 + 1.1 + 0.6 + 1.5 + 2.6$$

$$\mu_1 = 1.325$$

$$\mu_2 = 3.375$$

$$(0.5 \times 1.325)(0.9 \times 1.325)(1.3 \times 1.325)(2.1 \times 1.325) + (1.1 \times 1.325) + (0.6 \times 3.375) + (1.5 \times 1.325) + (2.6 \times 1.325)$$

7

-5

$$\sigma = 2.006$$

←

AVERAGE

Prairie View A&M University

Osesise Ayen
ELEG 4310-P01
ST/Machine Learning
Spring 2022 – Test

The copies of the lecture slides AND any textbooks are not allowed.

1. Describe 3 applications that are built with machine learning techniques (10 points).

Excel → It includes data entry and listing of data,
Apple cloud → It is built using python
PUBG G → It is a gaming application written in python
which is apart of machine learning

2. Using a concrete example to explain the steps for building a machine learning model (10 points).

- Step 1: Understand the business problem
- Step 2: Identify the data
- Step 3: Collect and prepare the data
- Step 4: Determine the model feature
- Step 5: Evaluate the model performance
- Step 6: Make sure it works
- Step 7: Iterate and adjust the model

If given a data with missing values from a business first we will need to identify the data then collect the values that are missing then determine the model feature, evaluate the model performance then make sure it works and adjust the model if it needs to be adjusted.

Prairie View A&M University

3. Why do we need to process missing values on the raw data for building machine learning models? Please use an example to illustrate the reasons, where the example must include the concrete methods to process the missing values (10 points).

We need to process missing values on the raw data so there won't be errors.

substitution method

Delete method

x_1	x_2	x_3	x_4	C_1
1.6	2.3	4.5		7.5
2.2	3.7	5.6	2.2	6.6
3.4	3.8	7.8	4.4	7.7

substitution method

delete method

-2

4. Write a function with Python that is to implement Addition, Subtraction, Multiplication, and Division between two numbers (10 points).

```
def value(num, x, y):  
    num * x = x  
    num * y = y  
    def add(num)  
    Print ("Addition: num x + num y")  
    def subtract(num)  
    Print ("Subtraction: num x + num y")  
    def Multiply(num)  
    Print ("Multiplication: num x + num y")  
    def Divide  
    Print ("Division: num x + num y")
```

-1

Prairie View A&M University

5. Describe limitations and advantages of Bayes model and Decision Tree model (15 points).

Bayes model advantage is it easy to implement, It doesn't requires much truing. It is fast and can be used to make real life predictions.

Decision Tree model It handles both continuous and discrete data.

Bayes model limitation $\rightarrow$ large data records are required to achieve a good results.

Decision Tree model $\rightarrow$ It only works with large data

-4

6. Suppose that you will build a Gaussian Naïve Bayes model on a dataset, where the dataset contains 10 features and 2 classes. How many parameters will be learned on the dataset when completing the training of the Gaussian Naïve Bayes model? Please write your answer with a detailed calculation procedure (10 points).

10 features and 2 classes

$$P(A|B) = P(A) \times P(B)$$

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

$$P(L|X) = \frac{P(X|L) \times P(L)}{P(X)} = \frac{\frac{10}{2} \times 2}{10}$$

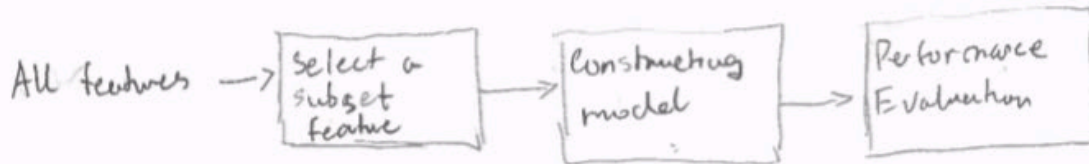
$$P = 1$$

-6

Prairie View A&M University

7. Feature Selection (15 points).

(1) Using figures to explain the flows of filter-based and model-based feature selection methods, respectively (8 points)



-4

(2) Explaining how to implement Chi Squared Test based feature selection (7 points)

-7

Prairie View A&M University

8. Suppose you have the following training examples containing four attributes (features), x_1 , x_2 , x_3 , and x_4 , and labeled by classes c_1 and c_2 .

x_1	x_2	x_3	x_4	class
0.5	0.9	1.3	2.1	c_1
1.1	0.6	1.5	2.6	c_1
1.3	2.4	2.5	1.7	c_2
6.3	1.9	3.8	6.4	c_2
2.1	3.2	3.5	5.4	c_2

Decide whether $x' = [1.5, 3.7, 3.8, 3.3]$ should belong to c_1 or c_2 with Gaussian Naive Bayes Model (20 points).

$$x' = [1.5, 3.7, 3.8, 3.3]$$
$$\text{mean} = \frac{1.5 + 3.7 + 3.8 + 3.3}{4} = 3.075$$
$$\text{variance} = 1.05$$

$$\begin{aligned} \text{mean of } c_1 &= 1.2 \\ \text{variance of } c_1 &= \\ \text{mean of } c_2 &= \end{aligned}$$

The mean and the variance are closer to c_1 than c_2

class value should belong to c_1

Answer is c_1

-10

BELOW AVERAGE

ELEG 4310-P01
ST/Machine Learning
Spring 2022 – Test

44

The copies of the lecture slides AND any textbooks are not allowed.

Oluwadamilola Oboye

1. Describe 3 applications that are built with machine learning techniques (10 points).

Application 1 I can say the Internet, and from my understanding how it uses cookies for knowing what might be searched up soon or the first 3 letters can pop up like it reading your mind but is based off of what the "world" might search up or what usually or constantly searched up by the user.

I would say "Wrapper method" as it selects and brings up the the best data.

Youtube - Filtered method, due to the World searches

Video game - uniqueness, Embedded method

-1

2. Using a concrete example to explain the steps for building a machine learning model (10 points).

Business understanding

Data understanding

Data interpretation

model

evaluation

-3

Prairie View A&M University

3. Why do we need to process missing values on the raw data for building machine learning models? Please use an example to illustrate the reasons, where the example must include the concrete methods to process the missing values (10 points).

The reason being that raw data must be pre processed because real world data is dirty

due to so much garbage in and garbage out
A method used is wrapper method which select the best features of the subsets.

Sometime data is filtered by both filter method,

Wrapper method ~~and~~ Embedded method !!
method called

-2

4. Write a function with Python that is to implement Addition, Subtraction, Multiplication, and Division between two numbers (10 points).

"sum" ←
- $x = 3$
- $y = 4$
- $value = x + y$
- Print "value of the numbers" Sum
- value of the numbers is "7"

u.sum" ←

$x = 3$
 $y = 4$
 $multiply = x * y$
- Print "multiplied value of the number" value
- value of the numbers is "12"

- $x = 4$
- $y = 3$
- $value = x - y$
- Print "value of the numbers" value
- value of the numbers is "1"

-2

$x = 4$
 $y = 2$
 $value = x / y$
- Print divide value of the number!
- value of the numbers "2"

Prairie View A&M University

5. Describe limitations and advantages of Bayes model and Decision Tree model (15 points).

Decision Tree

Advantage → Decision tree allows the user to go back in a
 doesn't require a lot of data preparation | ordered step within the branches and figure out
 Problems within the decided tree's system

Disadvantage: Data could be limited within the Decision tree
 can become very unstable in decision of system | branches branches



Advantage
 Use data with Prior Data.

$$P(A/B) = \frac{P(B/A) P(A)}{P(B)}$$

→ A being true given b is also true
 → B being true given A is also true

• B being True • A being true

Disadvantage: Doesn't give a way to figure out prior data

Limitation Some data could be blank from the Method due to its probability approach

6. Suppose that you will build a Gaussian Naïve Bayes model on a dataset, where the dataset contains 10 features and 2 classes. How many parameters will be learned on the dataset when completing the training of the Gaussian Naïve Bayes model? Please write your answer with a detailed calculation procedure (10 points).

-10

Prairie View A&M University

7. Feature Selection (15 points).

(1) Using figures to explain the flows of filter-based and model-based feature selection methods, respectively (8 points)

filter method -

Wrapper method - finding best features out of subsets

Embedded method - including both filtering methods

-4

(2) Explaining how to implement Chi Squared Test based feature selection (7 points)

$$-\sum_{i=1}^n p_i \log_2 p_i$$

to help find the best data subset best, features out of all dirty data, and clean garbage and find missing data.

-6

Prairie View A&M University

8. Suppose you have the following training examples containing four attributes (features), x_1 , x_2 , x_3 , and x_4 , and labeled by classes c_1 and c_2 .

x_1	x_2	x_3	x_4	class
0.5	0.9	1.3	2.1	c_1
1.1	0.6	1.5	2.6	c_1
1.3	2.4	2.5	1.7	c_2
6.3	1.9	3.8	6.4	c_2
2.1	3.2	3.5	5.4	c_2

Decide whether $x' = [1.5, 3.7, 3.8, 3.3]$ should belong to c_1 or c_2 with *Gaussian Naive Bayes Model* (20 points). ?

-20

**ST/MACHINE LEARNING
FALL 2022**

Title of Assignment: TEST 2

TOPICS COVERED IN TEST 2

Test 2 covered the following topics:

1. Nearest-Neighbor Classifiers
2. Linear and Polynomial Classifiers
3. Unsupervised Learning
4. Artificial Neural Networks

ST/MACHINE LEARNING

FALL 2022

Test 2 Question and Solution

1. Determine the class of $y = [2,3]$ using the 3-N classifier and the 5-NN classifier, both using the training examples from table below (25 points).

x1	2	2	1	4	3	2	6	4	8
x2	3	0	7	5	2	2	3	2	2
class	+	+	-	-	+	+	-	+	+

Solution:

$$D1 = (2-2)*(2-2) + (3-3)*(3-3) = 0 (+) \quad (2 \text{ points})$$

$$D2 = (2-2)*(2-2) + (0-3)*(0-3) = 9 (+) \quad (2 \text{ points})$$

$$D3 = (2-1)*(2-1) + (7-3)*(7-3) = 17 (-) \quad (2 \text{ points})$$

$$D4 = (2-4)*(2-4) + (5-3)*(5-3) = 8 (-) \quad (2 \text{ points})$$

$$D5 = (2-3)*(2-3) + (2-3)*(2-3) = 2 (+) \quad (2 \text{ points})$$

$$D6 = (2-2)*(2-2) + (2-3)*(2-3) = 1 (+) \quad (2 \text{ points})$$

$$D7 = (2-6)*(2-6) + (3-3)*(3-3) = 16 (-) \quad (2 \text{ points})$$

$$D8 = (2-4)*(2-4) + (2-3)*(2-3) = 5 (+) \quad (2 \text{ points})$$

$$D9 = (8-2)*(8-2) + (2-3)*(2-3) = 37 (+) \quad (2 \text{ points})$$

3-N classifier: D1 (+), D6 (+), D5 (+), applying majority voting and the class is positive (3 points)

5-N classifier: D1 (+), D6 (+), D5 (+), D8 (+), D4 (-), applying majority voting and the class is positive. (4 points)

2. Please explain details on implementing multiclass classification based on linear model (20 points).

Solution:

- Build N different binary classifiers.
 - For the i^{th} classifier, let the positive examples be all the points in class i , and let the negative examples be all the points not in class i .
 - Let f_i be the i^{th} classifier
 - On a new input x to make a prediction, pick the class i that maximizes

3. Explain vanishing gradient when applying sigmoid function as activation function to build a neural network (20 points).

Solution:

The neuron's activation saturates at either tail of 0 or 1. The curve is flat near 0 and 1. The gradient at these regions is almost zero. if the local gradient is very small, it will effectively "kill" the gradient during backpropagation. Thus, the weights in these neurons do not update. Not only that,

the weights of neurons connected to such neurons are also slowly updated. This problem is also known as vanishing gradient.

4. Given a vector $A = \langle 1.2, 1.3, 0.9, 2.5 \rangle$, show details on calculation steps of applying softmax function to map the vector A to a probability vector B (15 points).

Solution:

$$\hat{p}_i = \text{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$

$$e^{1.2} = 3.32 \text{ (1.5 point)}$$

$$e^{1.3} = 3.66 \text{ (1.5 point)}$$

$$e^{0.9} = 2.45 \text{ (1.5 point)}$$

$$e^{2.5} = 12.18 \text{ (1.5 point)}$$

$$\sum_j e^{z_j} = 3.32 + 3.66 + 2.45 + 12.18 = 21.51 \text{ (3 points)}$$

$$P_1 = 3.32/21.51 = 0.15 \text{ (1.5 point)}$$

$$P_2 = 3.66/21.51 = 0.17 \text{ (1.5 point)}$$

$$P_3 = 2.45/21.51 = 0.11 \text{ (1.5 point)}$$

$$P_4 = 12.18/21.51 = 0.57 \text{ (1.5 point)}$$

$$B = \langle 0.15, 0.17, 0.11, 0.57 \rangle$$

5. Explain the difference between supervised learning and unsupervised learning (20 points).

Solution:

- Supervised learning: (10 points).
 - Discover patterns in the data that relate data attributes with a target (class) attribute.
 - These patterns are then utilized to predict the values of the target attribute in future (unseen) data instances.
- Unsupervised learning: (10 points).
 - The data have no target attribute.
 - Explore the data to find some intrinsic structures in them.

ABOVE AVERAGE

Prairie View A&M University

ELEG 4310-P01
ST/Machine Learning
Spring 2022 – Test2

Brandon Beckwith

97

The copies of the lecture slides AND any textbooks are not allowed.

