COMP4220: Machine Learning, Spring 2022, Assignment 4

**Please submit one pdf le for all questions.**

1. KMeans:

#importing the libraries --add any additional libraries you will need here import numpy as np import pandas as pd from sklearn.cluster import KMeans

data = pd.read\_csv("titanic.csv") print(data)

# removing the columns not of interest data = data.drop(['PassengerId','Name','Ticket', 'Cabin','Embarked','Pclass','SibSp','Sex','P

# removing rows of data with NaN data = data[data['Age'].notna()]

PassengerId Survived Pclass \

1. 1 0 3
2. 2 1 1
3. 3 1 3
4. 4 1 1
5. 5 0 3

.. ... ... ...

1. 887 0 2
2. 888 1 1
3. 889 0 3
4. 890 1 1
5. 891 0 3

Name Sex Age SibSp \

1. Braund, Mr. Owen Harris male 22.0 1
2. Cumings, Mrs. John Bradley (Florence Briggs Th... female 38.0 1
3. Heikkinen, Miss. Laina female 26.0 0
4. Futrelle, Mrs. Jacques Heath (Lily May Peel) female 35.0 1
5. Allen, Mr. William Henry male 35.0 0 .. ... ... ... ...
6. Montvila, Rev. Juozas male 27.0 0
7. Graham, Miss. Margaret Edith female 19.0 0
8. Johnston, Miss. Catherine Helen "Carrie" female NaN 1
9. Behr, Mr. Karl Howell male 26.0 0
10. Dooley, Mr. Patrick male 32.0 0

Parch Ticket Fare Cabin Embarked

1. 0 A/5 21171 7.2500 NaN S
2. 0 PC 17599 71.2833 C85 C
3. 0 STON/O2. 3101282 7.9250 NaN S
4. 0 113803 53.1000 C123 S
5. 0 373450 8.0500 NaN S .. ... ... ... ... ...
6. 0 211536 13.0000 NaN S
7. 0 112053 30.0000 B42 S
8. 2 W./C. 6607 23.4500 NaN S
9. 0 111369 30.0000 C148 C
10. 0 370376 7.7500 NaN Q

[891 rows x 12 columns]

a) De ne X and y from the training data. Answer provided. Print X

and y to see data.

X = data.drop(['Survived'], 1).astype(float) y = data['Survived'] print(X) print(y)

Age

1. 22.0
2. 38.0
3. 26.0
4. 35.0
5. 35.0 .. ...
6. 39.0
7. 27.0
8. 19.0
9. 26.0
10. 32.0

[714 rows x 1 columns]

1. 0
2. 1
3. 1
4. 1
5. 0 ..
6. 0
7. 0
8. 1
9. 1
10. 0

Name: Survived, Length: 714, dtype: int64

/usr/local/lib/python3.7/dist-packages/ipykernel\_launcher.py:1: FutureWarning: In a fut

"""Entry point for launching an IPython kernel.

1. Perform KMeans on X

import sklearn.model\_selection as model

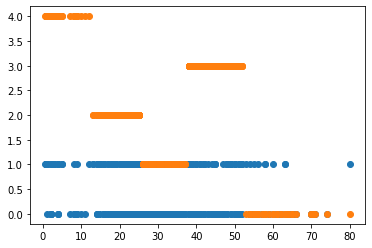
X\_train, X\_test, y\_train, y\_test = model.train\_test\_split(X, y, test\_size=0.33, random\_state= k = 5

kmeans = KMeans(n\_clusters=k, random\_state=69).fit(X\_train) y\_predict = kmeans.predict(X\_train)

1. Plot the prediction for X

import matplotlib.pyplot as plt

centers = kmeans.cluster\_centers\_ plt.scatter(X\_train, y\_train) plt.scatter(X\_train, y\_predict) plt.show()



1. Compute the accuracy

from sklearn.metrics import accuracy\_score accuracy\_score(y\_train, y\_predict)

0.18410041841004185

2

. Classi cation using SVM

This is data collected from brain waves collection during a pain detection research project.

import numpy as np from sklearn.pipeline import Pipeline from sklearn.preprocessing import StandardScaler from sklearn.svm import LinearSVC

painData = pd.read\_csv("pain.csv")

