**CS599 (Deep Learning)**

**Homework – 3**

1. **Python Code:**

import pandas as pd

from sklearn.model\_selection import KFold, GridSearchCV

import matplotlib

matplotlib.use("agg")

#preapring zip data for binary classification

zip\_df = pd.read\_csv("zip.test.gz", sep = " ", header = None)

zip\_label\_col\_num = 0

zip\_label\_vec = zip\_df.iloc[:, zip\_label\_col\_num]

is\_01 = zip\_label\_vec.isin([0,1])

zip\_01\_df = zip\_df.loc[is\_01, :]

is\_label\_col = zip\_01\_df.columns == zip\_label\_col\_num

zip\_features = zip\_01\_df.iloc[:, ~is\_label\_col]

zip\_labels = zip\_01\_df.iloc[:, is\_label\_col]

#preparing spam data for binary classification

spam\_df = pd.read\_csv("spam.data", sep= " ", header = None)

spam\_label\_col\_num = -1

spam\_label\_vec = spam\_df.iloc[:, spam\_label\_col\_num]

spam\_is\_01 = spam\_label\_vec.isin([0,1])

spam\_01\_df = spam\_df.loc[spam\_is\_01, :]

spam\_features = spam\_df.iloc[:, :spam\_label\_col\_num]

spam\_labels = spam\_df.iloc[:, spam\_label\_col\_num]

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

from sklearn.linear\_model import LogisticRegressionCV

from sklearn.pipeline import make\_pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import accuracy\_score

data\_dict = {

"zip" : (zip\_features, zip\_labels),

"spam" : (spam\_features, spam\_labels)

}

accuracy\_data\_frames = []

for data\_name, (data\_features, data\_labels) in data\_dict.items():

kf = KFold(n\_splits = 3, shuffle = True, random\_state = 3)

enum\_obj = enumerate(kf.split(data\_features))

for fold\_num, (train\_index, test\_index) in enum\_obj:

X\_train, X\_test = np.array(data\_features.iloc[train\_index]), np.array(data\_features.iloc[test\_index])

y\_train, y\_test = np.ravel(data\_labels.iloc[train\_index]), np.ravel(data\_labels.iloc[test\_index])

#K-nearest neighbors

knn = KNeighborsClassifier()

hp\_parameters = {"n\_neighbors": list(range(1,21))}

grid = GridSearchCV(knn, hp\_parameters, cv = 5)

grid.fit(X\_train, y\_train)

best\_n\_neighbors = grid.best\_params\_['n\_neighbors']

print("Best N-Neighbors = ", best\_n\_neighbors)

knn = KNeighborsClassifier(n\_neighbors = best\_n\_neighbors)

knn.fit(X\_train, y\_train)

knn\_pred = knn.predict(X\_test)

#Logistic Regression

pipe = make\_pipeline(StandardScaler(), LogisticRegressionCV(cv=5, max\_iter=2000))

pipe.fit(X\_train, y\_train)

lr\_pred = pipe.predict(X\_test)

y\_train\_series = pd.Series(y\_train)

most\_frequent\_class = y\_train\_series.value\_counts().idxmax()

print("Most Frequent Class = ", most\_frequent\_class)

#create a featureless baseline

featureless\_pred = np.repeat(most\_frequent\_class, len(y\_test))

#store predict data in dict

pred\_dict = {'nearest neighbors': knn\_pred,

'linear\_model': lr\_pred,

'featureless': featureless\_pred}

test\_accuracy = {}

for algorithm, predictions in pred\_dict.items():

accuracy = accuracy\_score(y\_test, predictions)

test\_accuracy[algorithm] = accuracy

for algorithm, accuracy in test\_accuracy.items():

print(f"{algorithm} Test Accuracy: {accuracy \* 100}")

accuracy\_df = pd.DataFrame({

"data\_set": [data\_name],

"fold\_id": [fold\_num],

"algorithm": [algorithm],

"accuracy": [test\_accuracy[algorithm]]})

accuracy\_data\_frames.append(accuracy\_df)

print(f"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of {data\_name}({fold\_num})\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

total\_accuracy\_df = pd.concat(accuracy\_data\_frames, ignore\_index = True)

print(total\_accuracy\_df)

import plotnine as p9

gg = p9.ggplot(total\_accuracy\_df, p9.aes(x ='accuracy', y = 'algorithm', fill = 'data\_set'))+\

p9.facet\_grid('.~data\_set') + p9.geom\_point()

gg.save("Output.png")

