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  \
0              1         0       3
1              2         1       1
2              3         1       3
3              4         1       1
4              5         0       3
..           ...       ...     ...
886          887         0       2
887          888         1       1
888          889         0       3
889          890         1       1
890          891         0       3

                                                  Name     Sex   Age  SibSp  \
0                                  Braund, Mr. Owen Harris    male  22.0      1
1                                  Cumings, Mrs. John Bradley (Florence Briggs Th...
                                   female  38.0      1
2                                  Heikkinen, Miss. Laina  female  26.0      0
3                                  Futrelle, Mrs. Jacques Heath (Lily May Peel)  female
                                   35.0      1
4                                  Allen, Mr. William Henry    male  35.0      0    ..
                                   ...      ...    ...     ...
886                                   Montvila, Rev. Juozas    male  27.0      0
887                                   Graham, Miss. Margaret Edith  female  19.0      0
888                                   Johnston, Miss. Catherine Helen "Carrie"  female
                                      NaN      1
889                                   Behr, Mr. Karl Howell    male  26.0      0
```

```
890                               Dooley, Mr. Patrick    male  32.0      0
     Parch            Ticket     Fare Cabin Embarked
0        0         A/5 21171   7.2500   NaN        S
1        0          PC 17599  71.2833   C85        C
2        0  STON/O2. 3101282   7.9250   NaN        S
3        0            113803  53.1000  C123        S
4        0            373450   8.0500   NaN        S  ..    ...               ...
       ...   ...       ...
886      0            211536  13.0000   NaN        S
887      0            112053  30.0000   B42        S
888      2        W./C. 6607  23.4500   NaN        S
889      0            111369  30.0000  C148        C
890      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
0    22.0
1    38.0
2    26.0
3    35.0
4    35.0 ..    ...
885  39.0
886  27.0
887  19.0
889  26.0
890  32.0

[714 rows x 1 columns]
0      0
1      1
2      1
3      1
4      0      ..
885    0
886    0
887    1
889    1
890    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.
```

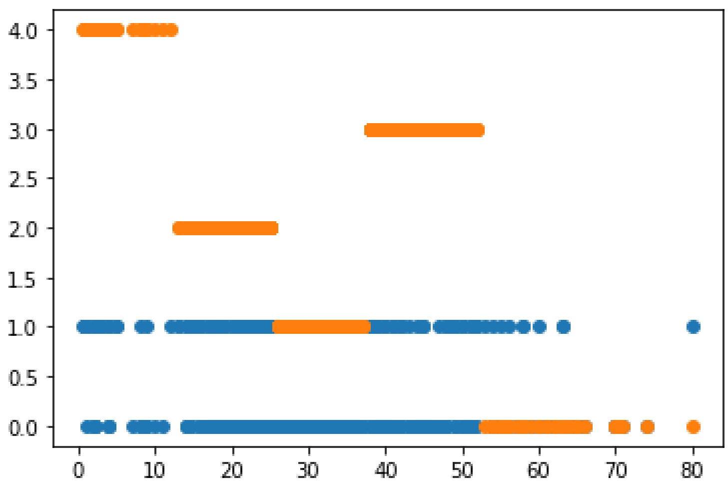
b) Perform KMeans on X

```
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)
```

c) 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()
```



d) 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
```

	PainType	TP9	AF7	AF8	TP10	Right Axis	label
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']
```

a) 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)
```

b) 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

a) Print the shape of X and y

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

b) Train using a decision tree classi er

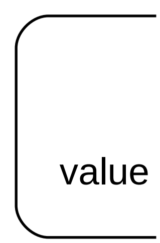
```
from sklearn.tree import DecisionTreeClassifier
```

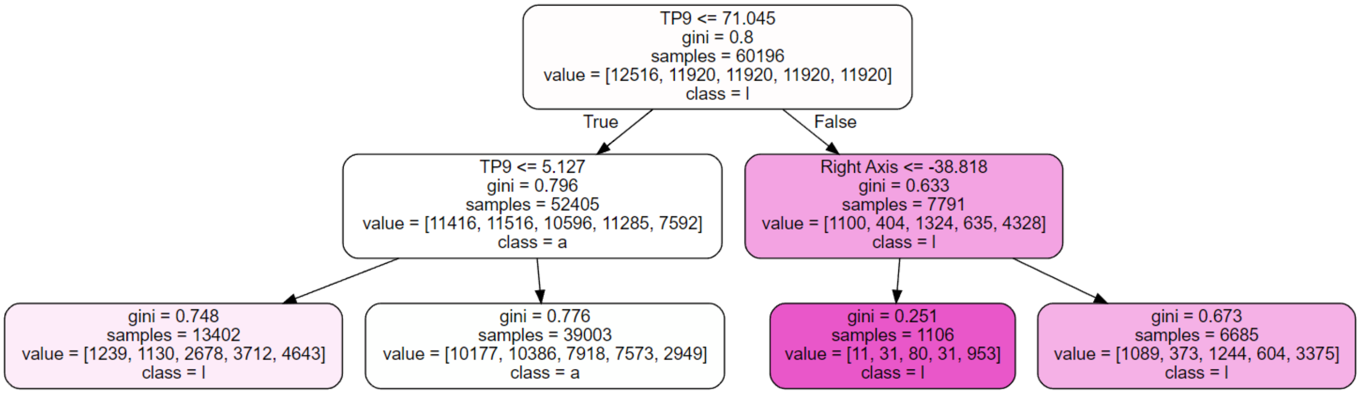
```
tree_clf = DecisionTreeClassifier(max_depth=2, random_state=42)
tree_clf.fit(X,y)
```

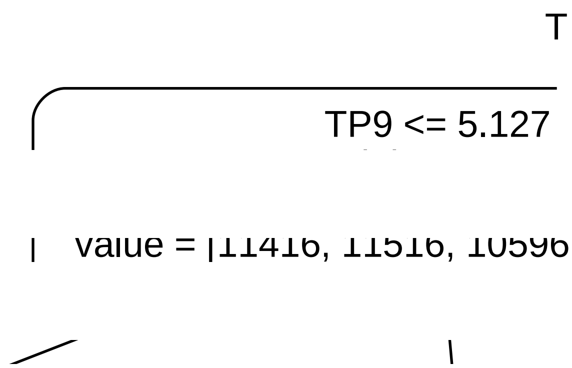
```
    DecisionTreeClassifier(max_depth=2, random_state=42)
```

c) 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"))
```







d) Plot the decision boundaries of the dataset

4. Ensemble Classi er and Random forest

Run on pain.csv

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
```

0s completed at 8:20 PM