painData = painData.drop(['SubjectID','Index','Date', 'Time'], axis=1) painData

**Right**

**PainType TP9 AF7 AF8 TP10 label**

**Axis**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | severe pain | 68.847656 | -73.242188 | 18.066406 | 27.832031 | 25.390625 | 3 |
| **1** | severe pain | 44.921875 | -235.351562 | 36.621094 | 27.832031 | -4.394531 | 3 |
| **2** | severe pain | -11.230469 | -81.054688 | 45.410156 | 29.296875 | 12.207031 | 3 |
| **3** | severe pain | -2.929688 | 17.089844 | 33.203125 | 24.902344 | 44.433594 | 3 |
| **4** | severe pain | 10.253906 | -58.105469 | 32.226562 | 14.648438 | -0.976562 | 3 |
| **...** | ... | ... | ... | ... | ... | ... | ... |
| **60191** | moderate pain | 33.203125 | 287.597656 | 45.898438 | 27.832031 | 25.878906 | 2 |
| **60192** | moderate pain | 24.414062 | -20.507812 | 32.226562 | 21.484375 | 34.179688 | 2 |
| **60193** | moderate pain | 28.808594 | -270.019531 | 24.902344 | 24.902344 | 34.667969 | 2 |
| **60194** | moderate pain | 37.109375 | -190.917969 | 30.761719 | 31.250000 | -36.132812 | 2 |

The label column is the target, and pain type is an explanation.

a) Get X and y from painData above. X is TP9 and Right Axis. Y is

label.

X = painData[['TP9', 'Right Axis']] y = painData['label']

1. Using a regularization parameter of c=1 and c=100, using a

LinearSVC.

scaler = StandardScaler()

svm\_cfm1 = LinearSVC(C=1, loss="hinge", random\_state=42) svm\_cfm100 = LinearSVC(C=100, loss="hinge", random\_state=42)

1. Scale the dataset using a pipeline

scaled\_svm\_cfm1 = Pipeline([

("scaler", scaler),

("linear\_svc", svm\_cfm1),

])

scaled\_svm\_cfm100 = Pipeline([

("scaler", scaler),

("linear\_svc", svm\_cfm100),

])

scaled\_svm\_cfm1.fit(X,y) scaled\_svm\_cfm100.fit(X,y)

Pipeline(steps=[('scaler', StandardScaler()),

('linear\_svc',

LinearSVC(C=100, loss='hinge', random\_state=42))])

c) Plot dataset using the regularization parameter of c=1 and c=100

3. Decision Trees:

Using the same dataset above, meaning X and y

1. Print the shape of X and y

print(X.shape) print(y.shape) (60196, 2) (60196,)

1. Train using a decision tree classi er

from sklearn.tree import DecisionTreeClassifier

tree\_clf = DecisionTreeClassifier(max\_depth=2, random\_state=42) tree\_clf.fit(X,y)

DecisionTreeClassifier(max\_depth=2, random\_state=42)

1. Visualize the dataset

import os

project\_root\_dir = "." chapter\_id = "decision\_trees" images\_path = os.path.join(project\_root\_dir, "images", chapter\_id) os.makedirs(images\_path,exist\_ok=True) from graphviz import Source from sklearn.tree import export\_graphviz

export\_graphviz( tree\_clf, out\_file=os.path.join(images\_path,"pain\_tree.dot"), feature\_names = ['TP9', 'Right Axis'], class\_names = 'label', rounded = True, filled = True

)

Source.from\_file(os.path.join(images\_path, "pain\_tree.dot"))

value

Diagram, schematic

Description automatically generated

TP9 <= 5.127

gini = 0.796

samples = 5240

value = [11416, 11516, 10596

class = a

T

gini = 0.748

samples = 13402

value = [1239, 1130, 2678, 3712, 4643]

class = l

gini = 0.776

samples = 390

value = [10177, 10386, 791

class = a

d) Plot the decision boundaries of the dataset

Run on pain.csv

4

. Ensemble Classi er and Random forest

a) Run a voting classi er that includes logistic regression, random

forest classi er and SVM

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y, random\_state = 69)

from sklearn.ensemble import RandomForestClassifier from sklearn.ensemble import VotingClassifier from sklearn.linear\_model import LogisticRegression from sklearn.svm import SVC

log\_clf = LogisticRegression(solver="lbfgs", random\_state=42) rnd\_clf = RandomForestClassifier(n\_estimators=100, random\_state=42) svm\_clf = SVC(gamma="scale", random\_state=42)

voting\_clf = VotingClassifier( estimators=[('lr', log\_clf), ('rf', rnd\_clf), ('svc', svm\_clf)], voting='hard')

b) Print the accuracy scores

from sklearn metrics import accuracy score

from sklearn.metrics import accuracy\_score

for clf in (log\_clf,rnd\_clf,svm\_clf,voting\_clf):

clf.fit(X\_train,y\_train) y\_pred = clf.predict(X\_test)

print(clf.\_\_class\_\_.\_\_name\_\_, accuracy\_score(y\_test, y\_pred))

LogisticRegression 0.2624094624227523

RandomForestClassifier 0.32414113894610935

SVC 0.39650475114625555

VotingClassifier 0.3594258754734534

**

0

s

completed at 8:20 PM