1. Determine the class of $y = [2, 3]$ using the ^{3-NN}~~3-NN~~ classifier and the 5-NN classifier, both using the training examples from table below (25 points).

x1	2	2	1	4	3	2	6	4	8
x2	3	0	7	5	2	2	3	2	2
class	+	+	-	-	+	+	-	+	+

$$y = [2, 3] \quad S_i = (x_1 - y_1)^2 + (x_2 - y_2)^2$$

sample	x1	x2	class	Distance
S_1	2	3	+	0
S_2	2	0	+	9
S_3	1	7	-	17
S_4	4	5	-	8
S_5	3	2	+	2
S_6	2	2	+	1
S_7	6	3	-	16
S_8	4	2	+	5
S_9	8	2	+	37

$$S_1 = (2-2)^2 + (3-3)^2 = 0$$

$$S_2 = (2-2)^2 + (0-3)^2 = 9$$

$$S_3 = (1-2)^2 + (7-3)^2 = 17$$

$$S_4 = (4-2)^2 + (5-3)^2 = 8$$

$$S_5 = (3-2)^2 + (2-3)^2 = 2$$

$$S_6 = (2-2)^2 + (2-3)^2 = 1$$

$$S_7 = (6-2)^2 + (3-3)^2 = 16$$

$$S_8 = (4-2)^2 + (2-3)^2 = 5$$

$$S_9 = (8-2)^2 + (2-3)^2 = 37$$

3-NN Classifier is positive

S_1 - positive - 0 using major vote

S_6 - positive - 1

S_5 - positive - 2

5-NN Classifier is positive

S_1 - positive

S_6 - positive

S_5 - positive

S_8 - positive

S_4 - negative

} positive
Classifier
using
major
vote

Prairie View A&M University

2. Please explain details on implementing multiclass classification based on linear model (20 points).

Use multiclass classification can be implemented by Naive Bayes, Theorem or One versus One method of Binary

One vs Rest

For example if C_1, C_2, C_3 are classified by binary

Logistic regression (LR)

$$LR_1 = \frac{C_1}{C_2 + C_3} = P_1$$

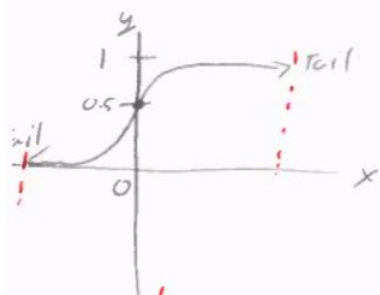
$$LR_2 = \frac{C_2}{C_1 + C_3} = P_2$$

$$LR_3 = \frac{C_3}{C_1 + C_2} = P_3$$

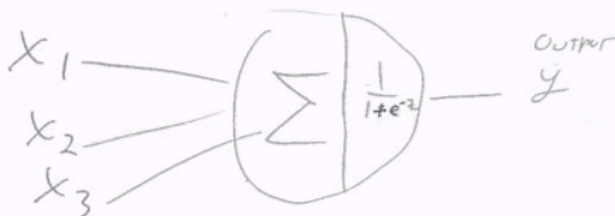
$$y = \max_i (Probability_i)$$



3. Explain vanishing gradient when applying sigmoid function as activation function to build a neural network (20 points).



Vanishing gradient is point at which curve becomes flat at either 0 or 1 of the tails. Essentially, point have the same values.



Explain the observation

"As the local gradient is very small, it will effectively 'kill' the gradient during Backpropagation"

Prairie View A&M University

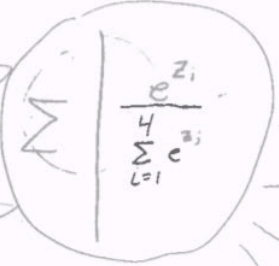
4. Given a vector $A = \langle 1.2, 1.3, 0.9, 2.5 \rangle$, show details on calculation steps of applying softmax function to map the vector A to a probability vector B (15 points).

Vector A

$$A_1 = 1.2$$

$$A_2 = 1.3$$

$$A_3 = 0.9$$

$$A_4 = 2.5$$


Vector B

$$P(A_1|B_1) = 0.1535 = 15\%$$

$$P(A_2|B_2) = 0.1698 = 17\%$$

$$P(A_3|B_3) = 0.1137 = 11\%$$

$$P(A_4|B_4) = 0.563 = 56\%$$

$$P(z_1|z_n) = \frac{e^{1.2}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.$$

$$P(z_2|z_n) = \frac{e^{1.3}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.1$$

$$P(z_3|z_n) = \frac{e^{0.9}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.1$$

$$P(z_4|z_n) = \frac{e^{2.5}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.5$$

5. Explain the difference between supervised learning and unsupervised learning (20 points).

Supervised learning has known data that has labels to find pattern in that data. It's use that pattern to establish target attribute to apply to unseen data.

Whereas unsupervised learning has no target attributes. It finds the structure of data and other data alike.

AVERAGE

Osesiye Ayeri

ELEG 4310-P01
ST/Machine Learning
Spring 2022 – Test2

95

The copies of the lecture slides AND any textbooks are not allowed.

1. Determine the class of $y = [2, 3]$ using the 3-NN classifier and the 5-NN classifier, both using the training examples from table below (25 points).

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9
x1	2	2	1	4	3	2	6	4	8
x2	3	0	7	5	2	2	3	2	2
class	+	+	-	-	+	+	-	+	+

$$d = (x_1 - x_2)^2 + (y_1 - y_2)^2 = \quad y = \begin{bmatrix} x_2 \\ y_2 \end{bmatrix}$$

$$d^1 = (2 - 2)^2 + (3 - 3)^2 = 0 \quad +$$

$$d^2 = (2 - 2)^2 + (0 - 3)^2 = 9$$

$$d^3 = (1 - 2)^2 + (7 - 3)^2 = 17$$

$$d^4 = (4 - 2)^2 + (5 - 3)^2 = 8 \quad -$$

$$d^5 = (3 - 2)^2 + (2 - 3)^2 = 2 \quad +$$

$$d^6 = (2 - 2)^2 + (2 - 3)^2 = 1 \quad +$$

$$d^7 = (6 - 2)^2 + (3 - 3)^2 = 16$$

$$d^8 = (4 - 2)^2 + (2 - 3)^2 = 5 \quad +$$

$$d^9 = (8 - 2)^2 + (2 - 3)^2 = 37$$

$$3NN = \begin{matrix} (+) & (+) & (+) \\ d_1 & d_5 & d_6 \end{matrix} = 3NN \quad \text{class is positive (+)}$$

$$5NN = \begin{matrix} (+) & (+) & (+) & (+) & (-) \\ d_1 & d_5 & d_6 & d_8 & d_4 \end{matrix} = \text{majority of the class is positive so } 5NN \text{ is positive (+)}$$

Prairie View A&M University

2. Please explain details on implementing multiclass classification based on linear model (20 points).

class = C_1^5, C_2^2, C_3^3

$LR_1 = \frac{C_1}{C_2, C_3}$

i_1
Pos
Neg

$LR_2 = \frac{C_2}{C_1, C_3}$

i_2
Pos
Neg

$LR_3 = \frac{C_3}{C_1, C_2}$

i_3
Pos
Neg

i_1
 i_2
 i_3

$LR_1 = \frac{5}{5} = 1$
 $LR_2 = \frac{2}{8} = .25$
 $LR_3 = \frac{3}{7} = .4$

Linear regression
↓
multiclass classifier

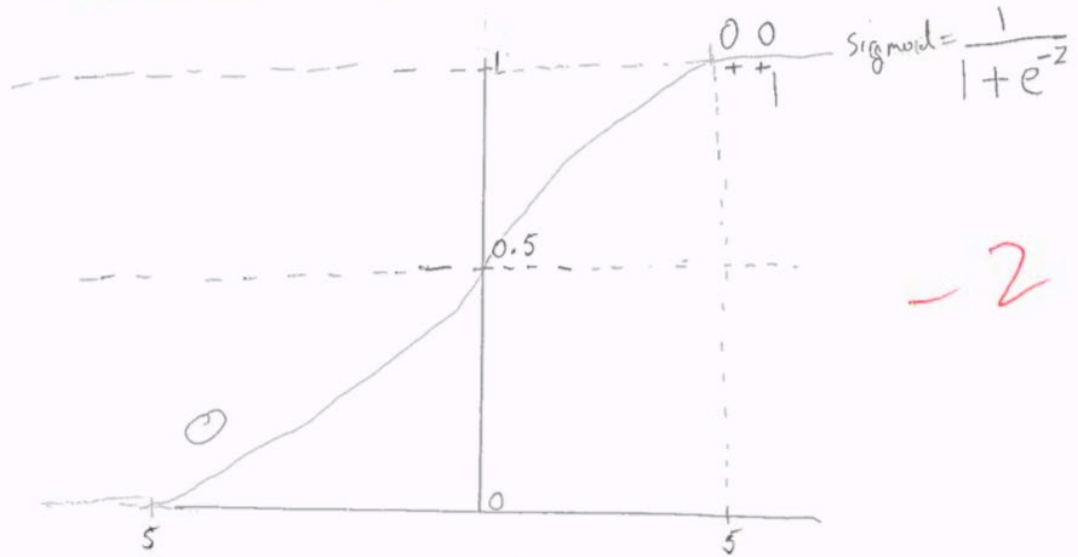
explain how to build the multiclass classifier

-3

3. Explain vanishing gradient when applying sigmoid function as activation function to build a neural network (20 points).

* Explain the in words

Vanishing gradient means 0



-2

Prairie View A&M University

4. Given a vector $A = \langle 1.2, 1.3, 0.9, 2.5 \rangle$, show details on calculation steps of applying softmax function to map the vector A to a probability vector B (15 points).

Vector input $\rightarrow$ softmax $\rightarrow$ output (probability)

$$\begin{array}{l}
 e_1: 1.2 \\
 e_2: 1.3 \\
 e_3: 0.9 \\
 e_4: 2.5
 \end{array}
 \rightarrow
 \frac{e^x}{\sum e}
 \rightarrow
 \begin{array}{l}
 0.1 \\
 0.1 \\
 0.1 \\
 0.3
 \end{array}$$

make sure the sum = to 1

$$e_1 = \frac{e^{1.2}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.0898$$

$$e_2 = \frac{e^{1.3}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.0992$$

$$e_3 = \frac{e^{0.9}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.0666$$

$$e_4 = \frac{e^{2.5}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} = 0.329$$

5. Explain the difference between supervised learning and unsupervised learning (20 points).

Supervised learning	Unsupervised learning
- Discovers pattern in the data that relates data attributes with a target audience - Predict value of the target attributes for unseen data - No Label - Predicts human-supplied data	- The data has no target attribute - Explore the data structure of the data - Unsupervised learning needs label

BELOW AVERAGE

ELEG 4310-P01
ST/Machine Learning
Spring 2022 – Test2

Eldric Bulley

90

The copies of the lecture slides AND any textbooks are not allowed.

max-x

max-min

1. Determine the class of $y = [2, 3]$ using the 3-NN classifier and the 5-NN classifier, both using the training examples from table below (25 points).

	s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	s_9
x_1	2	2	1	4	3	2	6	4	8
x_2	3	0	7	5	2	2	3	2	2
class	+	+	-	-	+	+	-	+	+

x_1	x_2	class	$y = [2, 3]$	Rank
2	3	+	$(2-2)^2 + (3-3)^2 = 0$	1
2	0	+	$(2-2)^2 + (0-3)^2 = 9$	6
1	7	-	$(1-2)^2 + (7-3)^2 = 17$	8
4	5	-	$(4-2)^2 + (5-3)^2 = 8$	5
3	2	+	$(3-2)^2 + (2-3)^2 = 2$	3
2	2	+	$(2-2)^2 + (2-3)^2 = 1$	2
6	3	-	$(6-2)^2 + (3-3)^2 = 16$	7
4	2	+	$(4-2)^2 + (2-3)^2 = 5$	4
8	2	+	$(8-2)^2 + (2-3)^2 = 37$	9

1

Prairie View A&M University

2. Please explain details on implementing multiclass classification based on linear model (20 points).

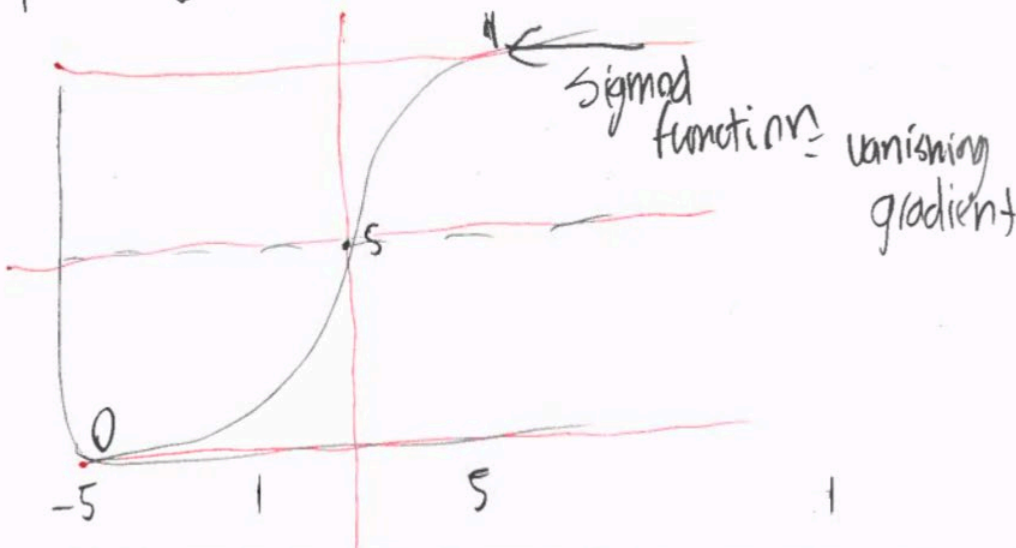
Linear Regression model

fb LR $\frac{C_1}{C_2 C_3}$ $\frac{C_2}{C_1 C_3}$ $\frac{C_3}{C_2 C_1}$

In order to create a multiclass classification with linear model you have to set up classes the rest. This means putting the class as the numerator and the other classes on the denominator.

3. Explain vanishing gradient when applying sigmoid function as activation function to build a neural network (20 points).

The vanishing gradient is a 0 slope and it is also called a sigmoid function. As the slopes goes to 0 this is where the sigmoid function is present.



Prairie View A&M University

4. Given a vector $A = \langle 1.2, 1.3, 0.9, 2.5 \rangle$, show details on calculation steps of applying softmax function to map the vector A to a probability vector B (15 points).

$$A = \langle 1.2, 1.3, 0.9, 2.5 \rangle$$
$$z_1 = \frac{e^{1.2}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} \quad z_2 = \frac{e^{1.3}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} \quad z_3 = \frac{e^{0.9}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}}$$
$$z_4 = \frac{e^{2.5}}{e^{1.2} + e^{1.3} + e^{0.9} + e^{2.5}} \rightarrow \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}$$

5. Explain the difference between supervised learning and unsupervised learning (20 points).

Supervised learning is labeled and
un Supervised learning is not labeled and
you have to find the intrinsic features
to relate the data.

-4

ST/Machine Learning
FALL 2022

HOMEWORK 1

TOPICS COVERED IN HOMEWORK 1

1. Introduction to Machine Learning
2. Python Programming
3. Feature Selection and Sampling

ST/Machine Learning

FALL 2022

HOMEWORK #1

Title: Data Preprocessing and Feature Selection.

