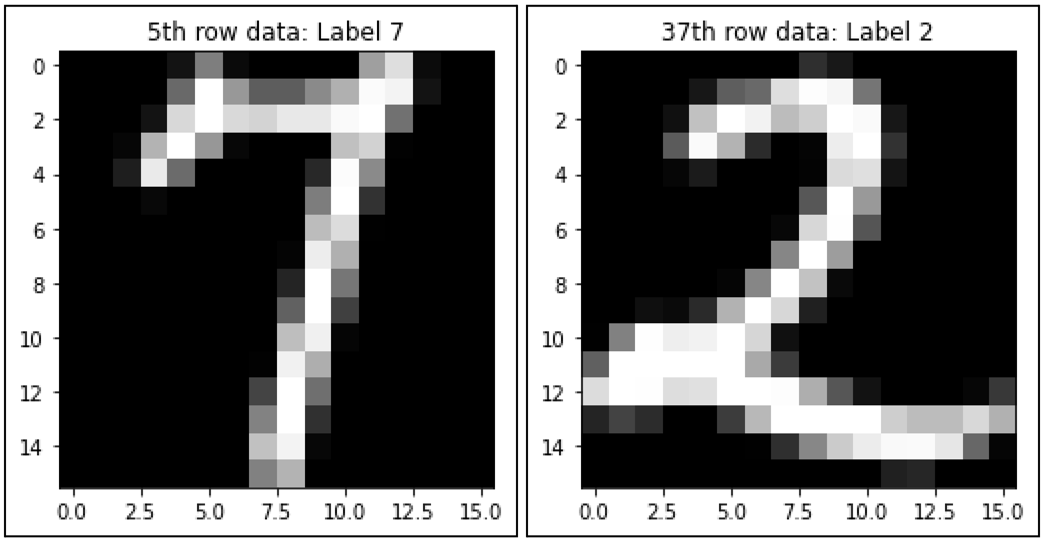
ISYE 7406 Homework #1

Fall 2022

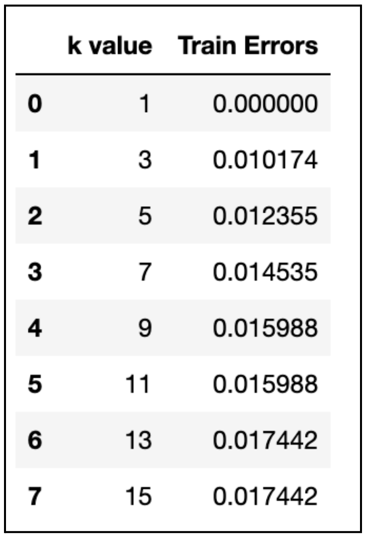
For this homework problem, we are asked to evaluate a dataset composed of digits from zip codes. Specifically, we will be classifying between the 2's and 7's within "zip.train.csv" and "zip.test.csv". We will also be evaluating model performance on this dataset comparing train and test error in prediction for Linear Regression and K-Nearest Neighbors. All code is written in Python and will be shown in the appendix at the end of the report.

(1) **Exploratory Data Analysis of Training Data**: After importing "zip.train.csv" and filtering to only rows with label 2 or 7, we see that its dimensions are 1376 rows x 257 columns. The first column is the label(Y) and the following 256 columns represent the pixels which can be reshaped to a 16x16 image.
We can also see that in the dataset of 2's and 7's, there are 731 rows with label "2" and 645 rows with label "7". Their class percentages are 53% and 47% respectively, suggesting that this dataset is comparably balanced, and does not need any manual sampling required. Additionally, the 256 columns are all at the same scale or -1 to +1 so no normalization is needed either. With that, we can try reshaping the 256 columns into 16x16 images for visualization:

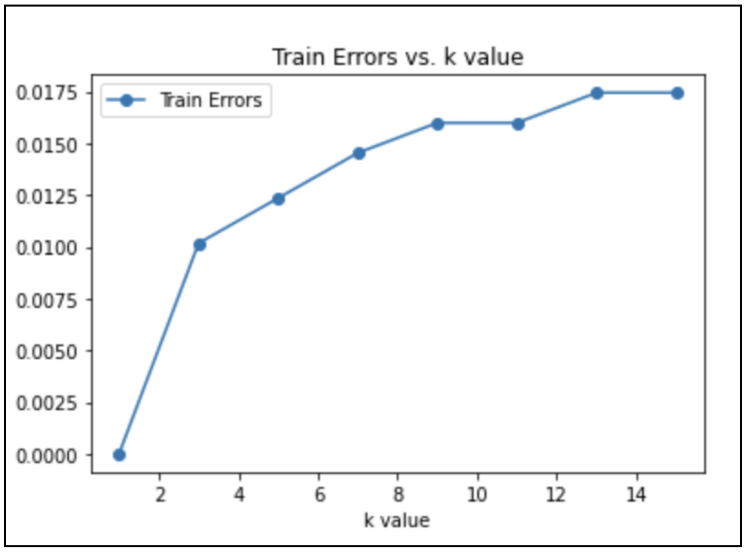


We can evaluate this dataset with classification models and comparing performance.

(2) Build the Classification Rule: First, we will consider a Linear Regression model that will be trained on the 1376 rows. Because the label is between 2 and 7, we will need to map the logit(or output) of the model to either 2 or 7 based on whether the logit is higher or lower than 4.5(the median of 2 and 7). After training, the train error is **0.0007267**. Next we will compare this with K-Nearest Neighbor models of k-value 1,3,5,7,9,11,13, and 15 and the train errors are shown below:

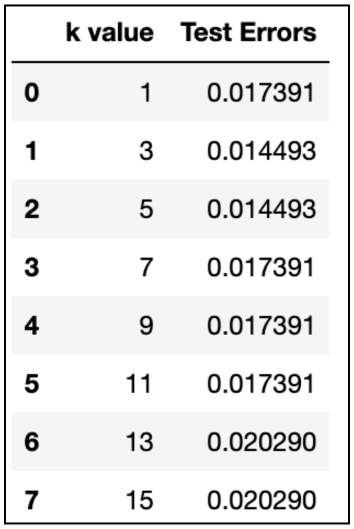


	k value	Train Errors
0	1	0.000000
1	3	0.010174
2	5	0.012355
3	7	0.014535
4	9	0.015988
5	11	0.015988
6	13	0.017442
7	15	0.017442

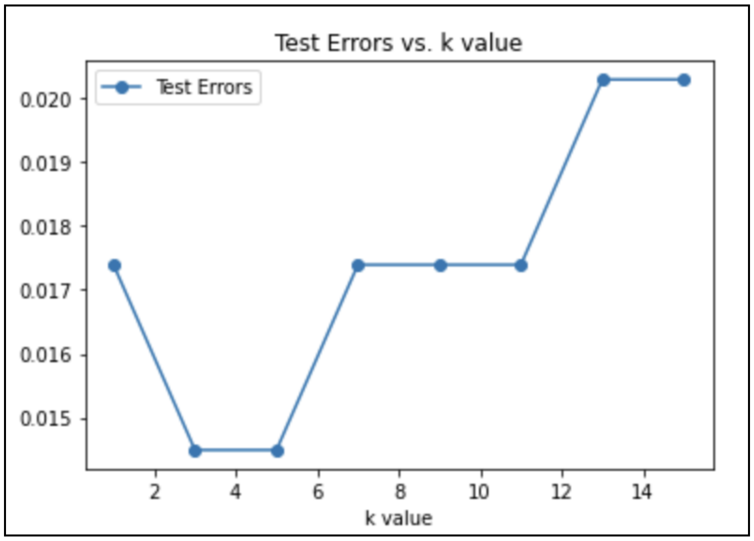


A k-value of 1 has the lowest train error of 0.00. However, we cannot assume that this k-value provides the best performance since we are predicting on the same dataset.

(3) **Testing Errors**: After training our models, we want to evaluate on a separate dataset denoted as the test dataset. After reading in "zip.test.csv", we see there are 345 rows that have label 2 or 7. We will use this dataset to compare model performance. For our linear regression model, the test error is **0.017391**. This value is higher than the train error, which is expected as the model is predicting on the data it has not seen before. As for our K-Nearest Neighbor model, the results are shown below:

