CSC 5741: Lecture #07—Linear Regression, Classification and Clustering

Lighton Phiri
<lighton.phiri@unza.zm>

May 14 2019

Contents

Introduction

During these “hands-on” activities, we look at practical examples of how to clean data and transform it into a form a computer—READ: algorithms—will be able to understand.

In all instances, you are encouraged to make reference to online Python documentation and documentation for specific libraries. You are also encouraged to look up and explore other libraries, especially as you work towards the Mini Projects.

[1]:

```python
# Import all libraries and modules for use during lecture session code walkthrough
import pandas as pd
import re
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import string

from collections import Counter
from IPython.core.interactiveshell import InteractiveShell
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import Binarizer

InteractiveShell.ast_node_interactivity = "all"
pd.set_option('display.latex.repr', True)
pd.set_option('display.latex.longtable', True)
```

Datasets

Example 1: 2017/18 ICT 1110 Continuous Assessment Scores

Dataset

[2]:
```
# Explore 2017/18 ICT 1110 Assessment Scores
!head -n 3 db-unza19-ict1110_2017_18.csv
```

```
StudentID|Gender|Minor|LastName|FirstName|Quiz1|Quiz2|Quiz3|ClassTest1|Quiz4|Quiz5|Quiz6|
Quiz7|ClassTest2|Quiz8|Quiz9|Quiz10|Quiz11|ClassTest3|Quiz12|Quiz13|Quiz14|Quiz15|ClassTe
st4|Quiz16|Quiz17|Quiz18|Quiz19|Quiz20|CA [50 Pts]|CA [%]|PassedTest1|GradeTest1|PassedTe
st2|GradeTest2|PassedTest3|GradeTest3|PassedTest4|GradeTest4|PassedCA|GradeCA
0786c2fc7a89596881a07df429364f9d|M|Geography|FName1|LName1|6|7|4|31|9|8|10|7.5|30|4|8|1|7
|17|9|10|7|10|31|10|5|9|8|10|31.3|62.6|YES|B|YES|B|NO|D|YES|B|YES|B
3283f0f7cce04d7ce5cd5dfdd6d191bf|M|Civic|FName2|LName2|0|0|0|0|0|0|0|0|0|0|0|0|6|8.5|3|0|
0|0|0|0|0|10|6|10|4.78|9.56|NO|D|NO|D|NO|D|NO|D|NO|D
```

[3]:
```
# Create and Inspect DataFrame
var_ict1110_2018_ca = pd.read_csv("db-unza19-ict1110_2017_18.csv", sep="|")

len(var_ict1110_2018_ca)
var_ict1110_2018_ca.columns
###var_ict1110_2018_ca.info()
var_ict1110_2018_ca.describe()
```

[3]:
```
99
```

[3]:
```
Index(['StudentID', 'Gender', 'Minor', 'LastName', 'FirstName', 'Quiz1',
       'Quiz2', 'Quiz3', 'ClassTest1', 'Quiz4', 'Quiz5', 'Quiz6', 'Quiz7',
       'ClassTest2', 'Quiz8', 'Quiz9', 'Quiz10', 'Quiz11', 'ClassTest3',
       'Quiz12', 'Quiz13', 'Quiz14', 'Quiz15', 'ClassTest4', 'Quiz16',
       'Quiz17', 'Quiz18', 'Quiz19', 'Quiz20', 'CA [50 Pts]', 'CA [%]',
       'PassedTest1', 'GradeTest1', 'PassedTest2', 'GradeTest2', 'PassedTest3',
       'GradeTest3', 'PassedTest4', 'GradeTest4', 'PassedCA', 'GradeCA'],
      dtype='object')
```

[3]:

	Quiz1	Quiz2	Quiz3	ClassTest1	Quiz4	Quiz5	Quiz6	Quiz7	ClassTest
count	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.000000	99.00000
mean	5.313131	4.727273	4.898990	26.171717	6.343434	6.757576	8.444444	3.929293	17.39393

	Quiz1	Quiz2	Quiz3	ClassTest1	Quiz4	Quiz5	Quiz6	Quiz7	ClassTest
std	2.023568	2.385407	1.798497	8.882212	2.875518	2.119352	2.763546	1.935187	7.72197
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	4.000000	3.000000	4.000000	22.500000	5.000000	5.000000	8.000000	3.000000	13.00000
50%	6.000000	5.000000	5.000000	26.000000	8.000000	7.000000	10.000000	4.000000	17.00000
75%	7.000000	6.000000	6.000000	31.250000	8.500000	8.000000	10.000000	5.500000	22.50000
max	9.000000	10.000000	8.000000	44.000000	9.000000	10.000000	10.000000	8.500000	35.00000