- Processing missing values on the dataset “pima-indians-diabetes” (<https://www.kaggle.com/uciml/pima-indians-diabetes-database?select=diabetes.csv>), where the missing values are zeros for these attributes
 - Displaying the number of missing values in the first 3 columns (attributes)
 - Replacing missing values in the first 3 columns with “NaN” and outputting the results
 - Fill missing values with mean column values for the first 3 columns
- Feature selection on the dataset “pima-indians-diabetes”
 - Selecting 3 features with Chi2 method from the dataset and validate the features with logistic regression model

Note: Download the dataset "pima-indians-diabetes" from <https://www.kaggle.com/uciml/pima-indians-diabetes-database?select=diabetes.csv>

Requirements:

- Submitting the jupyter notebook and the data, where the codes in the jupyter notebook have no bugs. (5 points)
- Write and submit a report with the following parts: (5 points)
 - Task description
 - Method description
 - Result analysis

FALL 2022

HOMEWORK #1

Coding Solution:

Part 1:

```
In [1]: #import pac
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

```
In [2]: # load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
dataset = pd.read_csv(url, header = None)
```

```
In [3]: print(dataset.describe())

mean = dataset.mean()
print(mean)
```

```
In [4]: #Prints the first 5 rows of data
print(dataset.head(5))
```

	0	1	2	3	4	5	6	7	8
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [5]: #Count the number of true values in each column
#zero values as True
print((dataset[[1,2,3]] == 0).sum())
```

	1	2	3
1	5		
2	35		
3	227		

dtype: int64

```
In [6]: # Mark zero values as missing or NaN
dataset[[1,2,3]] = dataset[[1,2,3]].replace(0, np.NaN)

# Count the number of NaN values in each column
print(dataset.isnull().sum())
```

```
In [7]: # Prints the first 5 rows of data
print(dataset.head(5))
```

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.0	0	33.6	0.627	50	1
1	1	85.0	66.0	29.0	0	26.6	0.351	31	0
2	8	183.0	64.0	NaN	0	23.3	0.672	32	1
3	1	89.0	66.0	23.0	94	28.1	0.167	21	0
4	0	137.0	40.0	35.0	168	43.1	2.288	33	1

```
In [8]: dataset[[1,2,3]] = dataset[[1,2,3]].replace(np.NaN, mean[3])
# Prints the first 5 rows of data
print(dataset.head(5))
```

	0	1	2	3	4	5	6	7	8
0	6	148.0	72.0	35.000000	0	33.6	0.627	50	1
1	1	85.0	66.0	29.000000	0	26.6	0.351	31	0
2	8	183.0	64.0	20.536458	0	23.3	0.672	32	1
3	1	89.0	66.0	23.000000	94	28.1	0.167	21	0
4	0	137.0	40.0	35.000000	168	43.1	2.288	33	1

Part 2:

```
In [1]: #import pac
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

```
In [2]: # load data

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pd.read_csv(url, names=names)
```

```
In [3]: # Extract data
array = dataframe.values
X = array[:500,0:8] # 0 ~ 7 columns for features
Y = array[:500,8] # 8th column for the label
print(X)
print(Y)
```

```
In [4]: # Evaluate an Logistic Regression (LR) model with Accuracy
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
result = cross_val_score(model, X, Y, cv = kfold, scoring='accuracy')
```

```
In [5]: # Display the mean of Accuracy of 3-fold cross validation
print('Accuracy mean of 3-fold cross validation is ', result.mean())
```

Accuracy mean of 3-fold cross validation is 0.7540581487627156

```
In [6]: # Scoring features with chi2
fs_chi2 = SelectKBest(score_func = chi2, k = 3)
fs = fs_chi2.fit(X, Y)
print(fs.scores_)

[ 71.03072347  856.99913158   3.92645367  17.31170936 1688.9109822
 100.05694658   3.87182694   86.94972523]
```

```
In [7]: # Selecting features with these scores
feature_selected = fs.transform(X)

# Displaying selected features
print(feature_selected[0:5,:])
```

```
In [8]: # Evaluate an Logistic Regression (LR) model with Accuracy on Selected Features
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
fs_result = cross_val_score(model, feature_selected, Y, cv = kfold, scoring='accuracy')
```

```
In [9]: # Display the mean of Accuracy of 3-fold cross validation
print('Accuracy mean of 3-fold cross validation is ', fs_result.mean())
```

Accuracy mean of 3-fold cross validation is 0.7460620926821057

```
In [10]: # Comparing performance between original data
# and data processing with feature selection
print('Original Accuracy: ', result.mean())
print('VS')
print('FS Accuracy: ', fs_result.mean())
```

Original Accuracy: 0.7540581487627156
VS
FS Accuracy: 0.7460620926821057

ABOVE AVERAGE

Report grading: **5 points**



ELEG 4310 Special Topic Machine Learning
Homework 1: Data Preprocessing and Feature Selection
Brandon Beckwith
Instructor: Dr. Xishuang Dong
Due February 25, 2022

Task description

- Data Process
 - Use file link “pima-indians-diabetes” file to process missing values
- Count the missing values
 - In the first 3 columns the number of missing values
- Substitute missing values
 - Replace missing values with “NaN” and outputting the results
- Fill missing values
 - Use mean values in place of the missing values

- Create Feature selection
 - To generate score for testing method
- Selecting 3 features
 - Using Chi2 method from the dataset and cross validation method

Method description

Data process

First , 'Import pac' script for libraries used within code, then utilize the url for the CSV file for the Data and set a unique function call for the CSV file. Isolating the first the columns by using the '.iloc[:,0:3]'. In the code shown, dataset is used as the function call to load the data, and 'FTC' is the function that isolates the first three column.

```
[1] #import pac
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

```
[2] # load data

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
dataset = pd.read_csv(url, names=names)
FTC = dataset.iloc[:,0:3]
c1 = FTC['Pregnancies'].mean()
c2 = FTC['Glucose'].mean()
c3 = FTC['BloodPressure'].mean()
```

```
[3] print(dataset.describe())
```

	Pregnancies	Glucose	...	Age	Outcome
count	768.000000	768.000000	...	768.000000	768.000000
mean	3.845052	120.894531	...	33.240885	0.348958
std	3.369578	31.972618	...	11.760232	0.476951
min	0.000000	0.000000	...	21.000000	0.000000
25%	1.000000	99.000000	...	24.000000	0.000000
50%	3.000000	117.000000	...	29.000000	0.000000
75%	6.000000	140.250000	...	41.000000	1.000000
max	17.000000	199.000000	...	81.000000	1.000000

[8 rows x 9 columns]

The Mean values for each column were assign in order a call function as followed c1, c2 c3 for future use.

Counting the missing values was generated by '.sum()'. By writing the '(FTC==0).sum()' was able to obtain the summation of the FTC (First Three Columns) using the print() command. The FTC columns acknowledged for the replacement of unknown as 0 to NaN using the function.

```
[4] print((FTC == 0).sum())

Pregnancies    111
Glucose         5
BloodPressure   35
dtype: int64
```

```
[5] FTC[['Pregnancies', 'Glucose', 'BloodPressure']] = FTC[['Pregnancies', 'Glucose', 'BloodPressure']].replace(0, 'NaN')
print(FTC)
```

	Pregnancies	Glucose	BloodPressure
0	6	148	72
1	1	85	66
2	8	183	64
3	1	89	66
4	NaN	137	40
..	...	...	...
763	10	101	76
764	2	122	70
765	5	121	72
766	1	126	60
767	1	93	70

[768 rows x 3 columns]

Writing the function 'FTC[['Pregnancies']] = FTC[['Pregnancies']].replace('NaN',c1)
 Print (FTC[['Pregnancies']])' applied the mean value generated by the function call 'c1' which was initialized at the beginning of the code. The same process was continue for rest of the columns.

```
FTC[['BloodPressure']] = FTC[['BloodPressure']].replace('NaN',c3)
print(FTC[['BloodPressure']])
```

	BloodPressure
0	72.0
1	66.0
2	64.0
3	66.0
4	40.0
..	...
763	76.0
764	70.0
765	72.0
766	60.0
767	70.0

[768 rows x 1 columns]

```
[9] print(FTC)
```

	Pregnancies	Glucose	BloodPressure
0	6.000000	148.0	72.0
1	1.000000	85.0	66.0
2	8.000000	183.0	64.0
3	1.000000	89.0	66.0
4	3.845052	137.0	40.0
..	...	...	...
763	10.000000	101.0	76.0
764	2.000000	122.0	70.0
765	5.000000	121.0	72.0
766	1.000000	126.0	60.0
767	1.000000	93.0	70.0

[768 rows x 3 columns]

The following step are the process to preparation data for testing using the Chi Squared and Logistic Regression Model. Removing unknown values within the dataset will give a better overall accuracy rating. This is done using the '.dropna()' which removes the rows that have 'NaN values'. Changes the 'NaN' back from the previous 'mean' value step was necessary process for this command to work properly.

```

✓ [10] Popeyes = dataset.iloc[:,0:3].replace(0,'NaN')
0=

✓ [11] ds = Popeyes
0=
    ds = ds.apply(pd.to_numeric, errors='coerce')
    ds = ds.dropna()
    ds = ds.reset_index(drop=True)

    # Summarize the number of rows and columns in the dataset
    print(ds)

```

	Pregnancies	Glucose	BloodPressure
0	6.0	148.0	72.0
1	1.0	85.0	66.0
2	8.0	183.0	64.0
3	1.0	89.0	66.0
4	5.0	116.0	74.0
..	...	...	...
622	10.0	101.0	76.0
623	2.0	122.0	70.0
624	5.0	121.0	72.0
625	1.0	126.0	60.0
626	1.0	93.0	70.0

```
[627 rows x 3 columns]
```

Intialize the dataset to an array, we can exact data values to a character sX (features) and Y(label) for the purpose of LR model using the k-fold validation method. The k-fold is run through a thousand iteration and folded into three. Th k-fold cross validation is followed by the Chi Square test using the k(features selection) of three. This process is pushed through a feature selector by using the 'feature_selected = fs.transform(X)' to generated a score. This LR model Accuracy on Selected Features processed for a second time and evaluated with the original accuracy of the 1st test.

```
# Extract data
array = dataset.values
X = array[:627,0:8] # 0 ~ 7 columns for features
Y = array[:627,8] # 8st column for the label
print(X)
print(Y)
```

```
[[6.00e+00 1.48e+02 7.20e+01 ... 3.36e+01 6.27e-01 5.00e+01]
 [1.00e+00 8.50e+01 6.60e+01 ... 2.66e+01 3.51e-01 3.10e+01]
 [8.00e+00 1.83e+02 6.40e+01 ... 2.33e+01 6.72e-01 3.20e+01]
 ...
 [2.00e+00 1.08e+02 6.40e+01 ... 3.08e+01 1.58e-01 2.10e+01]
 [4.00e+00 9.00e+01 8.80e+01 ... 3.77e+01 3.62e-01 2.90e+01]
 [0.00e+00 1.25e+02 6.80e+01 ... 2.47e+01 2.06e-01 2.10e+01]]
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 1. 1. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1. 0. 0.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 0.
 1. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 1. 0. 1.
 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0.
 1. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 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. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0.
 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0.
 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0.
 0. 0. 0.]
```

```
[14] # Evaluate an Logistic Regression (LR) model with Accuracy
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
result = cross_val_score(model, X, Y, cv = kfold, scoring='accuracy')
```

```
[18] # Evaluate an Logistic Regression (LR) model with Accuracy on Selected Features
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
fs_result = cross_val_score(model, feature_selected, Y, cv = kfold, scoring='accuracy')
```

```
[19] # Display the mean of Accuracy of 3-fold cross validation
print('Accuracy mean of 3-fold cross validation is ', fs_result.mean())
```

Accuracy mean of 3-fold cross validation is 0.7304625199362041

```
[20] # Comparing performance between original data
# and data processing with feature selection
print('Original Accuracy: ', result.mean())
print('VS')
print('FS Accuracy: ', fs_result.mean())
```

Original Accuracy: 0.7799043062200957
VS
FS Accuracy: 0.7304625199362041

Result Analysis

As the result from the Ch Squared, the accuracy diminish as the k-fold cross validation was processed the second time from .770004 to 0.73046:. This could mean the data processing procedure did not clean the dataset enough to provide an accurate test set for the model.