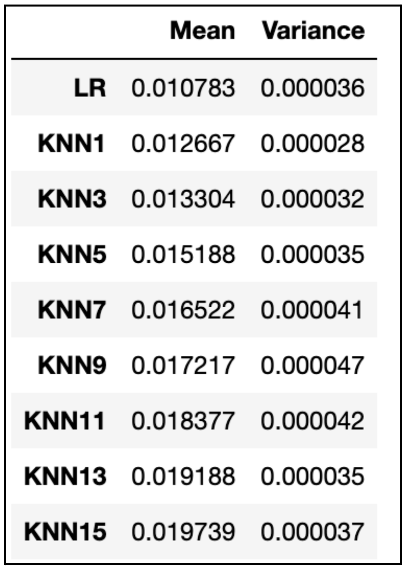


	k value	Test Errors
0	1	0.017391
1	3	0.014493
2	5	0.014493
3	7	0.017391
4	9	0.017391
5	11	0.017391
6	13	0.020290
7	15	0.020290

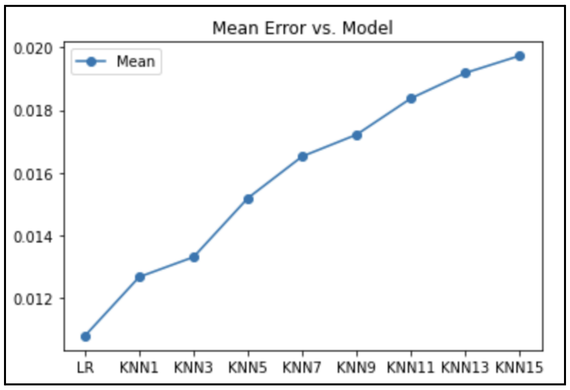


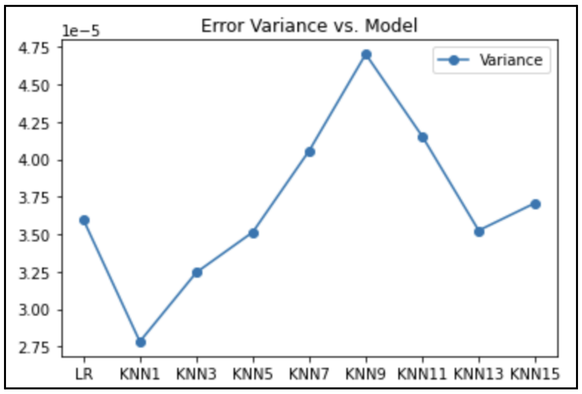
Based on the testing errors, the KNN model has the lowest training error with a k-value of either 3 or 5. Since this evaluation is made on a test dataset rather than a train dataset, it is better to with k=3 or k=5 for better performance.

(4) Cross-Validation: We will now consider Monte Carlo Cross Validation in evaluating our model performance. This approach will consist of randomly splitting our dataset into 1376 train rows and 345 test rows and training our linear regression and KNN models to predict on the respective splits. We will repeat this 100 times and summarize the average sample mean and variance for testing error. The results are shown below:



	Mean	Variance
LR	0.010783	0.000036
KNN1	0.012667	0.000028
KNN3	0.013304	0.000032
KNN5	0.015188	0.000035
KNN7	0.016522	0.000041
KNN9	0.017217	0.000047
KNN11	0.018377	0.000042
KNN13	0.019188	0.000035
KNN15	0.019739	0.000037





After our cross validation, we see that the reported variances are very low compared to the mean errors and our linear regression model has the lowest average testing error of **0.010783**. As for the KNN model, the test error has the lowest error mean of **0.012667** and variance of **0.000028** at k-value of 1. This suggests that the linear regression model outperforms the KNN model at all k-values with a lower testing error. In real-word applications, we would need to tune the parameters within each model such as classification threshold level for linear regression and distance metric for KNN to better conclude which model would have better performance on the dataset. As for KNN, typically a k-value of 1 often overfits to the training dataset, so a k-value of 3 or 5 is preferred and is shown above when we compare the testing error in section (3).