1. **Output:**

**>>> for data\_name, (data\_features, data\_labels) in data\_dict.items():**

**... kf = KFold(n\_splits = 3, shuffle = True, random\_state = 3)**

**... …**

**... print(f"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of {data\_name}({fold\_num})\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")**

Best N-Neighbors = 1

Most Frequent Class = 0

nearest neighbors Test Accuracy: 100.0

linear\_model Test Accuracy: 99.51923076923077

featureless Test Accuracy: 58.65384615384615

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of zip(0)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Best N-Neighbors = 1

Most Frequent Class = 0

nearest neighbors Test Accuracy: 99.51923076923077

linear\_model Test Accuracy: 99.03846153846155

featureless Test Accuracy: 57.21153846153846

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of zip(1)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Best N-Neighbors = 3

Most Frequent Class = 0

nearest neighbors Test Accuracy: 99.03381642512076

linear\_model Test Accuracy: 99.03381642512076

featureless Test Accuracy: 57.00483091787439

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of zip(2)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Best N-Neighbors = 3

Most Frequent Class = 0

nearest neighbors Test Accuracy: 79.85658409387223

linear\_model Test Accuracy: 91.39504563233378

featureless Test Accuracy: 60.88657105606258

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of spam(0)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Best N-Neighbors = 5

Most Frequent Class = 0

nearest neighbors Test Accuracy: 77.90091264667535

linear\_model Test Accuracy: 92.63363754889178

featureless Test Accuracy: 60.104302477183836

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of spam(1)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Best N-Neighbors = 1

Most Frequent Class = 0

nearest neighbors Test Accuracy: 81.99608610567515

linear\_model Test Accuracy: 92.8897586431833

featureless Test Accuracy: 60.79582517938682

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*End of spam(2)\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**>>> total\_accuracy\_df = pd.concat(accuracy\_data\_frames, ignore\_index = True)**

**>>> print(total\_accuracy\_df)**

data\_set fold\_id algorithm accuracy

0 zip 0 nearest neighbors 1.000000

1 zip 0 linear\_model 0.995192

2 zip 0 featureless 0.586538

3 zip 1 nearest neighbors 0.995192

4 zip 1 linear\_model 0.990385

5 zip 1 featureless 0.572115

6 zip 2 nearest neighbors 0.990338

7 zip 2 linear\_model 0.990338

8 zip 2 featureless 0.570048

9 spam 0 nearest neighbors 0.798566

10 spam 0 linear\_model 0.913950

11 spam 0 featureless 0.608866

12 spam 1 nearest neighbors 0.779009

13 spam 1 linear\_model 0.926336

14 spam 1 featureless 0.601043

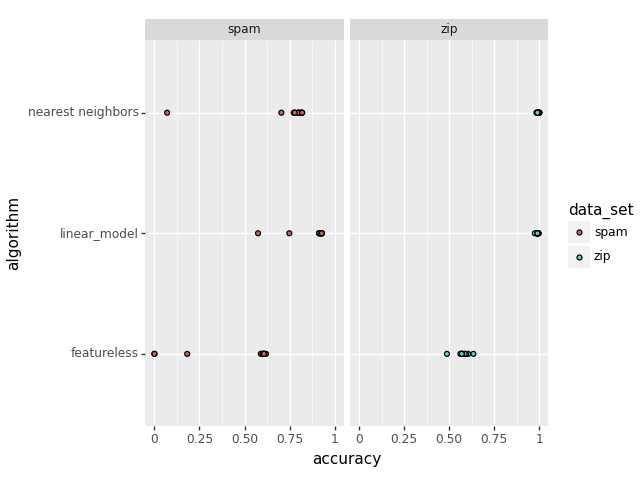
15 spam 2 nearest neighbors 0.819961

16 spam 2 linear\_model 0.928898

17 spam 2 featureless 0.607958

**>>> gg = p9.ggplot(total\_accuracy\_df, p9.aes(x ='accuracy', y = 'algorithm', fill = 'data\_set'))+\**

**... p9.facet\_grid('.~data\_set') + p9.geom\_point()**



1. **Summary:**

* First, we need to prepare the data such that it contains 0’s and 1’s in any of the labels, so that we can perform binary classification.
* To do that, we need to remove all non-01 labels from the both datasets.
* Need to create a data dictionary and run a loop over it.
* Use sklearn package to perform KFold validation, GridSearch, KNeighborsClassifier, LogisticRegression.
* Create a prediction dictionary and print all the 3 prediction accuracy. (Nearest Neighbors, Linear Model & Featureless)
* Make a ggplot using geom\_point().