[4]:
```
# Rename columns using appropriate names
var_ict1110_2018_ca.rename(columns={"CA [50 Pts]": "CAPoints", "CA [%]":␣
 ↪"CAPercentage"}, inplace=True)
var_ict1110_2018_ca.columns
```

[4]:
```
Index(['StudentID', 'Gender', 'Minor', 'LastName', 'FirstName', 'Quiz1',
       'Quiz2', 'Quiz3', 'ClassTest1', 'Quiz4', 'Quiz5', 'Quiz6', 'Quiz7',
       'ClassTest2', 'Quiz8', 'Quiz9', 'Quiz10', 'Quiz11', 'ClassTest3',
       'Quiz12', 'Quiz13', 'Quiz14', 'Quiz15', 'ClassTest4', 'Quiz16',
       'Quiz17', 'Quiz18', 'Quiz19', 'Quiz20', 'CAPoints', 'CAPercentage',
       'PassedTest1', 'GradeTest1', 'PassedTest2', 'GradeTest2', 'PassedTest3',
       'GradeTest3', 'PassedTest4', 'GradeTest4', 'PassedCA', 'GradeCA'],
      dtype='object')
```

[5]:
```
# Inspect DataFrame data
var_ict1110_2018_ca.head(1).T
```

[5]:

	0
StudentID	0786c2fc7a89596881a07df429364f9d
Gender	M
Minor	Geography
LastName	FName1
FirstName	LName1
Quiz1	6
Quiz2	7
Quiz3	4
ClassTest1	31
Quiz4	9
Quiz5	8
Quiz6	10
Quiz7	7.5
ClassTest2	30
Quiz8	4
Quiz9	8
Quiz10	1
Quiz11	7
ClassTest3	17
Quiz12	9
Quiz13	10
Quiz14	7

Continued on next page

	0
Quiz15	10
ClassTest4	31
Quiz16	10
Quiz17	5
Quiz18	9
Quiz19	8
Quiz20	10
CAPoints	31.3
CAPercentage	62.6
PassedTest1	YES
GradeTest1	B
PassedTest2	YES
GradeTest2	B
PassedTest3	NO
GradeTest3	D
PassedTest4	YES
GradeTest4	B
PassedCA	YES
GradeCA	B

Data Cleaning

[6]:
```
# Checking for NULL values
var_ict1110_2018_ca.isnull().values.any()

###var_ict1110_2018_ca[(var_ict1110_2018_ca[["Minor"]].isna().any(axis=1))]
var_ict1110_2018_ca[(var_ict1110_2018_ca.isna().any(axis=1))]
```

[6]:
```
True
```

[6]:

	StudentID	Gender	Minor	LastName	FirstName	Quiz1	Quiz2	Quiz3	C
43	900314909d0823191e822e949c6c22ad	NaN	NaN	FName44	LName44	0	0	0	
45	7c5de7d69b1137270a21bea1f8cd383a	NaN	NaN	FName46	LName46	0	4	5	
72	2759e9fd1e22c1fe44e49c1152d27c62	NaN	NaN	FName73	LName73	0	0	0	
74	0a0ee83d0235f8787c75444bb33635a5	NaN	NaN	FName75	LName75	3	4	6	

Handling NULL values

[7]:
```
# When dealing with Gender variable---if at all we will---we delete all records with␣
 ↪NULL Gender entries
var_ict1110_2018_ca_gender = var_ict1110_2018_ca[(var_ict1110_2018_ca[["Gender"]].
 ↪isna().any(axis=1)==False)]
# When dealing with Minor variable---if at all we will---we delete all records with␣
 ↪NULL MInor entries
var_ict1110_2018_ca_minors = var_ict1110_2018_ca[(var_ict1110_2018_ca[["Minor"]].
 ↪isna().any(axis=1)==False)]
```

Binary Classification

Class Test #4 (Pass/Fail) vs CA Score (Pass/Fail)

- Model uses Class Theory Test 4 to determine if student will pass CA or not
- NOTE: This is just for illustration purposes [...] In an ideal scenario, one would not use this feature in isolation

[8]:
```
var_ict1110_2018_ca_test4 = var_ict1110_2018_ca[["StudentID", "PassedTest4",␣
 ↪"PassedCA"]]
var_ict1110_2018_ca_test4.head(2)
```