Appendix: Python code

```
#Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split

#Read in dataset and filter to labels 2 or 7
ziptrain = pd.read_csv("zip.train.csv",header=None)
ziptrain27 = ziptrain[(ziptrain.iloc[:,0] == 2) | (ziptrain.iloc[:,0] == 7)]
ziptrain27.describe()

#Visualize data of 7
data_5 = ziptrain27.iloc[5,:][1:]
data_5_reshape = np.array(data_5).reshape((16,16))
plt.imshow(data_5_reshape, cmap='gray')
plt.title('5th row data: Label 7')
plt.show()

#Visualize data of 2
data_37 = ziptrain27.iloc[37,:][1:]
data_37_reshape = np.array(data_37).reshape((16,16))
plt.imshow(data_37_reshape, cmap='gray')
plt.title('37th row data: Label 2')
plt.show()

#Split into X and Y
X_train = ziptrain27.iloc[:,1:]
y_train = ziptrain27.iloc[:,0]

#Train linear regression model and predict
lm = LinearRegression().fit(X_train, y_train)
y_logit = lm.predict(X_train)
y_predict = 2+5*(y_logit>=4.5)
lm_train_error = np.mean(y_predict != y_train)

#Train KNN model and predict
k_range = [1,3,5,7,9,11,13,15]
knn_train_errors = []
for k in k_range:
    knn_model = KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
    y_knn_predict = knn_model.predict(X_train)
    knn_err = np.mean(y_knn_predict != y_train)
    knn_train_errors.append(knn_err)
knn_train_df = pd.DataFrame({'k value':k_range,'Train
Errors':knn_train_errors})
knn_train_df.plot.line('k value','Train Errors',title='Train Errors vs. k
value',marker='o')

#Read in test dataset and split into X and Y
ziptest = pd.read_csv("zip.test.csv",header=None)
ziptest27 = ziptest[(ziptest.iloc[:,0] == 2) | (ziptest.iloc[:,0] == 7)]
X_test = ziptest27.iloc[:,1:]
y_test = ziptest27.iloc[:,0]

#Predict on test dataset with linear regression model
y_test_logit = lm.predict(X_test)
y_test_predict = 2+5*(y_test_logit>=4.5)
lm_test_error = np.mean(y_test_predict != y_test)

#Predict on test dataset with KNN model
k_range = [1,3,5,7,9,11,13,15]
knn_test_errors = []
for k in k_range:
    knn_model = KNeighborsClassifier(n_neighbors=k).fit(X_train,y_train)
    y_knn_test_predict = knn_model.predict(X_test)
    test_knn_err = np.mean(y_knn_test_predict != y_test)
    knn_test_errors.append(test_knn_err)
knn_test_df = pd.DataFrame({'k value':k_range,'Test Errors':knn_test_errors})
knn_test_df.plot.line('k value','Test Errors',title='Test Errors vs. k
value',marker='o')

#Combine dataset
zipfull27 = pd.concat([ziptrain27,ziptest27], ignore_index=True)
X_full = zipfull27.iloc[:,1:]
y_full = zipfull27.iloc[:,0]

#Monte carlo cross validation
B = 100 #total loop times
all_test_errors = []
k_range = [1,3,5,7,9,11,13,15]

for b in range(B):
    test_errors = []
    #Split train and test dataset based on random state
    X_temp_train, X_temp_test, y_temp_train, y_temp_test =
train_test_split(X_full, y_full, test_size=345, random_state=b)
    #Train LinearRegression
    lr_model = LinearRegression().fit(X_temp_train, y_temp_train)
    y_temp_test_logit = lr_model.predict(X_temp_test)
    y_temp_predict = 2+5*(y_temp_test_logit>=4.5)
    lm_test_error = np.mean(y_temp_predict != y_temp_test)
    test_errors.append(lm_test_error)
    #Train KNN
    for k in k_range:
        knn_model =
KNeighborsClassifier(n_neighbors=k).fit(X_temp_train,y_temp_train)
        y_knn_temp_test_predict = knn_model.predict(X_temp_test)
        test_knn_err = np.mean(y_knn_temp_test_predict != y_temp_test)
        test_errors.append(test_knn_err)
    all_test_errors.append(test_errors)

all_test_errors = np.array(all_test_errors)
index_names = ['LR','KNN1','KNN3','KNN5','KNN7','KNN9','KNN11','KNN13','KNN15']
all_test_err_df = pd.DataFrame({'Mean':np.mean(all_test_errors,axis=0),
                                'Variance':np.var(all_test_errors, axis=0)},
                               index=index_names)

#Plot mean and variance test error for each model
all_test_err_df.plot.line(y='Mean',title='Mean Error vs. Model', marker='o')
all_test_err_df.plot.line(y='Variance',title='Error Variance vs. Model',
marker='o')
```