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")
```

2. 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)*****************************
```

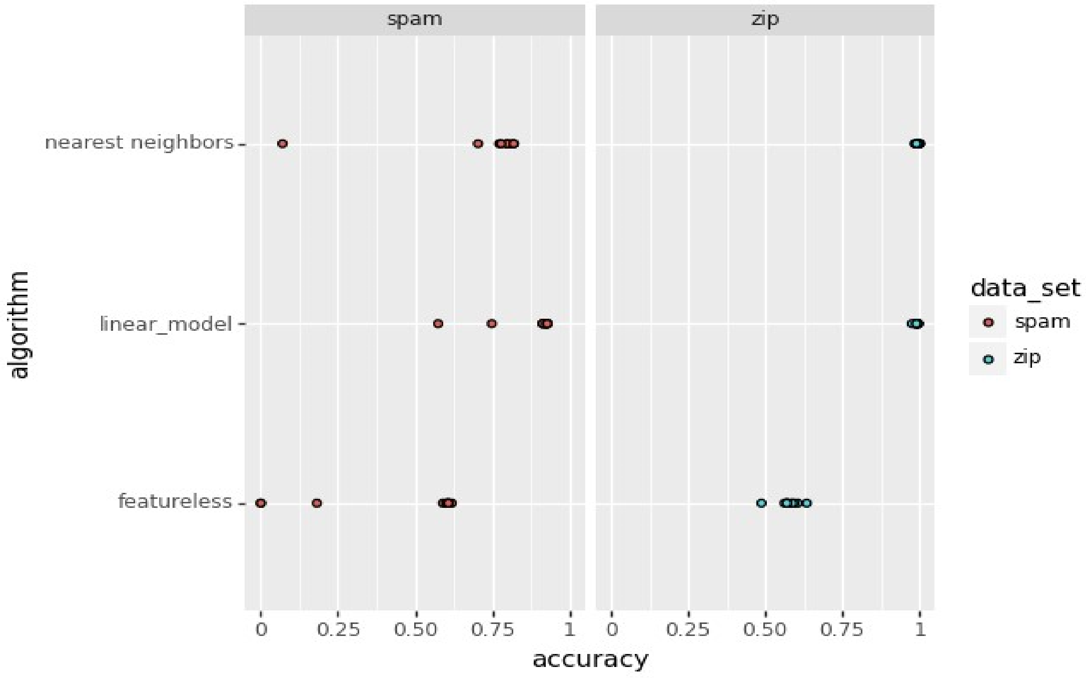
```
>>> total_accuracy_df = pd.concat(accuracy_data_frames, ignore_index = True)

>>> 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()
```



3. 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().