[8]:

	StudentID	PassedTest4	PassedCA
0	0786c2fc7a89596881a07df429364f9d	YES	YES
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	NO

1. Apply One Hot Encoding to X inputs

- Remember estimators take in numeric values
- Current X inputs are categorical

[9]:
```
# Use scikit learn to implement On Hot Encoding on X inputs
# Create instance of LabelEncoder
# Remember: You could just as well have used pandas' get dummies implementation for␣
 ↪this
var_ict1110_2018_test4_encoder = LabelEncoder()
```

[10]:
```
# Fit encoder on inputs
var_ict1110_2018_test4_encoder.fit(var_ict1110_2018_ca_test4["PassedTest4"])
```

[10]:
```
LabelEncoder()
```

[11]:
```
# Confirm categorical variables
var_ict1110_2018_test4_encoder.classes_
```

[11]:
```
array(['NO', 'YES'], dtype=object)
```

[12]:
```
# Apply fitted Label Encoder to applicable DataFrame column
var_ict1110_2018_test4_encoder.transform(var_ict1110_2018_ca_test4["PassedTest4"])
var_ict1110_2018_ca_test4.head(2)
```

[12]:
```
array([1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1])
```

[12]:

	StudentID	PassedTest4	PassedCA
0	0786c2fc7a89596881a07df429364f9d	YES	YES

Continued on next page

	StudentID	PassedTest4	PassedCA
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	NO

[13]:
```
# Create new column to hold encoded values
var_ict1110_2018_ca_test4["PassedTest4_X"] = var_ict1110_2018_test4_encoder.
 ↪transform(var_ict1110_2018_ca_test4["PassedTest4"])
var_ict1110_2018_ca_test4.head(2)
```

```
/home/lightonphiri/.local/lib/python3.6/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

[13]:

	StudentID	PassedTest4	PassedCA	PassedTest4_X
0	0786c2fc7a89596881a07df429364f9d	YES	YES	1
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	NO	0

2. Apply Label Encoders to Labels

- Remember estimators take in numeric values
- Current Y are categorical

[14]:
```
# Create instance of LabelEncoder
var_ict1110_2018_ca_encoder = LabelEncoder()
```

[15]:
```
# Fit encoder on inputs
var_ict1110_2018_ca_encoder.fit(var_ict1110_2018_ca_test4["PassedCA"])
```

[15]:
```
LabelEncoder()
```

[16]:
```
# Confirm categorical variables
var_ict1110_2018_ca_encoder.classes_
```

[16]:
```
array(['NO', 'YES'], dtype=object)
```

[17]:
```
# Apply fitted Label Encoder to applicable DataFrame column
var_ict1110_2018_ca_encoder.transform(var_ict1110_2018_ca_test4["PassedCA"])
var_ict1110_2018_ca_test4.head(2)
```

[17]:
```
array([1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0,
       0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0])
```

[17]:

	StudentID	PassedTest4	PassedCA	PassedTest4_X
0	0786c2fc7a89596881a07df429364f9d	YES	YES	1
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	NO	0

[18]:
```
# Create new column to hold encoded values
var_ict1110_2018_ca_test4["PassedCA_Y"] = var_ict1110_2018_ca_encoder.
 ↪transform(var_ict1110_2018_ca_test4["PassedCA"])
var_ict1110_2018_ca_test4.head(2)
```

```
/home/lightonphiri/.local/lib/python3.6/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

[18]:

	StudentID	PassedTest4	PassedCA	PassedTest4_X	PassedCA_Y
0	0786c2fc7a89596881a07df429364f9d	YES	YES	1	1
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	NO	0	0

3. Create Train/Test Sets

- Create test and training datasets

[19]:
```
var_x_train, var_x_test, var_y_train, var_y_test =␣
 ↪train_test_split(var_ict1110_2018_ca_test4["PassedTest4_X"],␣
 ↪var_ict1110_2018_ca_test4["PassedCA_Y"], test_size=0.20)
```

[20]:
```
len(var_x_train)
len(var_y_train)
```

[20]:
```
79
```

[20]:
```
79
```

4. Implement Model Using Logistic Regression

- Implement model using Logistic Regression, with Test 4 as input and CA as ouput
- Model predicts whether student passes or fails CA based on whether student passed or failed Test 4

[21]:
```
# Create instance of LogisticRegression
# from sklearn.linear_model import LogisticRegression
var_model_t4_ca_lr = LogisticRegression()
```