Coding grading: 5 points

```
#import pac
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

In [2]:

```
# load data
```

```
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indian
s-diabetes.data.csv"
names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin
', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
dataset = pd.read_csv(url, names=names)
FTC = dataset.iloc[:,0:3]
c1 = FTC['Pregnancies'].mean()
c2 = FTC['Glucose'].mean()
c3 = FTC['BloodPressure'].mean()
```

In [3]:

```
print(dataset.describe())
```

	Pregnancies	Glucose	...	Age	Outcome
count	768.000000	768.000000	...	768.000000	768.000000
mean	3.845052	120.894531	...	33.240885	0.348958
std	3.369578	31.972618	...	11.760232	0.476951
min	0.000000	0.000000	...	21.000000	0.000000
25%	1.000000	99.000000	...	24.000000	0.000000
50%	3.000000	117.000000	...	29.000000	0.000000
75%	6.000000	140.250000	...	41.000000	1.000000
max	17.000000	199.000000	...	81.000000	1.000000

```
[8 rows x 9 columns]
```

In [4]:

```
print((FTC == 0).sum())
Pregnancies      111
Glucose           5
BloodPressure     35
dtype: int64
```

In [5]:

```
FTC[['Pregnancies', 'Glucose', 'BloodPressure']] = FTC[['Pregnancies', 'Glucose', 'BloodPressure']].replace(0, 'NaN')
print(FTC)
```

	Pregnancies	Glucose	BloodPressure
0	6	148	72
1	1	85	66
2	8	183	64
3	1	89	66
4	NaN	137	40
..	...	...	...
763	10	101	76
764	2	122	70
765	5	121	72
766	1	126	60
767	1	93	70

[768 rows x 3 columns]

In [6]:

```
FTC[['Pregnancies']] = FTC[['Pregnancies']].replace('NaN',c1)
print(FTC[['Pregnancies']])
```

	Pregnancies
0	6.000000
1	1.000000
2	8.000000
3	1.000000
4	3.845052
..	...
763	10.000000
764	2.000000
765	5.000000
766	1.000000
767	1.000000

[768 rows x 1 columns]

In [7]:

```
FTC[['Glucose']] = FTC[['Glucose']].replace('NaN',c2)
print(FTC[['Glucose']])
```

	Glucose
0	148.0
1	85.0
2	183.0
3	89.0
4	137.0
..	...
763	101.0

```
764    122.0
765    121.0
766    126.0
767     93.0
```

```
[768 rows x 1 columns]
```

In [8]:

```
FTC[['BloodPressure']] = FTC[['BloodPressure']].replace('NaN',c3)
print(FTC[['BloodPressure']])
```

```
      BloodPressure
0              72.0
1              66.0
2              64.0
3              66.0
4              40.0
..              ...
763            76.0
764            70.0
765            72.0
766            60.0
767            70.0
```

```
[768 rows x 1 columns]
```

In [9]:

```
print(FTC)
```

```
      Pregnancies  Glucose  BloodPressure
0         6.000000   148.0             72.0
1         1.000000    85.0             66.0
2         8.000000   183.0             64.0
3         1.000000    89.0             66.0
4         3.845052   137.0             40.0
..          ...      ...             ...
763       10.000000   101.0             76.0
764         2.000000   122.0             70.0
765         5.000000   121.0             72.0
766         1.000000   126.0             60.0
767         1.000000    93.0             70.0
```

```
[768 rows x 3 columns]
```

In [10]:

```
Popeyes = dataset.iloc[:,0:3].replace(0, 'NaN')
```

In [11]:

```
ds = Popeyes
ds = ds.apply(pd.to_numeric, errors='coerce')
ds = ds.dropna()
ds = ds.reset_index(drop=True)
```

```
# Summarize the number of rows and columns in the dataset
```

```
print(ds)
```

	Pregnancies	Glucose	BloodPressure
0	6.0	148.0	72.0
1	1.0	85.0	66.0
2	8.0	183.0	64.0
3	1.0	89.0	66.0
4	5.0	116.0	74.0
..	...	...	...
622	10.0	101.0	76.0
623	2.0	122.0	70.0
624	5.0	121.0	72.0
625	1.0	126.0	60.0
626	1.0	93.0	70.0

```
[627 rows x 3 columns]
```

In [13]:

```
# Extract data
```

```
array = dataset.values
```

```
X = array[:627,0:8] # 0 ~ 7 columns for features
```

```
Y = array[:627,8] # 8st column for the label
```

```
print(X)
```

```
print(Y)
```

```
[[6.00e+00 1.48e+02 7.20e+01 ... 3.36e+01 6.27e-01 5.00e+01]
 [1.00e+00 8.50e+01 6.60e+01 ... 2.66e+01 3.51e-01 3.10e+01]
 [8.00e+00 1.83e+02 6.40e+01 ... 2.33e+01 6.72e-01 3.20e+01]
 ...
 [2.00e+00 1.08e+02 6.40e+01 ... 3.08e+01 1.58e-01 2.10e+01]
 [4.00e+00 9.00e+01 8.80e+01 ... 3.77e+01 3.62e-01 2.90e+01]
 [0.00e+00 1.25e+02 6.80e+01 ... 2.47e+01 2.06e-01 2.10e+01]]
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0.
 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 0. 1.]
```

```

1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.
0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0.
1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.
1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0.
0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0. 0. 1. 0.
0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0.
0. 0. 0.]

```

In [14]:

```

# Evaluate an Logistic Regression (LR) model with Accuracy
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
result = cross_val_score(model, X, Y, cv = kfold, scoring='accuracy')

```

In [15]:

```

fs_chi2 = SelectKBest(score_func = chi2, k = 3)
fs = fs_chi2.fit(X, Y)
print(fs.scores_)
[ 86.57381139 1148.83693175    7.83722286  46.85293726 2497.07113534
 122.73207892    5.20316257 131.76934424]

```

In [17]:

```

# Selecting features with these scores
feature_selected = fs.transform(X)

# Displaying selected features
print(feature_selected[0:768,0:3])
[[148.    0.   50.]
 [ 85.    0.   31.]
 [183.    0.   32.]
 ...
 [108.    0.   21.]
 [ 90.   54.   29.]
 [125.    0.   21.]]

```

In [18]:

```

# Evaluate an Logistic Regression (LR) model with Accuracy on Selected Features
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
fs_result = cross_val_score(model, feature_selected, Y, cv = kfold, scoring='accuracy')

```

In [19]:

```
# Display the mean of Accuracy of 3-fold cross validation
print('Accuracy mean of 3-fold cross validation is ', fs_result.mean())
Accuracy mean of 3-fold cross validation is  0.7304625199362041
```

In [20]:

```
# Comparing performance between original data
# and data processing with feature selection
print('Original Accuracy: ', result.mean())
print('VS')
print('FS Accuracy: ', fs_result.mean())
Original Accuracy:  0.7799043062200957
VS
FS Accuracy:  0.7304625199362041
```

AVERAGE

Report grading: 4 points

Comments: Result analysis is not associated with performance comparison and analysis.

Kevin McDonald
Professor Domg PHD
ST/ML
25, Feb 2022

Data Preprocessing and Feature Selection.

Task Description: To preprocess the “pima-indians-diabetes” dataset and display the number of missing values in the first three columns, replace the missing values in the first 3 columns with NaN and outputting the results, fill the missing values with the mean for the first 3 columns. Then use the Chi squared method to select three features from the dataset and validate them with the logistics regression model

Method Description

The process of completing the task started out difficult with the first impediment being able to read in only the first three columns of the dataset ('Preg', 'Gluc', 'Blood'). This was circumvented by creating a new variable to place hold the old variable until they could be inserted back into the data after being preprocessed

Display-Replace-Fill. 1.) The first task of displaying the count of missing values can be solved using a .sum() function to count the number of 0's in each column. The second task of replacing the missing values or 0's was addressed using a .replace function 3.) The third task combined both the use of the .mean and the . replace functions.

Feature Selection: This was perhaps the most difficult part of the whole assignment, i spent 85% of my assigned time just trying to figure out chi squared results were right because my target and test acc. Never lined up with the lab.

Result Analysis; More knowledge and studying is needed for me to truly understand how to feature select and reduce efficiently in python, other than that the assignment was familiar to my skill set.

Coding grading: 5 points

```
#import pac
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
```

In [3]:

```
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indian
s-diabetes.data.csv"
```

```
names = ['Preg', 'Gluc', 'Blood', 'Skin', 'Sugar', 'Mass', 'DP', 'Age', 'Outty',
']
dataset = pd.read_csv(url, names=names)
```

In [4]:

```
#variables
X = dataset.iloc[:, 0:3]
c1 = X['Preg'].mean()
c2=X['Gluc'].mean()
c3=X['Blood'].mean()
```

In [5]:

```
#Count the number of 0's in each column

print((dataset[['Preg', 'Gluc', 'Blood']] == 0).sum())
Preg      111
Gluc        5
Blood      35
dtype: int64
```

In [6]:

```
#replace missing values with NaN

X[['Preg', 'Gluc', 'Blood']]=X[['Preg', 'Gluc', 'Blood']].replace(0, 'NaN')
print(X)
      Preg  Gluc  Blood
0         6  148    72
1         1   85    66
2         8  183    64
3         1   89    66
4        NaN  137    40
..      ...   ...   ...
763      10  101    76
764         2  122    70
765         5  121    72
766         1  126    60
767         1   93    70
```

[768 rows x 3 columns]

In [7]:

```
#Fill missing values with mean
X[['Blood',]]=X[['Blood']].replace('NaN', c3)
X[['Gluc',]]=X[['Gluc']].replace('NaN', c2)
X[['Preg',]]=X[['Preg']].replace('NaN', c1)
print(X)
      Preg    Gluc  Blood
0    6.000000  148.0   72.0
1    1.000000   85.0   66.0
2    8.000000  183.0   64.0
```

```
3      1.000000    89.0    66.0
4      3.845052   137.0    40.0
..      ...      ...      ...
763   10.000000   101.0    76.0
764    2.000000   122.0    70.0
765    5.000000   121.0    72.0
766    1.000000   126.0    60.0
767    1.000000    93.0    70.0
```

[768 rows x 3 columns]

In [74]:

```
# Extract data
array = dataset.values
X = array[:615,0:8]
Y = array[:615,8]
print(X)
print(Y)
[[ 6.    148.    72.    ...  33.6    0.627  50.    ]
 [ 1.     85.    66.    ...  26.6    0.351  31.    ]
 [ 8.    183.    64.    ...  23.3    0.672  32.    ]
 ...
 [ 7.    168.    88.    ...  38.2    0.787  40.    ]
 [ 6.    105.    80.    ...  32.5    0.878  26.    ]
 [11.    138.    74.    ...  36.1    0.557  50.    ]]
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0.
 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1.
 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0.
 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0.
 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
```

```
0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.
0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0. 0. 1. 0.
0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1.]
```

In [94]:

```
# Evaluate an Logistic Regression accuracy using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3)
result = cross_val_score(model, X, Y, cv = kfold, scoring='accuracy')
```

In [95]:

```
print('Accuracy mean of 3-fold cross validation is ', result.mean())
Accuracy mean of 3-fold cross validation is  0.7739837398373983
```

In [88]:

```
# Scoring features with chi2
fs_chi2 = SelectKBest(score_func = chi2, k = 3)#selecting 3 best feats
fs = fs_chi2.fit(X, Y)
print(fs.scores_)
[ 83.69144803 1147.48911049  12.31847688  45.11572705 2452.73669015
 124.70004459   5.17928166 124.91281426]
```

In [96]:

```
# Selecting features with these scores
feature_selected = fs.transform(X)
```

```
# Displaying selected features
```

```
print(feature_selected[0:5,:])
[[148.   0.  50.]
 [ 85.   0.  31.]
 [183.   0.  32.]
 [ 89.  94.  21.]
 [137. 168.  33.]]
```

In [90]:

```
# Evaluate an Logistic Regression (LR) model with Accuracy on Selected Features
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
fs_result = cross_val_score(model, feature_selected, Y, cv = kfold, scoring='accuracy')
```

In [97]:

```
print('Accuracy mean of 3-fold cross validation is ', fs_result.mean())
Accuracy mean of 3-fold cross validation is  0.7300813008130081
```

In [98]:

```
# Comparing performance between original data and data processing with feature selection
print('Original Accuracy: ', result.mean())
print('VS')
```

```
print('FS Accuracy: ', fs_result.mean())  
Original Accuracy: 0.7739837398373983  
VS  
FS Accuracy: 0.7300813008130081
```

BELOW AVERAGE

Report Grading: 4 points

Comments: task description is not well-organized.

Prairie View A&M University

ROY G. PERRY COLLEGE OF ENGINEERING



Submitted by : Osesiye Ayeni

ELEG 4310 – P01

ST/Machine Learning

Spring 2022 – Homework 1

Due: 02/25/2022

Data Preprocessing and Feature Selection

Task description

This report aims to describe homework 1 in which I performed data to process and feature selection. The following tasks were performed:

1. Processing missing values on the dataset “Pima-Indians-diabetes”
(<https://www.kaggle.com/uciml/pima-indians-diabetes-database?select=diabetes.csv>),
where the missing values are zeros for these attributes
2. Displaying the number of missing values in the first 3 columns (attributes)
3. Replacing missing values in the first 3 columns with “NaN” and outputting the results
4. Fill missing values with mean column values for the first 3 columns
5. Feature selection on the dataset “Pima-Indians-diabetes”
6. Selecting 3 features with the Chi-squared method from the dataset and validating the features with the logistic regression model

Method description

There were two different tasks: data processing and feature extraction.

Data Processing

The real-world data is unclear and must be pre-processed so that it can be used for building machine learning models. The data was obtained from [Pima Indians Diabetes Database](#).

The data processing model processes the dataset by removing or updating missing values. The Pandas and NumPy libraries were used for data cleaning purposes. The pandas library is used for reading CSV files while the NumPy library is used for finding and replacing missing

values. The missing values are dropped, reducing the dataset size from 768 rows to 392 rows. The model is using the Logistic Regression (LR) model. This model finds the relationship between the dependent variable and independent variable by estimating probabilities using a logistic regression equation. The kFold model is used for 3-fold cross-validation. The kFold cross-validation is used to split the dataset into testing and training sets. Finally, the Cross_Val_Score model is used for performing evaluation and cross-validation configuration for each fold.

Feature Selection

The pre-processed data is used in this step. The feature selection and feature reduction methods are used. Feature selection is used to remove redundant or irrelevant features without loss of information or data. Feature Reduction is the transformation of data from a high-dimensional space into a low-dimensional space where low-dimensional representation retains meaningful properties of the original data.

The model is using the Logistic Regression (LR) model. This model finds the relationship between the dependent variable and independent variable by estimating probabilities using a logistic regression equation. The kFold model is used for 3-fold cross-validation. The kFold cross-validation is used to split the dataset into testing and training sets. Finally, the Cross_Val_Score model is used for performing evaluation and cross-validation configuration for each fold. The Chi-Squared model is used to measure how the model compares to the actual model. The SelectKBest method selects the features according to the k-highest score. The same process is repeated (creating Linear Regression Model, splitting, and cross-validating) for the new dataset that is created using SelectKBest. Finally, Principal Component Analysis (PCA) is used for feature reduction as it uses linear algebra to transform

the dataset into a compressed form. Again, the same steps were repeated for model building and cross-validation for the new compressed dataset.

Result analysis

The dataset included several missing values that were pre-processed using the NumPy library. The missing values were dropped, and a machine learning model was built. The original dataset has an accuracy mean of 0.7540581487627156 (first 500 rows). The same dataset was used for feature selection. First, the SelectKBest model was used for feature extraction using the chi-squared method. The same machine learning method was built and cross-validated. The accuracy mean was measured to be 0.7620902772767718. Finally, the feature reduction method has been used that compresses the data without losing information. The same machine learning method was applied on the new dataset and the accuracy mean was found to be 0.7620902772767718 which is the same as found in the case of feature extraction. It means that feature extraction and feature reduction have similar results where the feature reduction method compresses data significantly.

Coding grading: 0 point

Comments: Python scripts doesn't match to the report.

