COMP4220: Machine Learning, Spring 2022, Assignment 4

Please submit one pdf file 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
import matplotlib.pyplot as plt

X_train = pd.read_csv("titanic.csv")

# removing the columns not of interest
X_train = X_train.drop(['PassengerId','Name','Ticket',
'Cabin','Embarked','Pclass','SibSp','Sex','Parch','Fare'], axis=1)

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

X_train.isnull().values.any()
X_train.fillna(0, inplace=True)
```

a) Define X and y from the training data. Answer provided. Print X and y to see data.

```
X = X_train.drop(['Survived'], 1).astype(float)
y = X_train['Survived']

print(X.shape)

(714, 1)

print(y.shape)

(714,)
```

b) Perform KMeans on X

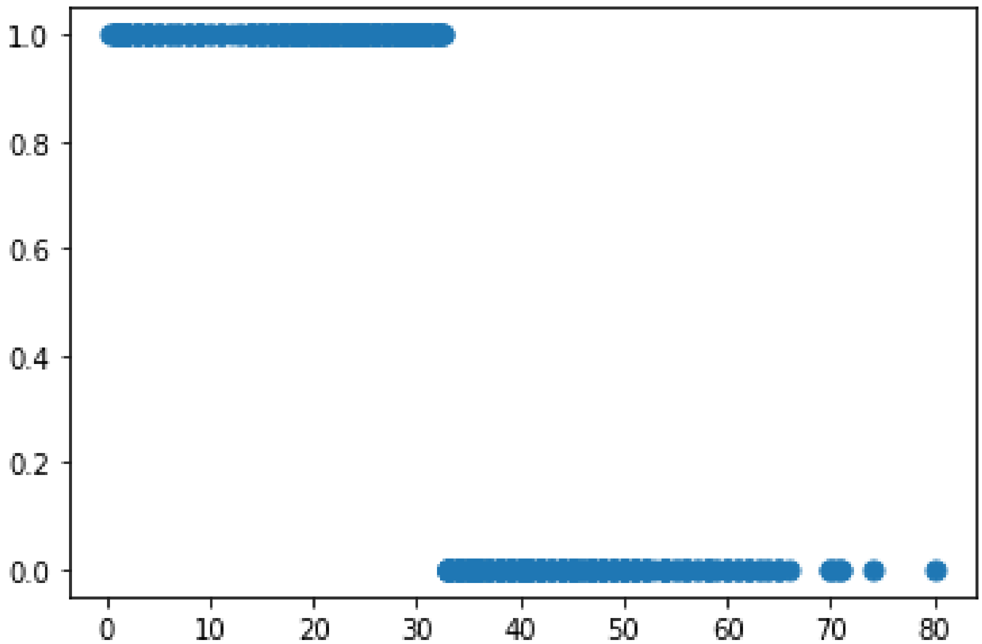
```
k = 2
kmeans = KMeans(n_clusters=k, random_state=42)
y_kmeans = kmeans.fit_predict(X)
centroids = kmeans.cluster_centers_
print(centroids)

[[44.60150376]
 [20.85082589]]
```

c) Plot the prediction for X

```
centers = kmeans.cluster_centers_
plt.scatter(X, y_kmeans, s=50, cmap='viridis')
```

```
<matplotlib.collections.PathCollection at 0x7f157c1e7e50>
```



d) Compute the accuracy

```
correct = 0

prediction = kmeans.predict(X)

pred_df = pd.DataFrame({'actual': y, 'prediction': prediction})
print(pred_df)
```

```
     actual  prediction
0         0           1
1         1           0
2         1           1
3         1           0
4         0           0
..      ...         ...
885       0           0
886       0           1
887       1           1
889       1           1
890       0           1

[714 rows x 2 columns]
```

2. Classification 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  \
0        severe pain  68.847656  -73.242188  18.066406  27.832031   
25.390625   
1        severe pain  44.921875 -235.351562  36.621094  27.832031   -
4.394531   
2        severe pain -11.230469  -81.054688  45.410156  29.296875   
12.207031   
3        severe pain  -2.929688   17.089844  33.203125  24.902344   
44.433594   
4        severe pain  10.253906  -58.105469  32.226562  14.648438   -
0.976562   
...              ...        ...         ...        ...        ...   
...   
19148  moderate pain   0.000000 -358.886719  35.644531   6.347656   
19.042969   
19149  moderate pain  17.089844  -14.648438  39.550781  40.039062   
38.085938   
19150  moderate pain  14.648438  -72.265625  27.343750  38.085938   
71.777344   
19151  moderate pain  -0.488281 -693.847656  20.507812  22.949219   
56.152344   
19152  moderate pain  -0.976562 -813.964844  27.343750  26.855469   -
2.900000   

       label  
0        3.0  
1        3.0  
2        3.0  
3        3.0  
4        3.0  
...      ...  
19148    2.0  
19149    2.0  
19150    2.0  
```

```
19151    2.0
19152    NaN

[19153 rows x 7 columns]
```

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.

```
painData = painData.dropna(how='any',axis=0)
X = painData[['TP9', 'Right Axis']]
#y = painData[painData['label'].notna()]
y = painData[['label']]

print(X.shape, y.shape)

(19152, 2) (19152, 1)

painData['y'].isnull().values.any()
```

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

```
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC

scaler = StandardScaler()
svm_clf1 = LinearSVC(C=1, loss="hinge", random_state=42)
svm_clf2 = LinearSVC(C=100, loss="hinge", random_state=42)
```

b) Scale the dataset using a pipeline

```
scaled_svm_clf1 = Pipeline([
                  ("scaler", scaler),
                  ("linear_svc", svm_clf1),
])

scaled_svm_clf2 = Pipeline([
                  ("scaler", scaler),
                  ("linear_svc", svm_clf2),
])
```

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

```
scaled_svm_clf1.fit(X, y)
scaled_svm_clf2.fit(X, y)

Pipeline(steps=[('scaler', StandardScaler()),
                ('linear_svc',
                 LinearSVC(C=100, loss='hinge', random_state=42))])
```

3. Decision Trees:

Using the same dataset above, meaning X and y

a) Print the shape of X and y

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

```
(19152, 2) (19152, 1)
```

b) Train using a decision tree classifier

```
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)
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, "painData.dot"),
    feature_names=None,
    class_names=None,
    rounded=True,
    filled=True
)
```

d) Plot the decision boundaries of the dataset

```
#skipped
```

4. Ensemble Classifier and Random forest

Run on pain.csv

a) Run a voting classifier that includes logistic regression, random forest classifier 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=42)
```

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import VotingClassifier
```

```python
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

```python
from sklearn.metrics import accuracy_score

import warnings
warnings.filterwarnings('ignore')

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 0.39807852965747703
RandomForestClassifier 0.6038011695906432
SVC 0.6203007518796992
VotingClassifier 0.5925229741019215
```