[22]:
```
# Train the model by feeding it training data
var_model_t4_ca_lr.fit(var_x_train.values.reshape(-1, 1), var_y_train.values.
 ↪reshape(-1, 1))
```

```
/home/lightonphiri/.local/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/home/lightonphiri/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[22]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

5. Make Predictions Using Model and Test Data

- Remember to use testing data generated

[23]:
```
var_model_t4_ca_lr.predict(var_x_test.values.reshape(-1, 1))
var_y_test.values
```

[23]:
```
array([0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 1])
```

[23]:
```
array([0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1])
```

6. Test Performance of Model

[24]:
```
# Computer model score
var_model_t4_ca_lr.score(var_x_test.values.reshape(-1, 1), var_y_test.values.
 ↪reshape(-1, 1))
```

[24]:
```
0.7
```

[25]:
```
# Illustrate results using Confusion Matrix
###from sklearn.metrics import confusion_matrix
confusion_matrix(var_model_t4_ca_lr.predict(var_x_test.values.reshape(-1, 1)),␣
 ↪var_y_test)
```

[25]:
```
array([[ 2,  5],
       [ 1, 12]])
```

Class Test #4 (Pass/Fail) vs CA Score (Pass/Fail) [Using K Fold Cross Validation]

[26]:
```
###
```

Exercise #1

1. Modify the current model so it uses Class Theory Test #4 Grades—A+, A, B+, B, C+, C, D+, D—as opposed to Pass/Fail input and compare the accuracy
2. Implement a model that uses student Minors as input features
3. Though controversial, implement a model that uses student Gender as input features
4. Implement a model that uses Test 1 and Test 2 as input features. The thinking here is that interventions can be initiated that will prevent at-risk students from failing the CA

Multi-Label Classification

Class Test #4 (Pass/Fail) vs CA Score Grade (A+, A, B+, B, C+, C, D+, D)

- Model uses Class Theory Test 4 to determine the grade—+, A, B+, B, C+, C, D+, D—that the student will get
- NOTE: This is just for illustration purposes [. . .] In an ideal scenario, one would not use this feature in isolation

[27]:
```
var_ict1110_2018_ca_grade_test4 = var_ict1110_2018_ca[["StudentID", "PassedTest4",␣
 ↪"GradeCA"]]
var_ict1110_2018_ca_grade_test4.head(2)
```

[27]:

	StudentID	PassedTest4	GradeCA
0	0786c2fc7a89596881a07df429364f9d	YES	B
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	D

1. Apply One Hot Encoding to X inputs

- Remember estimators take in numeric values
- Current X inputs are categorical

[28]:
```
# Use scikit learn to implement On Hot Encoding on X inputs
# Create instance of LabelEncoder
# Remember: You could just as well have used pandas' get dummies implementation for␣
 ↪this
var_ict1110_2018_test4_grade_encoder = LabelEncoder()
```

[29]:
```
# Fit encoder on inputs
var_ict1110_2018_test4_grade_encoder.
 ↪fit(var_ict1110_2018_ca_grade_test4["PassedTest4"])
```

[29]:
```
LabelEncoder()
```

[30]:
```
# Confirm categorical variables
var_ict1110_2018_test4_grade_encoder.classes_
```

[30]:
```
array(['NO', 'YES'], dtype=object)
```

[31]:
```
# Apply fitted Label Encoder to applicable DataFrame column
var_ict1110_2018_test4_grade_encoder.
 ↪transform(var_ict1110_2018_ca_grade_test4["PassedTest4"])
var_ict1110_2018_ca_grade_test4.head(2)
```

[31]:
```
array([1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1,
       0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1])
```

[31]:

	StudentID	PassedTest4	GradeCA
0	0786c2fc7a89596881a07df429364f9d	YES	B
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	D

[32]:

```
# Create new column to hold encoded values
var_ict1110_2018_ca_grade_test4["PassedTest4_X"] =␣
 ↪var_ict1110_2018_test4_grade_encoder.
 ↪transform(var_ict1110_2018_ca_grade_test4["PassedTest4"])
var_ict1110_2018_ca_grade_test4.head(2)
```

```
/home/lightonphiri/.local/lib/python3.6/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

[32]:

	StudentID	PassedTest4	GradeCA	PassedTest4_X
0	0786c2fc7a89596881a07df429364f9d	YES	B	1
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	D	0