```
In [1]: #import pac
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score

In [2]: # load data

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pd.read_csv(url, names=names)

In [3]: # Extract data
array = dataframe.values
X = array[:500,0:8] # 0 ~ 7 columns for features
Y = array[:500,8] # 8th column for the label
print(X)
print(Y)

[[6.00e+00 1.48e+02 7.20e+01 ... 3.36e+01 6.27e-01 5.00e+01]
 [1.00e+00 8.50e+01 6.60e+01 ... 2.66e+01 3.51e-01 3.10e+01]
 [8.00e+00 1.83e+02 6.40e+01 ... 2.33e+01 6.72e-01 3.20e+01]
 ...
 [2.00e+00 8.10e+01 7.20e+01 ... 3.01e+01 5.47e-01 2.50e+01]
 [7.00e+00 1.95e+02 7.00e+01 ... 2.51e+01 1.63e-01 5.50e+01]
 [6.00e+00 1.54e+02 7.40e+01 ... 2.93e+01 8.39e-01 3.90e+01]]
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1.
 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0.
 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.]
```

```
In [4]: # Evaluate an Logistic Regression (LR) model with Accuracy
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
result = cross_val_score(model, X, Y, cv = kfold, scoring='accuracy')
```

```
In [5]: # Display the mean of Accuracy of 3-fold cross validation
print('Accuracy mean of 3-fold cross validation is ', result.mean())

Accuracy mean of 3-fold cross validation is  0.7540581487627156
```

```
In [6]: # Scoring features with chi2
fs_chi2 = SelectKBest(score_func = chi2, k = 3)
fs = fs_chi2.fit(X, Y)
print(fs.scores_)

[ 71.03072347  856.99913158   3.92645367  17.31170936 1688.9109822
 100.05694658   3.87182694   86.94972523]
```

```
In [7]: # Selecting features with these scores
feature_selected = fs.transform(X)

# Displaying selected features
print(feature_selected[0:5,:])

[[148.    0.    33.6]
 [ 85.    0.    26.6]
 [183.    0.    23.3]
 [ 89.   94.    28.1]
 [137. 168.    43.1]]
```

```
In [8]: # Evaluate an Logistic Regression (LR) model with Accuracy on Selected Features
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 3) # 3-fold cross validation
fs_result = cross_val_score(model, feature_selected, Y, cv = kfold, scoring='accuracy')
```

```
In [9]: # Display the mean of Accuracy of 3-fold cross validation
print('Accuracy mean of 3-fold cross validation is ', fs_result.mean())

Accuracy mean of 3-fold cross validation is  0.7460620926821057
```

```
In [10]: # Comparing performance between original data
# and data processing with feature selection
print('Original Accuracy: ', result.mean())
print('VS')
print('FS Accuracy: ', fs_result.mean())

Original Accuracy:  0.7540581487627156
VS
FS Accuracy:  0.7460620926821057
```

**ST/Machine Learning
FALL 2022**

HOMEWORK 2

TOPICS COVERED IN HOMEWORK 2

1. K nearest neighbor (KNN) classifier
2. Weighted KNN

ST/Machine Learning

FALL 2022

HOMEWORK #2

1. Determine the class of $\mathbf{y} = [1, 22]$ using the 1-NN classifier and the 3-NN classifier, both using the training examples from Table 3.7. Explain why the two classifiers differ in their classification behavior. (8 points)
2. Use the examples from Table 3.7 to classify object $\mathbf{y} = [3, 3]$ with the 5-NN classifier. Note that two nearest neighbors are positive and three nearest neighbors are negative. Will weighted 5-NN classifier change anything? To see what is going on, plot the locations of the examples in a graph. (2 points)

Table 3.7 A simple set of training examples for the exercises

x_1	1	1	1	2	3	3	3	4	5
x_2	1	2	4	3	0	2	5	4	3
class	+	−	−	+	+	+	−	−	−

ST/Machine Learning FALL 2022

HOMEWORK #2

Solution:

Part 1 (8 points)

$$d(y, s1) = (1-1)*(1-1) + (1-22)*(1-22) = 441$$

$$d(y, s2) = (1-1)*(1-1) + (2-22)*(2-22) = 400$$

$$d(y, s3) = (1-1)*(1-1) + (4-22)*(4-22) = 324$$

$$d(y, s4) = (2-1)*(2-1) + (3-22)*(3-22) = 362$$

$$d(y, s5) = (3-1)*(3-1) + (0-22)*(0-22) = 488$$

$$d(y, s6) = (3-1)*(3-1) + (2-22)*(2-22) = 404$$

$$d(y, s7) = (3-1)*(3-1) + (5-22)*(5-22) = 293$$

$$d(y, s8) = (4-1)*(4-1) + (4-22)*(4-22) = 333$$

$$d(y, s9) = (5-1)*(5-1) + (3-22)*(3-22) = 377$$

For 1-NN, the NN will be s7, so the class will be negative.

For 3-NN, the NN will be s7 (-), s3 (+), and s8 (-). Based on majority rule, the class will be negative.

Part 2 (2 points)

$$Y = (3, 3)$$

$$d(y, s1) = (1-3)*(1-3) + (1-3)*(1-3) = 8$$

$$d(y, s2) = (1-3)*(1-3) + (2-3)*(2-3) = 5$$

$$d(y, s3) = (1-3)*(1-3) + (4-3)*(4-3) = 5$$

$$d(y, s4) = (2-3)*(2-3) + (3-3)*(3-3) = 1$$

$$d(y, s5) = (3-3)*(3-3) + (0-3)*(0-3) = 9$$

$$d(y, s6) = (3-3)*(3-3) + (2-3)*(2-3) = 1$$

$$d(y, s7) = (3-3)*(3-3) + (5-3)*(5-3) = 4$$

$$d(y, s8) = (4-3)*(4-3) + (4-3)*(4-3) = 2$$

$$d(y, s9) = (5-3)*(5-3) + (3-3)*(3-3) = 4$$

For 5-NN, the NN will be s4 (+), s6 (+), s8 (-), s7 (-), and s9 (-). The class will be negative by majority rule.

For weighted KNN,

$$d_{\min} = 1 = d(y, s4)$$

$$d_{\max} = 4 = d(y, s9)$$

$$w_i = \frac{d_{\max} - d_i}{d_{\max} - d_{\min}}$$

$$w_1 = \frac{4-1}{4-1} = 1, \text{ for } s4 (+)$$

$$w_2 = \frac{4-1}{4-1} = 1, \text{ for } s6 (+)$$

$$w_3 = \frac{4-2}{4-1} = 0.5, \text{ for } s8 (-)$$

$$w_4 = \frac{4-4}{4-1} = 0, \text{ for } s7 (-)$$

$$w_5 = \frac{4-4}{4-1} = 0, \text{ for s9 (-)}$$

$$\sum^+ = 1 + 1 = 2 > \sum^- = 0.5 + 0 + 0 = 0.5$$

The class will be positive.

ABOVE AVERAGE

Grading: 10 points

Joselye Ayeni

① $y = [1, 22]$
 $x_1 \quad x_2$

y	x_1	x_2	class	$D(x,y)$
d_1	1	1	+	21
d_2	1	2	-	20
d_3	1	4	-	18
d_4	2	3	+	19.026
d_5	3	0	+	22.09
d_6	3	2	+	20.099
d_7	3	5	-	17.117
d_8	4	4	-	18.248
d_9	5	3	-	19.41

$$d(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

$$d_1 = \sqrt{(1-1)^2 + (1-22)^2} = 21$$

$$d_2 = \sqrt{(1-1)^2 + (2-22)^2} = 20$$

$$d_3 = \sqrt{(1-1)^2 + (4-22)^2} = 18$$

$$d_4 = \sqrt{(2-1)^2 + (3-22)^2} = 19.026$$

$$d_5 = \sqrt{(3-1)^2 + (0-22)^2} = 22.09$$

$$d_6 = \sqrt{(3-1)^2 + (2-22)^2} = 20.099$$

$$d_7 = \sqrt{(3-1)^2 + (5-22)^2} = 17.117$$

$$d_8 = \sqrt{(4-1)^2 + (4-22)^2} = 18.248$$

$$d_9 = \sqrt{(5-1)^2 + (3-22)^2} = 19.41$$

For 1-NN = $d_7 = 17.117$ min distance is neg (-)
 neg

3-NN = $d_7 = 17.117$
 $d_8 = 18$
 $d_9 = 18.248$

The minimum distance at 3 and class is negative so it is assigned to neg (-) class ($y = [1, 22]$)

3-NN

top 3 minimum distance at $(1, 4), (3, 5), (4, 4)$, the point is assigned to neg (-) class

$$\textcircled{2} \quad y = \begin{bmatrix} x_1 & x_2 \\ 3 & 3 \end{bmatrix}$$

y	x_1	x_2	class	$D(x, y)$
d_1	1	1	+	2.828
d_2	1	2	-	2.236
d_3	1	4	-	2.236
d_4	2	3	+	1
d_5	3	0	+	3
d_6	3	2	+	1
d_7	3	5	-	2
d_8	4	4	-	1.414
d_9	5	3	-	2

$$w_1 = \frac{d_k - d_1}{d_k - d_1} = \frac{9 - 5}{9 - 1} = \frac{3}{8}$$

$$\begin{aligned} 5\text{-NN } d_4 &= (2, 3) \quad + \\ d_6 &= (3, 2) \quad + \\ d_8 &= (4, 4) \quad - \\ d_7 &= (3, 5) \quad - \\ d_9 &= (5, 3) \quad - \end{aligned}$$

$$\begin{aligned} d_4 &= 1 \\ d_6 &= 1 \\ d_8 &= 1.414 \\ d_7 &= 2 \\ d_9 &= 2 \end{aligned}$$

$$w_i = \frac{2 - d_i}{2 - 1}$$

$$w_4 = 2 - 1 = 1$$

$$w_6 = 2 - 1 = 1$$

$$w_8 = 2 - 1.414 = 0.59$$

$$w_7 = 2 - 2 = 0$$

$$w_9 = 2 - 2 = 0$$

If the two nearest neighbors are positive and the remaining three are negative, then $\sum^+ y$ is classified positive

$$\sum^+ = 1 + 1 > \sum^- = 0.59 + 0 + 0$$

$$\boxed{E^+ = 2 > E^- = 0.59}$$

weighted 5-NN

$$d_1 = \sqrt{(1-3)^2 + (1-3)^2} = 2.828$$

$$d_2 = \sqrt{(1-3)^2 + (2-3)^2} = 2.236$$

$$d_3 = \sqrt{(1-3)^2 + (4-3)^2} = 2.236$$

$$d_4 = \sqrt{(2-3)^2 + (3-3)^2} = 1$$

$$d_5 = \sqrt{(3-3)^2 + (0-3)^2} = 3$$

$$d_6 = \sqrt{(3-3)^2 + (2-3)^2} = 1$$

$$d_7 = \sqrt{(3-3)^2 + (5-3)^2} = 2$$

$$d_8 = \sqrt{(4-3)^2 + (4-3)^2} = 1.414$$

$$d_9 = \sqrt{(5-3)^2 + (3-3)^2} = 2$$

AVERAGE

Grading: 9 points

Comments: missing results for weighted KNN

Eklie Bulley
April 5, 2022

Homework 3

X_1	X_2	Class	$y = [1, 22]$	^{Squared} distance	Rank
1	1	+	$(1-1)^2 + (1-22)^2 = 441$	8	
1	2	-	$(1-1)^2 + (2-22)^2 = 440$	6	
1	4	-	$(1-1)^2 + (4-22)^2 = 324$	2	
2	3	+	$(2-1)^2 + (3-22)^2 = 362$	4	
3	0	+	$(3-1)^2 + (0-22)^2 = 488$	9	
3	2	+	$(3-1)^2 + (2-22)^2 = 404$	7	
3	5	-	$(3-1)^2 + (5-22)^2 = 293$	1	
4	4	-	$(4-1)^2 + (4-22)^2 = 333$	3	
5	3	-	$(5-1)^2 + (3-22)^2 = 377$	5	

1. 1NN
The NN (3,5) is in a negative class.

3NN
The top 3 NN are (3,5) (1,4) (4,4). These 3 are all in the negative class.

When the Nearest Neighbor (NN) are 1NN and 3NN the class is negative. Because on the chart the points with the closest distance are all classified as negative.

Eldric Bulley

X_i	X_c	Class	$(x_1 - 3)^2 + (x_2 - 3)^2$	$(x_3 - 3)^2 + (x_4 - 3)^2$	distance
1	1	+	$(1-3)^2 + (1-3)^2$	$-2^2 + -2^2$	8
1	2	-	$(1-3)^2 + (2-3)^2$	$-2^2 + -1^2$	5
1	4	-	$(1-3)^2 + (4-3)^2$	$-2^2 + 1^2$	5
2	3	+	$(2-3)^2 + (3-3)^2$	$1^2 + 0^2$	1
3	0	+	$(3-3)^2 + (0-3)^2$	$0^2 + -3^2$	9
3	2	+	$(3-3)^2 + (2-3)^2$	$0^2 + -1^2$	1
3	5	-	$(3-3)^2 + (5-3)^2$	$0^2 + 2^2$	4
4	4	-	$(4-3)^2 + (4-3)^2$	$1^2 + 1^2$	2
5	3	-	$(5-3)^2 + (3-3)^2$	$2^2 + 0^2$	4

$$\frac{9-4}{8} \quad \frac{9-5}{8} \quad \frac{9-5}{8} \quad \frac{9-1}{8} \quad \frac{9-9}{8} \quad \frac{9-1}{8} \quad \frac{9-4}{8}$$

$$-\frac{7}{8} \quad -\frac{5}{8}$$

$$\frac{9-2}{8} \quad \frac{9-4}{8}$$

The top 5 NN are (2,3), (3,2), (3,5), (4,4), (5,3)

$$+ \frac{8}{8} + \frac{8}{8} = \frac{16}{8} = 2$$

$$- \frac{5}{8} - \frac{7}{8} - \frac{5}{8} = \frac{-17}{8} = 2\frac{1}{8}$$

BELOW AVERAGE

Comments: missing the subtask of weighted KNN

	X_1	X_2	Class	$D(X, y)$	$D(X, y)$
1	1	1	+	21	4
2	1	2	-	20	3
3	1	4	-	18	3
4	2	3	+	20	1
5	3	0	+	24	3
6	3	2	+	22	1
7	3	5	-	19	2
8	4	4	-	21	2
9	5	3	-	23	2

1-NN

Min distance at 3 and class is - So it is assigned to
- class 1 $y = (1, 22)$

3-NN

The point is assigned '-1' class

5-NN

Point is assigned to Negative class due to vote is 3 so it is assigned to Negative class

ST/MACHINE LEARNING SEMESTER PROJECT FALL 2022

PROJECT TITLE: Building a Classifier to Judge Whether Patients Have Diabetes

TOPICS COVERED IN PROJRCT

1. Python Programming
2. Feature Selection and Sampling
3. Bayesian Classifiers
4. Nearest-Neighbor Classifiers
5. Linear and Polynomial Classifiers
6. Decision Tree
7. Artificial Neural Networks

ST/MACHINE LEARNING SEMESTER PROJECT FALL 2022

PROJECT PROBLEM STATEMENT

Diabetes is a disease that occurs when the blood glucose sugar is quite high.

Detailed steps are below.

- Download the data (Pima Indians Diabetes)
(<https://gist.github.com/ktisha/c21e73a1bd1700294ef790c56c8aec1f>)
- Splitting the data into training and testing sets with splitting rate 0.8
- Processing missing value on the training and testing sets to generate the preprocessed training set D1 and the preprocessed testing set D2
- Selecting 5 features on D1 with chi2 to conduct D3
- Building a classifier on D3
- Testing the classifier on D2 with the 5 selected features and evaluating the testing results with evaluation metrics, namely, precision, recall, and F-score, where the testing means to perform prediction on D2
- Comparing the performance between the model built on the raw data and that built on the preprocessed data and providing some reasons to explain the performance differences

REQUIREMENTS

- Submit the source code (jupyter notebook files) and the data you use for the project, where the codes have no bugs.
- Write and submit the project report with the following parts: (25 points)
 - Project description
 - Model description
 - Result analysis
- Project Presentation (25 points)
 - 10 ~ 15 slides
 - 10 ~ 15 minutes' presentation for each student
 - Slide Outline
 - Project description
 - Model description
 - Result analysis

**ABOVE AVERAGE
PROJECT REPORT AND PRESENTATION**

Report grading: 25 points

Prairie View A&M University

ROY G. PERRY COLLEGE OF ENGINEERING



Submitted by: Osesiye Ayeni

ELEG 4310 – P01

ST/Machine Learning

Spring 2022 – Project

Due: 05/10/2022

Building a classifier

Task description

The objective of this project is to build a machine learning model to predict based on diagnostic measurements whether a patient has diabetes. This is a binary classification project with supervised learning.

The following tasks were performed:

1. Download the data (Pima Indians Diabetes)
(<https://gist.github.com/ktisha/c21e73a1bd1700294ef790c56c8aec1f>)
2. Splitting the data into training and testing sets with splitting rate 0.8
3. Processing missing value on the training and testing sets to generate the preprocessed training set D1 and the preprocessed testing set D2
4. Selecting 5 features on D1 with chi2 to conduct D3
5. Building a classifier on D3
6. Testing the classifier on D2 with the 5 selected features and evaluating the testing results with evaluation metrics, namely, precision, recall, and F-score, where the testing means to perform prediction on D2
7. Comparing the performance between the model built on the raw data and that built on the preprocessed data and providing some reasons to explain the performance differences

Method description

Data Processing

The real-world data is unclear and must be pre-processed so that it can be used for building machine learning models. The data was obtained from [Pima Indians Diabetes Database](#).

Step 1: Import relevant libraries

The standard libraries of Pandas and Numpy are imported, along with visualization libraries of Matplotlib and Seaborn. The Pandas and NumPy libraries were used for data cleaning purposes. The pandas library is used for reading CSV files while the NumPy library is used for finding and replacing missing values.