2. Apply Label Encoders to Labels

- Remember estimators take in numeric values
- Current Y are categorical

[33]:

```
# Create instance of LabelEncoder
var_ict1110_2018_ca_grade_encoder = LabelEncoder()
```

[34]:

```
# Fit encoder on inputs
var_ict1110_2018_ca_grade_encoder.fit(var_ict1110_2018_ca_grade_test4["GradeCA"])
```

[34]:

```
LabelEncoder()
```

[35]:

```
# Confirm categorical variables
var_ict1110_2018_ca_grade_encoder.classes_
```

[35]:

```
array(['A', 'B', 'B+', 'C', 'C+', 'D', 'D+'], dtype=object)
```

[36]:

```
# Apply fitted Label Encoder to applicable DataFrame column
var_ict1110_2018_ca_grade_encoder.transform(var_ict1110_2018_ca_grade_test4["GradeCA"])
var_ict1110_2018_ca_grade_test4.head(2)
```

[36]:

```
array([1, 5, 3, 4, 5, 5, 1, 3, 6, 4, 1, 4, 5, 4, 4, 3, 3, 4, 5, 4, 3, 1,
       3, 4, 4, 1, 3, 4, 3, 4, 6, 5, 3, 1, 4, 5, 4, 6, 3, 4, 3, 5, 6, 5,
       5, 5, 5, 4, 4, 4, 4, 6, 6, 5, 2, 5, 0, 5, 4, 1, 6, 1, 2, 5, 5, 4,
       4, 5, 4, 5, 3, 4, 5, 4, 5, 6, 6, 4, 4, 6, 4, 5, 6, 1, 3, 4, 6, 3,
        6, 4, 5, 4, 4, 5, 4, 1, 4, 4, 6])
```

[36]:

	StudentID	PassedTest4	GradeCA	PassedTest4_X
0	0786c2fc7a89596881a07df429364f9d	YES	B	1
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	D	0

[37]:
```
# Create new column to hold encoded values
var_ict1110_2018_ca_grade_test4["GradeCA_Y"] = var_ict1110_2018_ca_grade_encoder.
 ↪transform(var_ict1110_2018_ca_grade_test4["GradeCA"])
var_ict1110_2018_ca_grade_test4.head(2)
```

```
/home/lightonphiri/.local/lib/python3.6/site-packages/ipykernel_launcher.py:2:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

[37]:

	StudentID	PassedTest4	GradeCA	PassedTest4_X	GradeCA_Y
0	0786c2fc7a89596881a07df429364f9d	YES	B	1	1
1	3283f0f7cce04d7ce5cd5dfdd6d191bf	NO	D	0	5

3. Create Train/Test Sets

- Create test and training datasets

[38]:
```
var_x_train, var_x_test, var_y_train, var_y_test =␣
 ↪train_test_split(var_ict1110_2018_ca_grade_test4["PassedTest4_X"],␣
 ↪var_ict1110_2018_ca_grade_test4["GradeCA_Y"], test_size=0.20)
```

[39]:
```
len(var_x_train)
len(var_y_train)
```

[39]:
```
79
```

[39]:
```
79
```

4. Implement Model Using Logistic Regression

- Implement model using Logistic Regression, with Test 4 as input and CA as ouput
- Model predicts whether student passes or fails CA based on whether student passed or failed Test 4

[40]:
```
# Create instance of LogisticRegression
# from sklearn.linear_model import LogisticRegression
var_model_t4_ca_grade_lr = LogisticRegression()
```

[41]:
```
# Train the model by feeding it training data
```

```
var_model_t4_ca_grade_lr.fit(var_x_train.values.reshape(-1, 1), var_y_train.values.
 ↪reshape(-1, 1))
```

```
/home/lightonphiri/.local/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:433: FutureWarning: Default solver will be
changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
/home/lightonphiri/.local/lib/python3.6/site-packages/sklearn/utils/validation.py:761:
DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please
change the shape of y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/home/lightonphiri/.local/lib/python3.6/site-
packages/sklearn/linear_model/logistic.py:460: FutureWarning: Default multi_class will be
changed to 'auto' in 0.22. Specify the multi_class option to silence this warning.
  "this warning.", FutureWarning)
```

[41]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

5. Make Predictions Using Model and Test Data

- Remember to use testing data generated

[42]:
```
var_model_t4_ca_grade_lr.predict(var_x_test.values.reshape(-1, 1))
var_y_test.values
```

[42]:
```
array([4, 4, 4, 4, 4, 5, 4, 5, 5, 5, 5, 4, 5, 5, 4, 5, 5, 5, 5, 5])
```

[42]:
```
array([6, 3, 6, 3, 1, 5, 1, 4, 5, 4, 4, 2, 5, 3, 1, 5, 4, 3, 5, 6])
```

6. Test Performance of Model

[43]:
```
# Computer model score
var_model_t4_ca_grade_lr.score(var_x_test.values.reshape(-1, 1), var_y_test.values.
 ↪reshape(-1, 1))
```

[43]:
```
0.25
```

[44]:
```
# Illustrate results using Confusion Matrix
###from sklearn.metrics import confusion_matrix
var_confusion_matrix_ca_graded = confusion_matrix(var_model_t4_ca_grade_lr.
 ↪predict(var_x_test.values.reshape(-1, 1)), var_y_test)
var_confusion_matrix_ca_graded
```

[44]:
```
array([[0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [3, 1, 2, 0, 0, 2],
       [0, 0, 2, 4, 5, 1],
       [0, 0, 0, 0, 0, 0]])
```

[51]:

```
# Plot confusion matrix with raw values
# 'A', 'B', 'B+', 'C', 'C+', 'D', 'D+'
var_confusion_matrix_ca_graded_plot = pd.DataFrame(var_confusion_matrix_ca_graded,␣
 ↪index = [i for i in ["A", "B", "B+", "C", "C+", "D"]], columns = [i for i in ["A",␣
 ↪"B", "B+", "C", "C+", "D"]])

plt.figure(figsize = (10,5))
sns.set(font_scale=0.9)
sns.heatmap(var_confusion_matrix_ca_graded, annot=True, annot_kws={"size": 12},␣
 ↪cmap='Reds', fmt='g')


# Add labels

plt.figure(figsize = (10,5))
sns.set(font_scale=0.9)
sns.heatmap(var_confusion_matrix_ca_graded_plot, annot=True, annot_kws={"size": 12},␣
 ↪cmap='Reds', fmt='g')

# Normalised plot
var_confusion_matrix_normalised = var_confusion_matrix_ca_graded.astype('float') /␣
 ↪var_confusion_matrix_ca_graded.sum(axis=1)[:, np.newaxis]

var_confusion_matrix_normalised = var_confusion_matrix_normalised.round(2)
var_confusion_matrix_normalised

var_confusion_matrix_normalised_plot = pd.DataFrame(var_confusion_matrix_normalised,␣
 ↪index = [i for i in ["A", "B", "B+", "C", "C+", "D"]], columns = [i for i in ["A",␣
 ↪"B", "B+", "C", "C+", "D"]])

plt.figure(figsize = (10,5))
sns.set(font_scale=0.9)
sns.heatmap(var_confusion_matrix_normalised_plot, annot=True, annot_kws={"size": 12},␣
 ↪cmap='Reds', fmt='g')
```

[51]:

```
<Figure size 720x360 with 0 Axes>
```

[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f944fbc2908>
```