```
[ ] # Importing libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

Step 2: Load data & understand the data with descriptive statistics

To predict the likelihood of diabetes, it is important to visualize the basic details about the dataset. The data set is loaded, below is the python code for loading the data set. Used Panda to read the cvs file “pd.read.csv”.

```
# load data
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
df_names = {'preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class'}
diab = pd.read_csv(url, names=df_names)

[3] # Prints the first 5 rows of data
diab.head()
```

Review of the first five rows of the data

	preg	plas	pres	skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

There are 768 observations with 8 medical predictor features (input) and 1 target variable (output 0 for “no diabetes” or 1 for yes”). Most of the attributes are integers and the mass and pedi are floating point.



diab.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
#   Column   Non-Null Count  Dtype
---  ---
0    preg    768 non-null    int64
1    plas    768 non-null    int64
2    pres    768 non-null    int64
3    skin    768 non-null    int64
4    test    768 non-null    int64
5    mass    768 non-null    float64
6    pedi    768 non-null    float64
7    age     768 non-null    int64
8    class   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Although in this dataset none of the columns contain missing values, some of the measurements (Glucose, Blood Pressure, Skin Thickness, Insulin and BMI) have values of 0, which is not possible for a living human. Organism.

```
# SUMMARY STATISTICS
diab.describe(include = 'all')
```

	preg	plas	pres	skin	test	mass	pedi	age	class
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.884160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.243750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

Below shows they are no missing values

```
# checking no of missing values
diab.isnull().sum()
```

```
preg      0
plas      0
pres      0
skin      0
test      0
mass      0
pedi      0
age       0
class     0
dtype: int64
```

With the knowledge of the data set we know missing values exist, so a code is written to count the zeros in each column

```
[ ] diab.columns
Index(['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class'], dtype='object')
```

```
# count of
diab[diab[diab.columns] == 0].count()

preg      111
plas       5
pres      35
skin     227
test     374
mass      11
pedi       0
age        0
class    500
dtype: int64
```

Step 3: Replace missing values with Mean

There are some zeros in the data set, which will affect the training accuracy, which will affect the training accuracy. I replaced them with the mean value. To replace zero with mean value, first we will have to replace the zero by NaN, then replace Nan by median (because NaN will not influence the median).

```
[ ] # Mark zero values as NaN
diab[['preg', 'plas', 'pres', 'skin', 'test', 'mass']] = diab[['preg', 'plas', 'pres', 'skin', 'test', 'mass']].replace(0, np.NaN)
```

Below is the number of Nan values in each column

```
[ ] # Count the number of NaN values in each column
diab.isnull().sum()

preg      111
plas       5
pres      35
skin     227
test     374
mass      11
pedi       0
age        0
class      0
dtype: int64
```

Replaced missing values in each column with Mean

```
[ ] # Replace missing values in each column with Mean
    # diab = diab.fillna(diab.mean())
    diab.fillna(diab.mean(),inplace = True)
```

After replacing the missing values with mean, you can below there is no missing values in the column.

```
[ ] # checking if missing values has been handled
    diab.isnull().sum()

preg      0
plas      0
pres      0
skin      0
test      0
mass      0
pedi      0
age       0
class     0
dtype: int64
```

Step 4: Splitting the data

In completion of processing the data, we split the data into training and testing sets with splitting the data to 80:20 ratio. Where D1 becomes the training set D2 the testing set. Used panda DataFrames method `sample()` which returns a random sample of items from an axis of object. Also made use of `random_state` for reproducibility. The code used to split the data for training and testing: The `random_state` corresponds to the seed, so that results are reproducible. We create the set by dropping the corresponding indices from the original DataFrame which are now included in the training set.

```
[ ] # Splitting the data into training and testing using pandas
D1 = diab.sample(frac=0.8, random_state=25) # training set
D2 = diab.drop(D1.index) # testing set

print(f"No. of training set: {D1.shape[0]}")
print(f"No. of testing set: {D2.shape[0]}")

No. of training set: 614
No. of testing set: 154
```

As we can see the training set contains 614, which aligns with the fraction that was requested when sampling the original modelling DataFrame. The remaining 154 were packed into the testing set.

Feature Selection

The pre-processed data is used in this step. The feature selection and feature reduction methods are used. Feature selection is used to remove redundant or irrelevant features without loss of information or data. Feature Reduction is the transformation of data from a high-dimensional space into a low-dimensional space where low-dimensional representation retains meaningful properties of the original data.

Step 1: Treating outliers and non-Normality

Outliers are extreme values existing in the dataset. It is necessary to treat outliers if a distance-based algorithm (logistic regression, SVM, etc) is applied on the dataset. In this case we are going to be selecting 5 features on D1. Splitting the training set D1 for feature selection, separating the independent features and dependent features.

```
# splitting the training set D1 for feature selection
x = D1.drop(['class'], axis = 1) # independent features
y = D1['class'] # dependent features
y.value_counts()
```

```
0    402
1    212
Name: class, dtype: int64
```

Step 2: Feature selection with chi-square

Feature selection is also known as attribute selection is a process of extracting the most relevant features from the dataset and then applying machine learning algorithms for the better performance of the model. Many irrelative features increase the training time exponentially and increase the risk of overfitting. square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with best Chi scores. It determines if the association between two categorical variables of the sample would reflect their real association in the population.

```
# feature selection with chisquare
from sklearn.feature_selection import chi2, SelectKBest
best5_features = SelectKBest(score_func = chi2, k = 5)
features = best5_features.fit_transform(x, y)
```

```
# getting the best 5 features name
best5_features.get_feature_names_out(input_features=None)

array(['plas', 'skin', 'test', 'mass', 'age'], dtype=object)
```

Selected 5 features with highest chi-squared statistics are selected on D1 to conduct D3. For our given dataset we have plas, skin, test, mass, and age as the best selected 5 features.

```
# creating training set dataframe D3 using the best 5 features
D3 = D1[['plas', 'skin', 'test', 'mass', 'age', 'class']]
D3.head()
```

	plas	skin	test	mass	age	class
459	134.0	33.00000	60.000000	25.9	81	0
39	111.0	47.00000	207.000000	37.1	56	1
344	95.0	29.15342	155.548223	36.8	57	0
84	137.0	29.15342	155.548223	48.8	37	1
700	122.0	27.00000	200.000000	35.9	26	0

The training set dataframe D3 using the best 5 features

Classification model

Machine learning techniques can be generally divided into either supervised or unsupervised methods. Unsupervised learning techniques mainly involve clustering and regression such as principal component analysis (PCA) for data dimensionality reduction. These unsupervised techniques do not require labeled data so that the model output is predicted or classified based on patterns on the input data.

In comparison, supervised techniques require required the data to be labelled in addition to being split into training and testing datasets. Supervised learning techniques mainly deal with classification (training a model to predict what animal is in a photo) or regression (predicting the price of a house based on number of rooms, neighborhood, area, etc.). However, in the case of supervised classification algorithms, they cannot predict an animal if it was not present during model training. Data is split so that the model can be evaluated on data it has not seen

before (i.e., testing data). After a model has been fitted a confusion matrix can be generated to evaluate the model in addition to other metrics.

```
# splitting the D3 training set
x_train= D3.drop(['class'], axis = 1) # independent features
y_train = D3['class'] # dependent features
print(x)
print(y)

# creating testing set dataframe D4 using the best 5 features
D4 = D2[['plas', 'skin', 'test', 'mass', 'age', 'class']]
D4.head()
```

	plas	skin	test	mass	age	class
8	197.0	45.000000	543.000000	30.5	53	1
14	166.0	19.000000	175.000000	25.8	51	1
15	100.0	29.15342	155.548223	30.0	32	1
20	126.0	41.000000	235.000000	39.3	27	0
22	196.0	29.15342	155.548223	39.8	41	1

Before classifying our data with logic regression, D3 training set is split so we can test the classifier on D2 with the 5 selected features. We used logistic regression

Logistic Regression

Logistic regression (or logit regression) is a supervised classification algorithm (despite regression in the name) that is used to estimate the probability that an occurrence belongs to a classification (e.g. spam folder of email) where a threshold probability of >50% the model will predict that the occurrence belongs to the positive class (denoted as 0, i.e. normal email) or else predicts that the classification belongs to the negative class (denoted as 1, i.e. spam). The logistic function (denoted as σ) is a sigmoid function with the output bound between 0 and 1.

```
[22] from sklearn.linear_model import LogisticRegression
      lr = LogisticRegression()
      lr.fit(x_train, y_train) # training the model with training set

      LogisticRegression()
```

```
lr_pred = lr.predict(x_test) #predicting on testing set
```

Building the logistic Regression model using the features selected

```
[ ] y_test

      8      1
      14     1
      15     1
      20     0
      22     1
      ..
     756     0
     757     1
     761     1
     765     0
     766     1
      Name: class, Length: 154, dtype: int64
```

Predicting the result and checking the accuracy

```
from sklearn.metrics import classification_report
print(classification_report(y_test, lr_pred)) # this is the report
```

	precision	recall	f1-score	support
0	0.80	0.83	0.81	98
1	0.68	0.64	0.66	56
accuracy			0.76	154
macro avg	0.74	0.73	0.74	154
weighted avg	0.76	0.76	0.76	154

The scikit-learn library provides a convenience report when working on classification problems to give us a quick idea of the accuracy of a model using a few measures. The `classification_report()` function displays the precision, recall, F1-score and support for each class. The accuracy using the logistic regression model is 76%, comparing it to using the K Neighbors classifier and SVC, the percentage is higher.

The definitions for the evaluation:

Precision: The precision is the ratio $tp / (tp + fp)$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

Recall: is the ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

F-score: can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0. weights recall more than precision by a factor of beta. $\beta = 1.0$ means recall and precision are equally important.

The support is the number of occurrences of each class in `y true`

K Neighbors Classifier

The k-Nearest Neighbors algorithm (or KNN) uses a distance metric to find the k most similar instances in the training data for a new instance and takes the mean outcome of the neighbors as the prediction.

```
[ ] from sklearn.neighbors import KNeighborsClassifier
    knn = KNeighborsClassifier()
    knn.fit(x_train, y_train) # training the model with training set
    knn_pred = knn.predict(x_test) #predicting on testing set
```

```
[ ] print(classification_report(y_test, knn_pred)) # this is the report for 1
```

	precision	recall	f1-score	support
0	0.82	0.76	0.79	98
1	0.62	0.71	0.67	56
accuracy			0.74	154
macro avg	0.72	0.73	0.73	154
weighted avg	0.75	0.74	0.74	154

The accuracy for the testing set using KNN classifier is 75%

```
[ ] from sklearn.svm import SVC
    svc = SVC()
    svc.fit(x_train, y_train) # training the model with training set
    svc_pred = svc.predict(x_test) #predicting on testing set

[ ] print(classification_report(y_test, svc_pred)) # this is the report for 1
```

	precision	recall	f1-score	support
0	0.74	0.87	0.80	98
1	0.67	0.46	0.55	56
accuracy			0.72	154
macro avg	0.70	0.67	0.67	154
weighted avg	0.71	0.72	0.71	154

The accuracy for the testing set using SVC classifier is 71%

Result analysis

The dataset included several missing values that were pre-processed using the NumPy library. The missing values were dropped, and a machine learning model was built using logistic regression. The original dataset has an accuracy mean of 0.7540581487627156 (first 500 rows). The same dataset was used for feature selection. First, the SelectKBest model was used for feature extraction using the chi-squared method. The same machine learning method was built and cross-validated. The accuracy mean was measured to be 0.7620902772767718. Finally, the feature reduction method has been used that compresses the data without losing information. The same machine learning method was applied on the new dataset and the accuracy mean was found to be 0.7620902772767718 which is the same as found in the case of feature extraction. It means that feature extraction and feature reduction have similar results where the feature reduction method compresses data significantly.

Prairie View A&M University

In conclusion, in this project the logistic regression model has achieved a prediction (precision) score of 76%, out of all diabetic patients, 76% of them will be correctly classified using medical diagnostic measurements.

Presentation grading: 24 points

Comment: missing result analysis on precision, recall, and fscore.



PRAIRIE VIEW
A&M UNIVERSITY

ELEG 4310-P01 – Building a classifier to judge whether the patient has diabetes

Department of ELECTRICAL
ENGINEERING

May 10, 2021

Osesiye Ayeni

Introduction

Diabetes is a chronic condition that causes a person's blood sugar level to become too high. The Pima Indians of Arizona for nearly 30 years, researchers are certain of one thing: The switch to a high-fat diet is making the Native American group sick at a faster rate than other Americans.

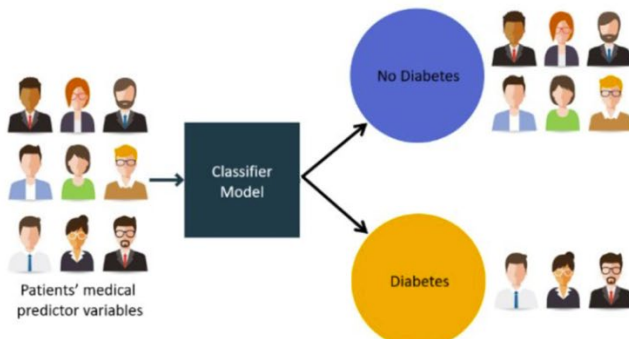
The data from the study of genetically similar group of Pimas has shown that the Arizona Pimas are sicker than their Mexican relatives, with higher rates of diabetes, obesity and kidney disease.

Several constraints were placed on the selection of these instances from a larger database:

1. All patients are female
2. All patients are at least 21 years old
3. All patients are of Pima Indian heritage

Objectives

- The objective of this project is to build a machine learning model to predict based on diagnostic measurements whether a patient has diabetes. This is a binary classification project with supervised learning.



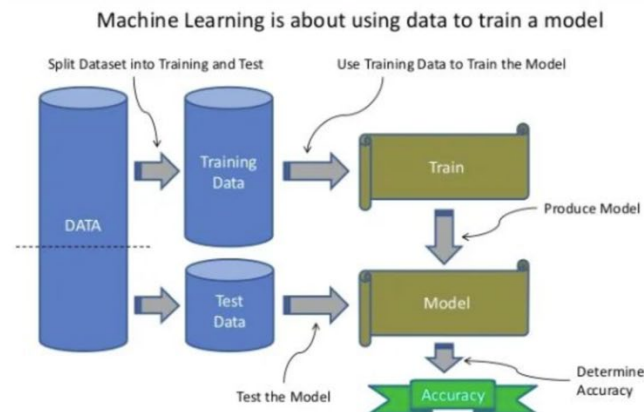
Training and Testing data set

In machine learning, datasets are split into two subsets.

The first subset is known as the training data - it's a portion of our actual dataset that is fed into the machine learning model to discover and learn patterns. In this way, it trains our model.

The other subset is known as the testing data. We'll cover more on this below.

Training data is typically larger than testing data. This is because we want to feed the model with as much data as possible to find and learn meaningful patterns. Once data from our datasets are fed to a machine learning algorithm, it learns patterns from the data and makes decisions.



Chi-Squared Selection Feature

Many irrelative features increase the training time exponentially and increase the risk of overfitting. square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with best Chi scores. It determines if the association between two categorical variables of the sample would reflect their real association in the population.

```
# feature selection with chisquare
from sklearn.feature_selection import chi2, SelectKBest
best5_features = SelectKBest(score_func = chi2, k = 5)
features = best5_features.fit_transform(x, y)

# getting the best 5 features name
best5_features.get_feature_names_out(input_features=None)

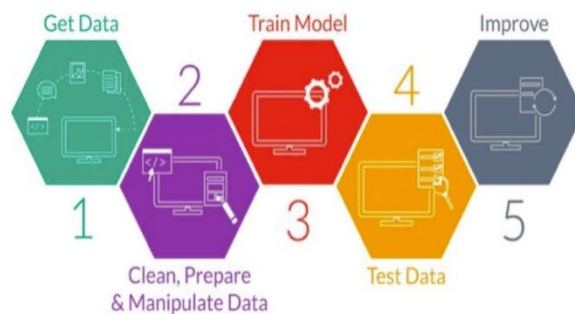
array(['plas', 'skin', 'test', 'mass', 'age'], dtype=object)
```

Building A Classifier

Machine learning techniques can be generally divided into either supervised or unsupervised methods. Unsupervised learning techniques mainly involve clustering and regression such as principal component analysis (PCA) for data dimensionality reduction. These unsupervised techniques do not require labeled data so that the model output is predicted or classified based on patterns on the input data.

Steps to building a classifier:

1. Step 1: Load Python packages. Copy code snippet.
2. Step 2: Pre-Process the data.
3. Step 4: Split the data into train and test sets.
4. Step 5: Build a Random Forest Classifier.
5. Step 6: Predict.
6. Step 7: Check the Accuracy of the Model.
7. Step 8: Check Feature Importance.

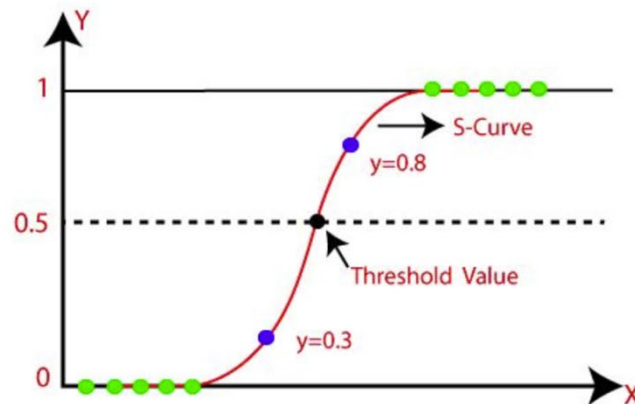


Machine Learning Work Flow

Logistic Regression

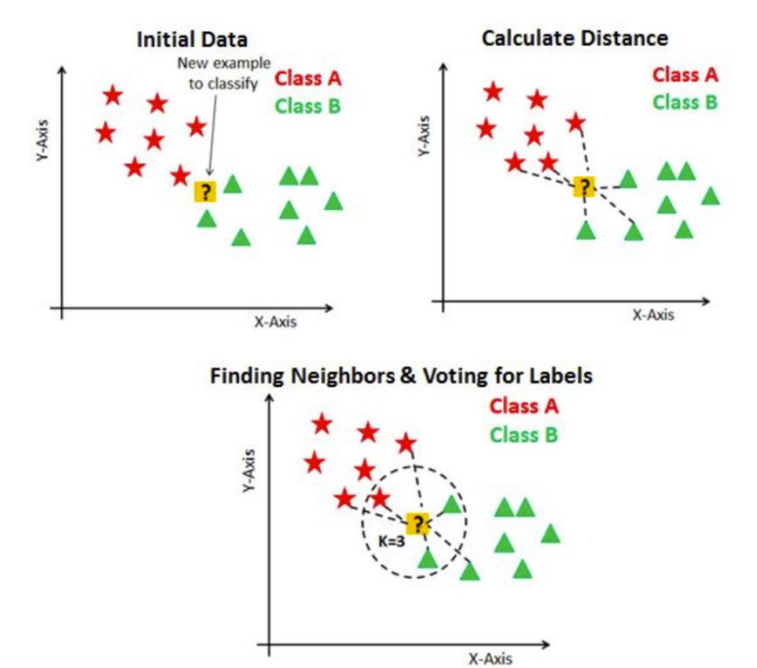
Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function.



K - Nearest Neighbors

The k-Nearest Neighbors algorithm (or KNN) uses a distance metric to find the k most similar instances in the training data for a new instance and takes the mean outcome of the neighbors as the prediction.



Precision

The precision is the ratio $tp / (tp + fp)$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.

- True Positive (TP): The actual positive class is predicted positive.
- True Negative (TN): The actual negative class is predicted negative.
- False Positive (FP): The actual class is negative but predicted class is Positive
- False Negative (FN): The actual class is positive but predicted class is negative.

		Predicted	
		Positive	Negative
Actual	Positive	TP = 50	FN = 10
	Negative	FP = 5	TN = 20

Recall

is the ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.

```
from sklearn.metrics import classification_report
print(classification_report(y_test, lr_pred)) # this is the report
```

	precision	recall	f1-score	support
0	0.80	0.83	0.81	98
1	0.68	0.64	0.66	56
accuracy			0.76	154
macro avg	0.74	0.73	0.74	154
weighted avg	0.76	0.76	0.76	154

F-Score

can be interpreted as a weighted harmonic mean of the precision and recall, where an F-beta score reaches its best value at 1 and worst score at 0. weights recall more than precision by a factor of beta. $\beta = 1.0$ means recall and precision are equally important.

Finally we have the F1-score, which takes both precision and recall into account to ultimately measure the accuracy of the model. But what's the difference between this metric and accuracy? Well as we talked about in the beginning, false positives and false negatives can be absolutely crucial to the study, while true negatives are often less important to what ever problem you're trying to solve especially in a business setting. The F1 score tries to take this into account, giving more weight to false negatives and false positives while not letting large numbers of true negatives influence your score.

AVERAGE

PROJECT REPORT AND PRESENTATION

Report grading: 18 points

Comment: missing results on precision, recall, and f score

Eldric Bulley
May 5, 2022
ELEG 4
Professor Dong

“Binary Classifier”

“Simulating the behavior of 100 billion neurons of human brain is not feasible by classical computer but quantum machine learning promises to fulfill that requirement” is a famous quote once said by Amit ray. Being able to implement codes is a primary part of building a machine that can do the unimagined. In the machine learning class instructed by Professor Dong we as Class had assigned us a project to create a Binary classifier and implement the classifier in a python code. As I created the code, I relearned logistic regression techniques as well as numerous equation such as precision, recall and f score and at the end of the python code we compared the results of the equations from the raw data to the new data. In this Report I will demonstrate step by step on how to create this specialized code.

First, I downloaded the information from the website that was provided. I then uploaded the website to callabotory. After, printing the data to calaboatory I was then able to start processing the data. I want to have 5 features for the data.

Next, I want to rid of the missing values inside the data, so I start by labeling the missing values “NaN” and later changing the missing values to the mean value of the data listed. This way the data can be more precise. By changing the value to the mean value, I can specify my data to a more specific group.

Lastly, after classifying my data and finding the best features I am now able to start adding equations to the code. Now I add precision, recall and f score to the code so I can compare the data.

To conclude this project, the f-score, precision and recall were all processed as 1. The original mean for the data is .7885864793678665 and the new mean is. 7827280245829675.I conclude that These 2 values are similar.

Sources

<https://quotefancy.com/quote/2673190/Amit-Ray-Simulating-the-behavior-of-100-billion-neurons-of-human-brain-is-not-feasible-by>

https://en.wikipedia.org/wiki/Binary_classification

<https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv>

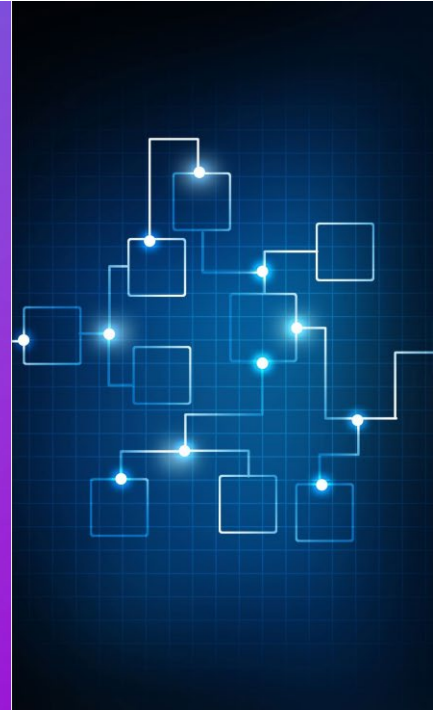
<https://en.wikipedia.org/wiki/F-score>

Presentation grading: 23 points

Comment: missing detailed results and analysis on precision, recall, and fscore.

BY: ELDRIC BULLEY

BINARY CLASSIFIER PROJECT

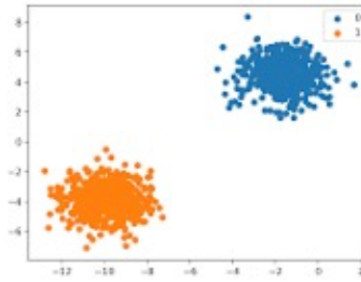


PIMA INDIAN DATA BASE

- <https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv>



BINARY CLASSIFIER



- Binary Classification. Binary classification refers to those classification tasks that have two class labels.



K FOLD

K FOLD

- K-Fold is validation technique in which we split the data into k-subsets and the holdout method is repeated k-times where each of the k subsets are used as test set and other k-1 subsets are used for the training purpose.

A decorative background featuring a grid of numbers in a light blue font, arranged in a pattern that suggests a data table. A large, bold red 'X' shape is superimposed over the grid, extending from the top-left to the bottom-right. The background is framed by a teal bar at the top and a blue bar at the bottom.

PROCESSING MISSING VALUE

PROCESSING MISSING VALUES

- There are many ways to process missing values but I used the mean value method.

MEAN

Find the Mean:

15, 10, 19, 19, 7, 11, 15, 19, 20, 12, 17, 18

$$\text{Mean} = \frac{\text{Sum of the numbers}}{\text{How many numbers}} = \frac{182}{12}$$



SPLITTING UP THE DATA

CHI2 METHOD

$$\chi^2_c = \sum \frac{(O_i - E_i)^2}{E_i}$$

chi-squared test is used to determine whether there is a statistically significant difference between the expected frequencies and the observed frequencies in one or more categories of a contingency table.

PRECISION

- The precision is the ratio $tp / (tp + fp)$ where tp is the number of true positives and fp the number of false positives. The precision is intuitively the ability of the classifier not to label as positive a sample that is negative.



RECALL

- The recall is the ratio $tp / (tp + fn)$ where tp is the number of true positives and fn the number of false negatives. The recall is intuitively the ability of the classifier to find all the positive samples.



F - SCORE

THE F-SCORE, ALSO CALLED THE F1-SCORE, IS A MEASURE OF A MODEL'S ACCURACY ON A DATASET.

The F-score

The F-score is the Harmonic mean of Precision and Recall.

$$F = \frac{2}{\frac{1}{\text{Recall}} + \frac{1}{\text{Precision}}}$$

Alternatively

$$F = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

COMPARING THE PERFORMANCE

- To conclude this project, the f-score, precision and recall were all processed as 1. The original mean for the data is .7885864793678665 and the new mean is .7827280245829675. I conclude that These 2 values are similar.

```

-----
DataFrame, dtype='object')

# Summarize the number of rows and columns in the dataset
print(DataFrame.shape)

# Splitting the data using pandas
training_data = DataFrame.sample(frac=0.8, random_state=25)
testing_data = DataFrame.drop(training_data.index)

print("No. of training examples: {}".format(training_data.shape[0]))
print("No. of testing examples: {}".format(testing_data.shape[0]))

# Extract data
array = DataFrame.values
X = array[:,0:8] # 0 - 7 columns for features
Y = array[:,8] # 8th column for the label
print(X)
print(Y)

# Evaluate an Logistic Regression (LR) model with Accuracy
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter=1000)
kfold = KFold(n_splits=5) # 5-fold cross validation
result = cross_val_score(model, X, Y, cv=kfold, scoring='accuracy')

# Display the mean of Accuracy of 5-fold cross validation
print('Accuracy mean of 5-fold cross validation is ', result.mean())

# Scoring features with chi2
fs_chi2 = SelectKBest(score_func=chi2, k=5)
fs = fs_chi2.fit(X, Y)
print(fs.feature_names_)

# Selecting features with these scores
feature_selected = fs.transform(X)
# Evaluate an Logistic Regression (LR) model with Accuracy on Selected Features
# using k-fold cross validation
model = LogisticRegression(random_state=0, max_iter=1000)
kfold = KFold(n_splits=5) # 5-fold cross validation
fs_result = cross_val_score(model, feature_selected, Y, cv=kfold, scoring='accuracy')
# Displaying selected features
print(feature_selected[0:5,:])

# Display the mean of Accuracy of 5-fold cross validation
print('Accuracy mean of 5-fold cross validation is ', fs_result.mean())

y_true = np.array(['preg', 'plas', 'pres', 'skin', 'test'])
y_pred = np.array(['preg', 'plas', 'pres', 'skin', 'test'])
precision_recall_fscore_support(y_true, y_pred, average='macro')
precision_recall_fscore_support(y_true, y_pred, average='micro')
precision_recall_fscore_support(y_true, y_pred, average='weighted')
precision_recall_fscore_support(y_true, y_pred, average=None,
labels=['preg', 'plas', 'pres', 'skin', 'test'])

#Comparing performance between original data
# and data processing with feature selection
print('Original Accuracy: ', result.mean())
print('VS')
print('FS Accuracy: '.format(fs_result.mean()))

```

CODE

SOURCES

- <https://machinelearningmastery.com/types-of-classification-in-machine-learning/>
- https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
- <https://deeptai.org/machine-learning-glossary-and-terms/f-score>

BELOW AVERAGE

PROJECT REPORT AND PRESENTATION

Report grading: 0 point

Comment: no report submitted

Presentation grading: 23 points

Comment: missing detailed method introduction.

ST/Machine Learning Final Project

Building a Classifier to determine Diabetes



By Kevin Mcdonald
For PVAMU ELEG 4310
Dr. Dong's "Machine Learning" Course

Project Description:

The point of this project was to test our knowledge of Machine Learning concepts like that of preprocessing data, feature selection, classifiers, etc etc... To prove our ability and understanding we were tasked with building a model that could accurately predict if a patient has diabetes based upon the Pima Indians Diabetes data. We will split the data, select five features, build a classifier, test the classifier, then compare the model performance of two models; one built on preprocessed data and one on raw data.

Model Description :

My model is separated into Four/Five sections:

- Initializing Data/Libraries/Variables
- Splitting/preprocessing Data
- Feature Selection
- Classifying
- Comparison

Initializing Data/Libraries/Variables

```
[85] #import pac
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.feature_selection import chi2
from sklearn.feature_selection import SelectKBest
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from pandas.core.frame import DataFrame
from sklearn.metrics import precision_recall_fscore_support

[88] url = "https://raw.githubusercontent.com/jbrownlee/datasets/master/pima-indians-diabetes.data.csv"
names = ['Preg', 'Gluc', 'Blood', 'Skin', 'Sugar', 'Mass', 'DP', 'Age', 'Outty']
RAWD = pd.read_csv(url, names=names)
Pdata = RAWD

[89] print(Pdata.describe())

[90] #variables
Pdata = RAWD.iloc[:, 0:9]
c1 = Pdata['Preg'].mean()
c2=Pdata['Gluc'].mean()
c3=Pdata['Blood'].mean()
c4=Pdata['Skin'].mean()
c5=Pdata['Sugar'].mean()
c6=Pdata['Mass'].mean()
c7=Pdata['DP'].mean()
c8=Pdata['Age'].mean()
```

```
Pdata[['Preg', 'Gluc', 'Blood', 'Skin', 'Sugar', 'Mass', 'DP', 'Age']] = Pdata[['Preg', 'Gluc', 'Blood', 'Skin', 'Sugar', 'Mass', 'DP', 'Age']].replace(0, 'NaN')
print(Pdata)
```

```
0    8  148  72  35  NaN  23.6  0.627  50  1
1    1   85  66  30  NaN  25.0  0.151  31  0
2    0  103  64  NaN  NaN  23.3  0.072  32  1
3    1   89  66  23  94  28.1  0.167  21  0
4  NaN  137  48  35  159  43.1  2.268  33  1
...
763  10  101  76  48  130  32.0  0.171  63  0
764  2  122  76  27  100  30.0  0.340  27  0
765  5  121  72  23  112  28.2  0.245  30  0
766  1  120  68  NaN  NaN  30.1  0.149  47  1
767  1   91  70  31  100  30.4  0.135  23  0
```

[768 rows x 9 columns]

```
#fill missing values with mean
Pdata[['Age']] = Pdata[['Age']].replace('NaN', c6)
Pdata[['DP']] = Pdata[['DP']].replace('NaN', c7)
Pdata[['Mass']] = Pdata[['Mass']].replace('NaN', c5)
Pdata[['Sugar']] = Pdata[['Sugar']].replace('NaN', c4)
Pdata[['Skin']] = Pdata[['Skin']].replace('NaN', c4)
Pdata[['Blood']] = Pdata[['Blood']].replace('NaN', c3)
Pdata[['Gluc']] = Pdata[['Gluc']].replace('NaN', c2)
Pdata[['Preg']] = Pdata[['Preg']].replace('NaN', c1)
print(Pdata)
```

```
0    8.000000  148.0  72.0  35.000000  76.799479  33.6  0.627  50  1
1    1.000000   85.0  66.0  30.000000  70.799479  25.0  0.151  31  0
2    0.000000  103.0  64.0  20.536458  70.799479  23.3  0.072  32  1
3    1.000000   89.0  66.0  23.000000  94.000000  28.1  0.167  21  0
4    5.949552  137.0  48.0  35.000000  159.000000  43.1  2.268  33  1
...
763  10.000000  101.0  76.0  48.000000  130.000000  32.0  0.171  63  0
764  2.000000  122.0  76.0  27.000000  100.000000  30.0  0.340  27  0
765  5.000000  121.0  72.0  23.000000  112.000000  28.2  0.245  30  0
766  1.000000  120.0  68.0  20.536458  70.799479  30.1  0.149  47  1
767  1.000000   91.0  70.0  31.000000  100.000000  30.4  0.135  23  0
```

[768 rows x 9 columns]

Preprocessing the raw data

```
[94] R_training_data = RAWD.sample(frac=0.8, random_state=26)
R_testing_data = RAWD.drop(R_training_data.index)
```

```
print(f"No. of training examples: {R_training_data.shape[0]}")
print(f"No. of testing examples: {R_testing_data.shape[0]}")
```

```
No. of training examples: 614
No. of testing examples: 154
```

```
[95] training_data = processedD.sample(frac=0.8, random_state=25)
testing_data = processedD.drop(training_data.index)
```

```
print(f"No. of training examples: {training_data.shape[0]}")
print(f"No. of testing examples: {testing_data.shape[0]}")
```

```
No. of training examples: 614
No. of testing examples: 154
```

```
[96] OGarray1 = R_training_data.values
OG_x = OGarray1[:,0:8]
OG_y = OGarray1[:,8]

OGarray2 = R_testing_data.values
OG_A = OGarray2[:,0:8]
OG_B = OGarray2[:,8]
```

```
[99] arrayD1 = D1.values
X = arrayD1[:,0:8]
Y = arrayD1[:,8]

arrayD2 = D2.values
A = arrayD2[:,0:8]
B = arrayD2[:,8]
```

Splitting the data into Training & Testing Data

```
[15] model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 5)
RD1= cross_val_score(model, OG_x, OG_y, cv = kfold, scoring='accuracy')
print('Accuracy mean of 5-fold cross validation is ', RD1.mean())
```

Accuracy mean of 5-fold cross validation is 0.7605757696921231

```
[16] fs_chi2 = SelectKBest(score_func = chi2, k = 5)
fs = fs_chi2.fit(OG_x, OG_y)
print(fs.scores_)
```

```
[ 108.08854833  1098.50164397   8.79525175  38.16236759  1413.51077882
  101.88445255   5.21775175  160.38554249]
```

```
[17] feature_selected = fs.transform(OG_x)
print(feature_selected[0:5,:])
```

```
[[ 9.  165.  0.  30.4  49. ]
 [ 5.  132.  0.  26.8  69. ]
 [ 3.   84.  106.  31.9  25. ]
 [ 8.  143.  0.  34.9  41. ]
 [ 7.  160.  175.  30.5  39. ]]
```

Testing the selected features to be used on testing data R2D2

Testing the classifier on the training data RD1 and selecting features

```
[18] model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 5)
RD3= cross_val_score(model, feature_selected, OG_y, cv = kfold, scoring='accuracy')
print('Accuracy mean of 5-fold cross validation is ', RD3.mean())
```

Accuracy mean of 5-fold cross validation is 0.7556977209116353

```
[19] model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 5)
R2D2= cross_val_score(model, OG_A, OG_B, cv = kfold, scoring='accuracy')
print('Accuracy mean of 5-fold cross validation is ', R2D2.mean())
```

Accuracy mean of 5-fold cross validation is 0.7264516129032258

```
[20] model = LogisticRegression(random_state=0, max_iter = 1000)
      kfold = KFold(n_splits = 5)
      resultD1 = cross_val_score(model, X, Y, cv = kfold, scoring='accuracy')
      print('Accuracy mean of 5-fold cross validation is ', resultD1.mean())
```

Accuracy mean of 5-fold cross validation is 0.7785019325603092

```
[21] fs_chi2 = SelectKBest(score_func = chi2, k = 5)
      fs = fs_chi2.fit(X, Y)
      print(fs.scores_)
```

```
[ 66.7185308 1109.12204082 45.47909911 86.92552453 1872.8586048
 84.7211432 5.15551051 141.78150281]
```

```
[22] feature_selected = fs.transform(X)
      print(feature_selected[0:5,:])
```

```
[[134.    33.    60.    25.9    81.
 111.    47.   207.    37.1    56.
 [ 95.   20.53645833 79.79947917 36.8    57.
 [137.   20.53645833 79.79947917 48.8    37.
 [122.   27.    200.    35.9    26.]
```

Testing the selected features to be used on testing data D2



Testing the classifier on the training data D1 and selecting features

```
model = LogisticRegression(random_state=0, max_iter = 1000)
kfold = KFold(n_splits = 5)
resultD3 = cross_val_score(model, feature_selected, Y, cv = kfold, scoring='accuracy')
print('Accuracy mean of 5-fold cross validation is ', resultD3.mean())
```

Accuracy mean of 5-fold cross validation is 0.7687591630014661

```
[24] model = LogisticRegression(random_state=0, max_iter = 1000)
      kfold = KFold(n_splits = 8)
      resultD2 = cross_val_score(model, A, B, cv = kfold, scoring='accuracy')
      print('Accuracy mean of 5-fold cross validation is ', resultD2.mean())
```

Accuracy mean of 5-fold cross validation is 0.7526315789473684

Precision, recall, fscore

```
[25] H_true = np.array(['Preg', 'Gluc', 'Blood', 'Skin', 'Sugar', 'Mass', 'DP', 'Age'])
      H_pred = np.array(['Preg', 'Gluc', 'Blood', 'Skin', 'Sugar', 'Mass', 'DP', 'Age'])
      precision_recall_fscore_support(H_true, H_pred, average = 'macro')
      precision_recall_fscore_support(H_true, H_pred, average = 'micro')
      precision_recall_fscore_support(H_true, H_pred, average = 'weighted')
```

(1.0, 1.0, 1.0, None)

```
[26] precision_recall_fscore_support(H_true, H_pred, average=None, labels = ['Preg', 'Gluc', 'Blood',
```

```
array([1., 1., 1., 1., 1., 1., 1., 1.]),
array([1., 1., 1., 1., 1., 1., 1., 1.]),
array([1., 1., 1., 1., 1., 1., 1., 1.]),
array([1, 1, 1, 1, 1, 1, 1, 1]))
```

Result Analysis:

Here we see the comparison between the models built from raw data and the preprocessed data



```
print('Raw Training "RD1" Accuracy', RD1.mean())
print('&')
print('Raw Test "R2D2" Accuracy', R2D2.mean())
print('&')
print('Raw D3 Accuracy', RD3.mean())
print('\n VS \n')
print('Training "D1" Accuracy', resultD1.mean())
print('&')
print('Test "D2" Accuracy', resultD2.mean())
print('&')
print('D3 Accuracy', resultD3.mean())
```

```
Raw Training "RD1" Accuracy 0.7605757696921231
&
Raw Test "R2D2" Accuracy 0.7264516129032258
&
Raw D3 Accuracy 0.7556977209116353

VS

Training "D1" Accuracy 0.7785019325603092
&
Test "D2" Accuracy 0.7526315789473684
&
D3 Accuracy 0.7687591630014661
```