[51]:

```
<Figure size 720x360 with 0 Axes>
```

[51]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93ff241eb8>
```

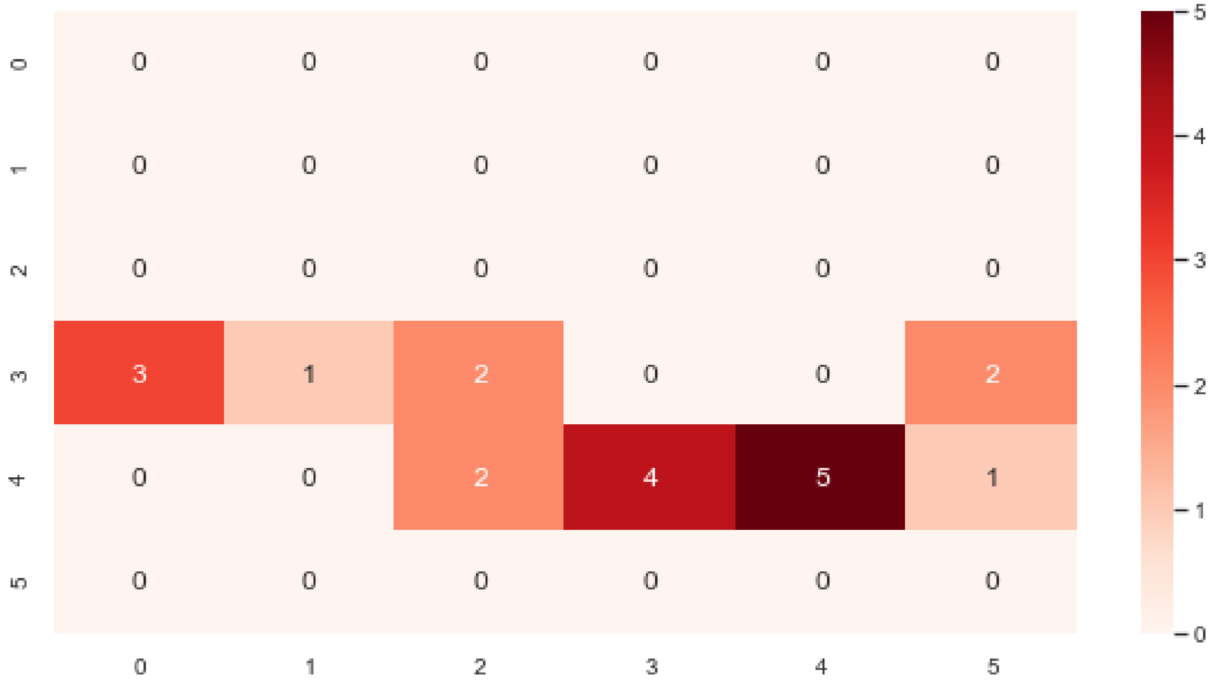
```
/home/lightonphiri/.local/lib/python3.6/site-packages/ipykernel_launcher.py:17:
RuntimeWarning: invalid value encountered in true_divide
```

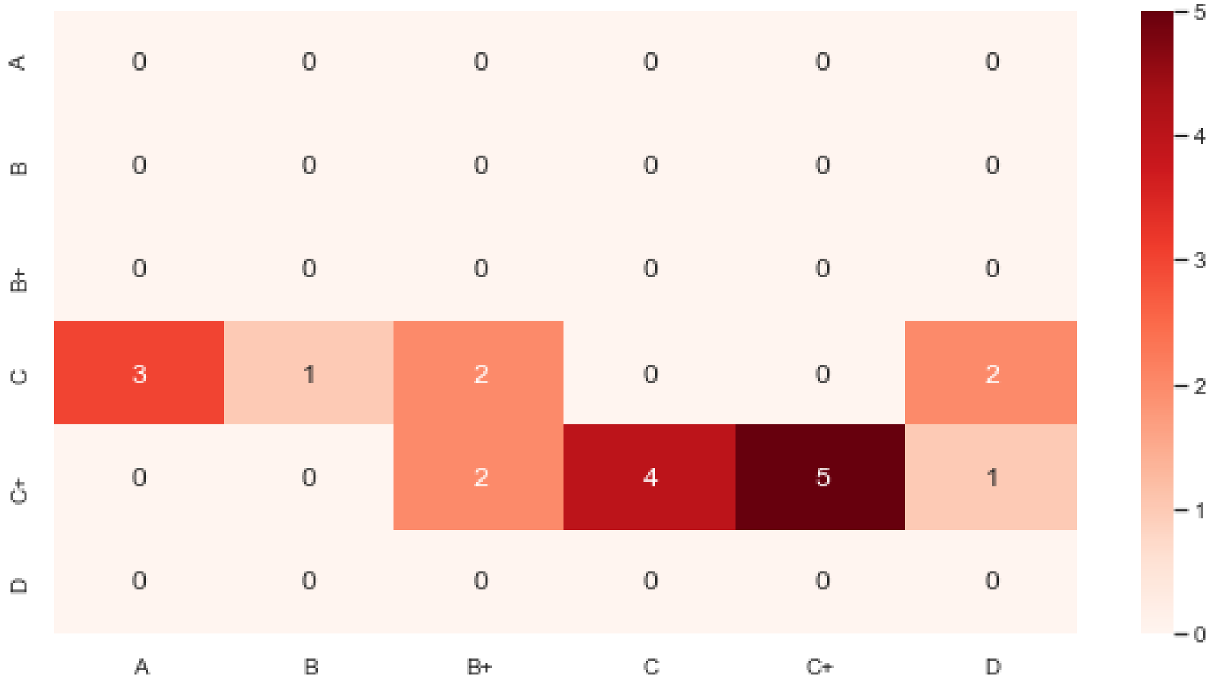
[51]:

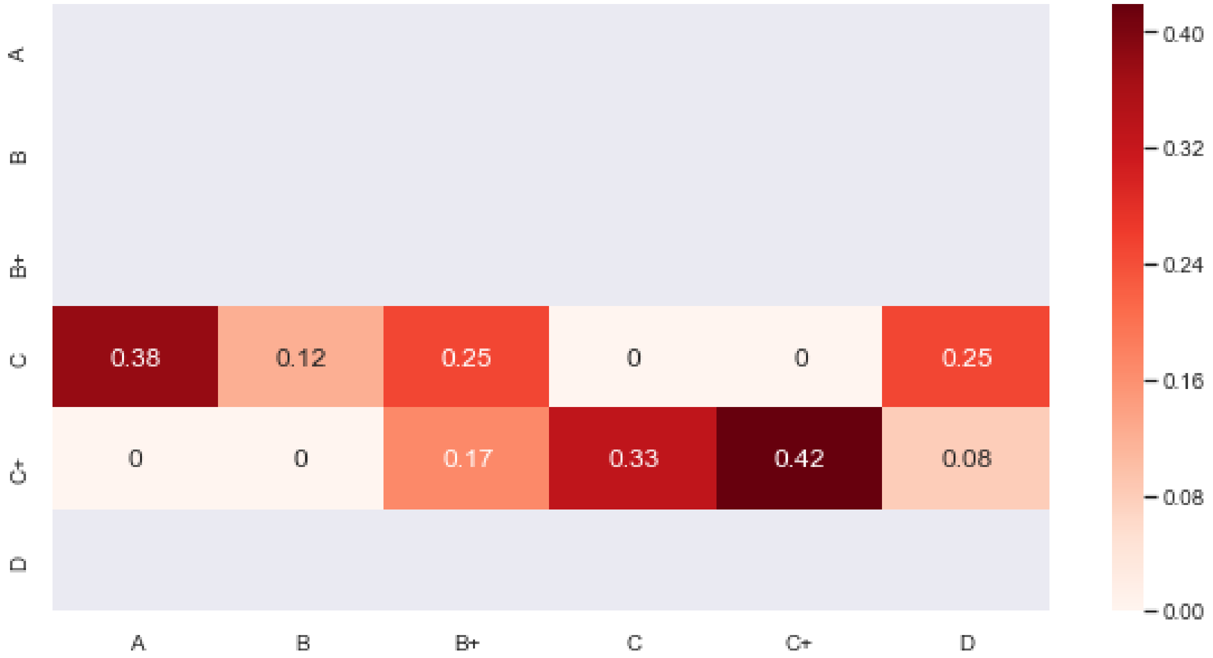
```
array([[ nan,  nan,  nan,  nan,  nan,  nan],
       [ nan,  nan,  nan,  nan,  nan,  nan],
       [ nan,  nan,  nan,  nan,  nan,  nan],
       [0.38, 0.12, 0.25, 0.  , 0.  , 0.25],
       [0.  , 0.  , 0.17, 0.33, 0.42, 0.08],
        [ nan,  nan,  nan,  nan,  nan,  nan]])
```

[51]:
```
<Figure size 720x360 with 0 Axes>
```

[51]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f93fb6a6ac8>
```







[46]:

```
# df.groupby(['Col1', 'Col2']).size().reset_index(name='Freq')
# https://stackoverflow.com/a/44906862/664424
var_ict1110_2018_ca_grade_test4.groupby(["GradeCA", "GradeCA_Y"]).size().
 ↪reset_index(name='Freq')
```

[46]:

	GradeCA	GradeCA_Y	Freq
0	A	0	1
1	B	1	10
2	B+	2	2
3	C	3	14
4	C+	4	34
5	D	5	24
6	D+	6	14

[47]:

```
var_y_test.unique()
np.unique(var_model_t4_ca_grade_lr.predict(var_x_test.values.reshape(-1, 1)))
```

[47]:

```
array([6, 3, 1, 5, 4, 2])
```

[47]:

```
array([4, 5])
```

[48]:

```
var_confusion_matrix_ca_graded
```

[48]:

```
array([[0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0],
       [3, 1, 2, 0, 0, 2],
       [0, 0, 2, 4, 5, 1],
       [0, 0, 0, 0, 0, 0]])
```

[50]:

```
var_ict1110_2018_ca_grade_encoder.classes_
```

[50]:

```
array(['A', 'B', 'B+', 'C', 'C+', 'D', 'D+'], dtype=object)
```

[ ]: