

# ISYE 7406 - Homework 1

Shehzad Anwar

August 31, 2025

## Introduction

The purpose of this homework assignment is to perform analysis on and compare two methods of analysis, Linear Regression and KNN classification, using the *zipcode* data set. The data set contains 16 x 16 grayscale images of handwritten digits, ranging from 1-9.

The problem that was in this assignment was comparing the performances of the two methods, the error variables, and focusing on two specific digits, '2' and '7', for evaluating the data set.

## Exploratory Data Analysis

Filtering the data set to include only the response value digits, 2 and 7, we were able to find that the data set consists of 257 columns, the first one representing the response variable and the rest of the 256 representing their own pixel value. There is a total of 1376 observations in the data set. The total number of response variables that are '2' is 731, 53%, with '7' having 645.

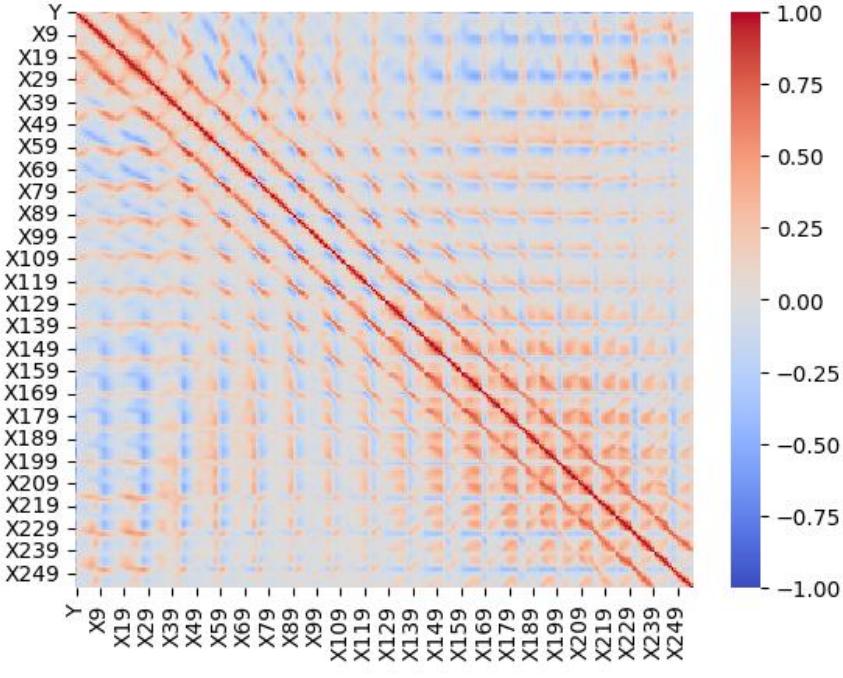


Image 1

Image 1 in the previous page shows a graph, more specifically, a correlation heatmap with the colors inverted to allow for easier differentiation and viewing of the pixels. Red indicates positive correlation, white indicates no correlation and blue indicates negative correlation. Overall there does seem to be a overall high level of correlation with the response variable, both positive and negative. There is high negative correlation in between the first couple of rows and pixels towards the end. It is likely that if the pixels are positively correlated, they are within the vicinity of other pixels like them.

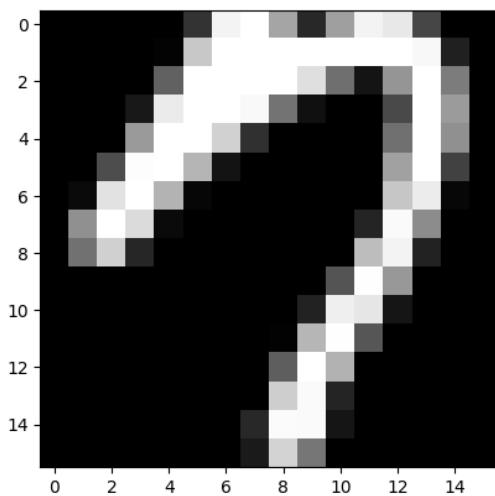
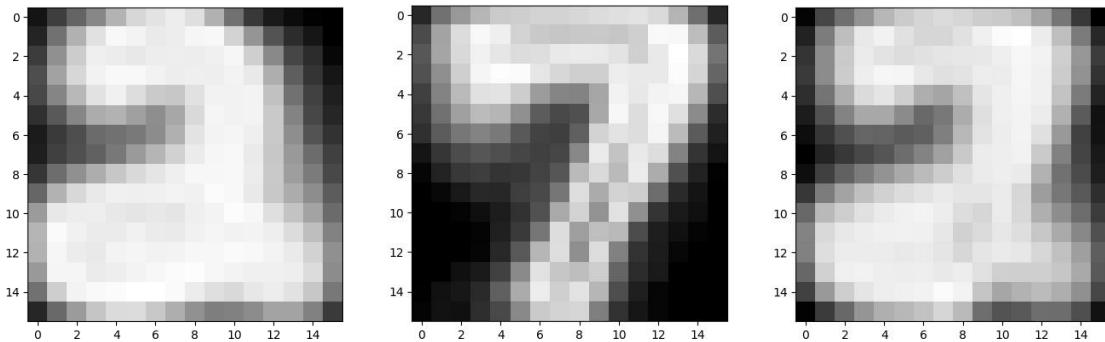


Image 2

Image 2 is what is output when each row of pixels is represented as an image. The grayscale value for each is a number will be between black, -1, and white, +1. Image 2 is representative of Y=7.



The above images represent the average images for each response type. The first image is the average of “2” as the response variable, the second is the average of “7” as the response variable, and the third is the average of the all the observations.

## Methodology

There were two types of models that were utilized in this analysis. The first was a linear regression model that was built based on the data. The second was the K Nearest Neighbors (KNN) algorithm. The linear regression model was developed using 2 or 7 as the response variables. The midpoint between the two numbers, 4.5, was set as the boundary value. I was able to calculate the error measurements, i.e. training error, testing error and cross-validation. As for the KNN model, the k values used were (1, 3, 5, 7, 9, 11, 13, 15). The error measurements for this model were also found. The cross-validation process that was used in this analysis was the Monte Carlo validation.

## Results

Model	Training Error %	Test Error %
Linear Regression	0.073	1.739
KNN, k=1	0.000	1.739
KNN, k=3	1.017	1.449
KNN, k=5	1.236	1.449
KNN, k=7	1.454	1.739
KNN, k=9	1.599	1.739
KNN, k=11	1.599	1.739
KNN, k=13	1.744	2.029
KNN, k=15	1.744	2.029

The table above shows the training and test error percentages from the different model tests. All the model results show training error percentages less than the test error percentages, but despite that, they are close in value to each other, indicating that the models are accurate. In terms of performance, k=3 and k=5 models were the most accurate, while k=13 and k=15 had the highest training and test error percentages. Utilizing higher k values will cause issues on the edges of the data as the outliers could cause a shift in the readings.

Model	Cross-Valid. Test Error %	Cross-Valid. Test Variance
Linear Regression	1.176	$2.6 \times 10^{-5}$
KNN, k=1	1.258	$2.5 \times 10^{-5}$
KNN, k=3	1.310	$2.8 \times 10^{-5}$
KNN, k=5	1.528	$4 \times 10^{-5}$
KNN, k=7	1.673	$4.5 \times 10^{-5}$
KNN, k=9	1.742	$4.8 \times 10^{-5}$
KNN, k=11	1.855	$5.2 \times 10^{-5}$
KNN, k=13	1.947	$5.1 \times 10^{-5}$
KNN, k=15	1.986	$5.6 \times 10^{-5}$

The table above shows the Monte Carlo cross-validation readings. Among the models, linear regression definitely has the least amount of test error, while also having the second least test variance. As for the KNN models, k=1 is the most accurate, having the least amount of test error percentage as well as the least, out of all the tests, cross-validation test variance. Judging by the numbers, the higher the k value is, the higher the error percentage and variance seems to be.

## Conclusion

From the analysis and resulting values from the tests, we can conclude that the linear Regression and KNN are both reliable models of analysis. The linear regression tests performed well when performed on the data set. The KNN model showed that the higher values of k you use, the higher the error percentage potentially, as well as increasing variance, meaning that its less accurate. Its important to note that adding in cross-validation reduced the average test error percentage across the board, apart from the higher k values. In conclusion, using low k parameters will yield the most accurate results in this particular data set.

## Appendix

Anwar\_HW1\_cv.py (used for cross-validation):

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier

N_ROUNDS = 100
R_SEED = 7406

def read_data() -> tuple:
    ziptrain = pd.read_csv("ISYE7406/HW1/zip.train.csv", header=None)
    ziptest = pd.read_csv("ISYE7406/HW1/zip.test.csv", header=None)
    col_names = ["Y"]
    col_names.extend([str(i) for i in range(ziptrain.shape[1]-1)])
```

```

ziptrain.columns = col_names
ziptrain = ziptrain[ziptrain["Y"].isin([2,7])]
ziptest.columns = col_names
ziptest = ziptest[ziptest["Y"].isin([2,7])]
return (ziptrain, ziptest)

def monteCarlo(df, ntest=-1, rand=None) -> tuple:
    y = df["Y"]
    x = df.drop(columns=[ "Y"])
    if ntest == -1:
        ntest = np.floor(df.shape[0] * 0.3)
    split = train_test_split(x, y, test_size=ntest, random_state=rand)
    return split

def LR_error(train_x, test_x, train_y, test_y) -> float:
    model = LinearRegression().fit(train_x, train_y)
    preds = model.predict(test_x)
    preds = 2+5*(preds>=4.5)
    error = np.mean(preds != test_y)
    return error

def knn_error(train_x, test_x, train_y, test_y, k=1) -> float:
    model = KNeighborsClassifier(n_neighbors=k)
    model.fit(train_x, train_y)
    preds = model.predict(test_x)
    error = np.mean(preds != test_y)
    return error

def main():
    rand = np.random.RandomState(R_SEED)
    train, test = read_data()
    ntest = test.shape[0]
    df = pd.concat([train, test])
    col_names = ["LinearReg", "KNN1", "KNN3", "KNN5", "KNN7", "KNN9", "KNN11",
    "KNN13", "KNN15"]
    err_test = pd.DataFrame(None, columns=col_names)

    for i in range(N_ROUNDS):
        current_row = []
        train_x, test_x, train_y, test_y = monteCarlo(df, ntest, rand)
        lin_error = LR_error(train_x, test_x, train_y, test_y)
        current_row.append(lin_error)
        k_choice = list(range(1, 16, 2))

```

```
for k in k_choice:
    knn_err = knn_error(train_x, test_x, train_y, test_y, k)
    current_row.append(knn_err)
current_row = pd.DataFrame([current_row], columns=col_names)
err_test = pd.concat([err_test, current_row], ignore_index=True)

summary = err_test.describe()
summary = summary.apply(lambda x: np.square(x) if x.name == 'std' else x)
summary = summary.rename(index={"std": "var"})
print(summary)

if __name__ == "__main__":
    main()
```

Anwar\_Shehzad\_HW1.ipynb (used for Exploratory Data Analysis, Training and Test Error):

```

D ✓
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.neighbors import KNeighborsClassifier
[80] Python

Read and Train Data
Generate + Code + Markdown

ziptrain = pd.read_csv('zip.train.csv', header=None)
ziptest = pd.read_csv('zip.test.csv', header=None)
col_names = ["Y"]
col_names.extend(["X%d" % i for i in range(ziptrain.shape[1]-1)])
ziptrain.columns = col_names
ziptest.columns = col_names
ziptrain27 = ziptrain[ziptrain["Y"].isin([2, 7])]
ziptest27 = ziptest[ziptest["Y"].isin([2, 7])]
ziptrain27.head()
[81] Python
...
   Y   X0   X1   X2   X3   X4   X5   X6   X7   X8   ...   X246   X247   X248   X249   X250   X251   X252   X253   X254
3  7 -1.0 -1.0 -1.0 -1.000 -1.000 -0.273  0.684  0.960  0.450 ... -0.318  1.000  0.536 -0.987 -1.0 -1.0 -1.0 -1.0 -1
10 7 -1.0 -1.0 -1.0 -1.000 -1.000 -0.596  0.912  1.000  0.290 ... -1.000 -0.795  0.663 -0.074 -1.0 -1.0 -1.0 -1.0 -1
14 7 -1.0 -1.0 -1.0 -1.000 -1.000 -1.000 -1.000 -1.000 -0.632 ... -1.000 -0.967  0.866 -0.001 -1.0 -1.0 -1.0 -1.0 -1
15 7 -1.0 -1.0 -1.0 -0.929  0.351  0.798  0.806  0.114  0.015 ...  0.835 -0.086 -0.991 -1.000 -1.0 -1.0 -1.0 -1.0 -1
22 7 -1.0 -1.0 -1.0 -1.000 -0.869  0.777 -0.007 -0.697 -1.000 ... -0.933  0.667 -0.315 -1.000 -1.0 -1.0 -1.0 -1.0 -1
5 rows × 257 columns

```

1. Exploratory Data Analysis of Training Data

```

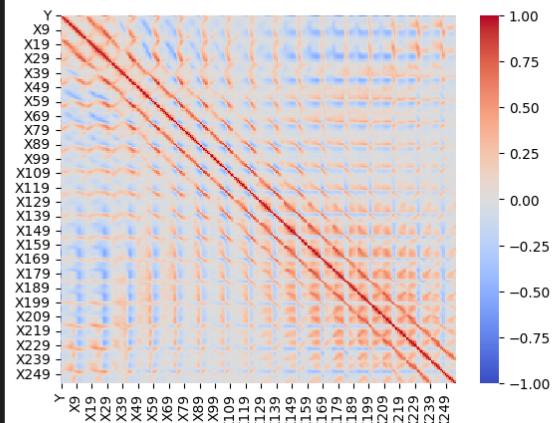
summary = ziptrain27.describe()
summary
[82] Python
...
   Y      X0      X1      X2      X3      X4      X5      X6      X7      X8      X9      X10      X11      X12      X13      X14      X15      X16      X17      X18      X19      X20      X21      X22      X23      X24      X25      X26      X27      X28      X29      X30      X31      X32      X33      X34      X35      X36      X37      X38      X39      X40      X41      X42      X43      X44      X45      X46      X47      X48      X49      X50      X51      X52      X53      X54      X55      X56      X57      X58      X59      X60      X61      X62      X63      X64      X65      X66      X67      X68      X69      X70      X71      X72      X73      X74      X75      X76      X77      X78      X79      X80      X81      X82      X83      X84      X85      X86      X87      X88      X89      X90      X91      X92      X93      X94      X95      X96      X97      X98      X99      X100      X101      X102      X103      X104      X105      X106      X107      X108      X109      X110      X111      X112      X113      X114      X115      X116      X117      X118      X119      X120      X121      X122      X123      X124      X125      X126      X127      X128      X129      X130      X131      X132      X133      X134      X135      X136      X137      X138      X139      X140      X141      X142      X143      X144      X145      X146      X147      X148      X149      X150      X151      X152      X153      X154      X155      X156      X157      X158      X159      X159      X160      X161      X162      X163      X164      X165      X166      X167      X168      X169      X170      X171      X172      X173      X174      X175      X176      X177      X178      X179      X180      X181      X182      X183      X184      X185      X186      X187      X188      X189      X189      X190      X191      X192      X193      X194      X195      X196      X197      X198      X199      X199      X200      X201      X202      X203      X204      X205      X206      X207      X208      X209      X209      X210      X211      X212      X213      X214      X215      X216      X217      X218      X219      X219      X220      X221      X222      X223      X224      X225      X226      X227      X228      X229      X229      X230      X231      X232      X233      X234      X235      X236      X237      X238      X239      X239      X240      X241      X242      X243      X244      X245      X246      X247      X248      X249      X249      X250      X251      X252      X253      X254      X255      X256      X257      X258      X259      X259      X260      X261      X262      X263      X264      X265      X266      X267      X268      X269      X269      X270      X271      X272      X273      X274      X275      X276      X277      X278      X279      X279      X280      X281      X282      X283      X284      X285      X286      X287      X288      X289      X289      X290      X291      X292      X293      X294      X295      X296      X297      X298      X299      X299      X300      X301      X302      X303      X304      X305      X306      X307      X308      X309      X309      X310      X311      X312      X313      X314      X315      X316      X317      X318      X319      X319      X320      X321      X322      X323      X324      X325      X326      X327      X328      X329      X329      X330      X331      X332      X333      X334      X335      X336      X337      X338      X339      X339      X340      X341      X342      X343      X344      X345      X346      X347      X348      X349      X349      X350      X351      X352      X353      X354      X355      X356      X357      X358      X359      X359      X360      X361      X362      X363      X364      X365      X366      X367      X368      X369      X369      X370      X371      X372      X373      X374      X375      X376      X377      X378      X379      X379      X380      X381      X382      X383      X384      X385      X386      X387      X388      X389      X389      X390      X391      X392      X393      X394      X395      X396      X397      X398      X399      X399      X400      X401      X402      X403      X404      X405      X406      X407      X408      X409      X409      X410      X411      X412      X413      X414      X415      X416      X417      X418      X419      X419      X420      X421      X422      X423      X424      X425      X426      X427      X428      X429      X429      X430      X431      X432      X433      X434      X435      X436      X437      X438      X439      X439      X440      X441      X442      X443      X444      X445      X446      X447      X448      X449      X449      X450      X451      X452      X453      X454      X455      X456      X457      X458      X459      X459      X460      X461      X462      X463      X464      X465      X466      X467      X468      X469      X469      X470      X471      X472      X473      X474      X475      X476      X477      X478      X479      X479      X480      X481      X482      X483      X484      X485      X486      X487      X488      X489      X489      X490      X491      X492      X493      X494      X495      X496      X497      X498      X499      X499      X500      X501      X502      X503      X504      X505      X506      X507      X508      X509      X509      X510      X511      X512      X513      X514      X515      X516      X517      X518      X519      X519      X520      X521      X522      X523      X524      X525      X526      X527      X528      X529      X529      X530      X531      X532      X533      X534      X535      X536      X537      X538      X539      X539      X540      X541      X542      X543      X544      X545      X546      X547      X548      X549      X549      X550      X551      X552      X553      X554      X555      X556      X557      X558      X559      X559      X560      X561      X562      X563      X564      X565      X566      X567      X568      X569      X569      X570      X571      X572      X573      X574      X575      X576      X577      X578      X579      X579      X580      X581      X582      X583      X584      X585      X586      X587      X588      X589      X589      X590      X591      X592      X593      X594      X595      X596      X597      X598      X599      X599      X600      X601      X602      X603      X604      X605      X606      X607      X608      X609      X609      X610      X611      X612      X613      X614      X615      X616      X617      X618      X619      X619      X620      X621      X622      X623      X624      X625      X626      X627      X628      X629      X629      X630      X631      X632      X633      X634      X635      X636      X637      X638      X639      X639      X640      X641      X642      X643      X644      X645      X646      X647      X648      X649      X649      X650      X651      X652      X653      X654      X655      X656      X657      X658      X659      X659      X660      X661      X662      X663      X664      X665      X666      X667      X668      X669      X669      X670      X671      X672      X673      X674      X675      X676      X677      X678      X679      X679      X680      X681      X682      X683      X684      X685      X686      X687      X688      X689      X689      X690      X691      X692      X693      X694      X695      X696      X697      X698      X699      X699      X700      X701      X702      X703      X704      X705      X706      X707      X708      X709      X709      X710      X711      X712      X713      X714      X715      X716      X717      X718      X719      X719      X720      X721      X722      X723      X724      X725      X726      X727      X728      X729      X729      X730      X731      X732      X733      X734      X735      X736      X737      X738      X739      X739      X740      X741      X742      X743      X744      X745      X746      X747      X748      X749      X749      X750      X751      X752      X753      X754      X755      X756      X757      X758      X759      X759      X760      X761      X762      X763      X764      X765      X766      X767      X768      X769      X769      X770      X771      X772      X773      X774      X775      X776      X777      X778      X779      X779      X780      X781      X782      X783      X784      X785      X786      X787      X788      X789      X789      X790      X791      X792      X793      X794      X795      X796      X797      X798      X799      X799      X800      X801      X802      X803      X804      X805      X806      X807      X808      X809      X809      X810      X811      X812      X813      X814      X815      X816      X817      X818      X819      X819      X820      X821      X822      X823      X824      X825      X826      X827      X828      X829      X829      X830      X831      X832      X833      X834      X835      X836      X837      X838      X839      X839      X840      X841      X842      X843      X844      X845      X846      X847      X848      X849      X849      X850      X851      X852      X853      X854      X855      X856      X857      X858      X859      X859      X860      X861      X862      X863      X864      X865      X866      X867      X868      X869      X869      X870      X871      X872      X873      X874      X875      X876      X877      X878      X879      X879      X880      X881      X882      X883      X884      X885      X886      X887      X888      X889      X889      X890      X891      X892      X893      X894      X895      X896      X897      X898      X899      X899      X900      X901      X902      X903      X904      X905      X906      X907      X908      X909      X909      X910      X911      X912      X913      X914      X915      X916      X917      X918      X919      X919      X920      X921      X922      X923      X924      X925      X926      X927      X928      X929      X929      X930      X931      X932      X933      X934      X935      X936      X937      X938      X939      X939      X940      X941      X942      X943      X944      X945      X946      X947      X948      X949      X949      X950      X951      X952      X953      X954      X955      X956      X957      X958      X959      X959      X960      X961      X962      X963      X964      X965      X966      X967      X968      X969      X969      X970      X971      X972      X973      X974      X975      X976      X977      X978      X979      X979      X980      X981      X982      X983      X984      X985      X986      X987      X988      X989      X989      X990      X991      X992      X993      X994      X995      X996      X997      X998      X999      X999      X1000      X1001      X1002      X1003      X1004      X1005      X1006      X1007      X1008      X1009      X1009      X1010      X1011      X1012      X1013      X1014      X1015      X1016      X1017      X1018      X1019      X1019      X1020      X1021      X1022      X1023      X1024      X1025      X1026      X1027      X1028      X1029      X1029      X1030      X1031      X1032      X1033      X1034      X1035      X1036      X1037      X1038      X1039      X1039      X1040      X1041      X1042      X1043      X1044      X1045      X1046      X1047      X1048      X1049      X1049      X1050      X1051      X1052      X1053      X1054      X1055      X1056      X1057      X1058      X1059      X1059      X1060      X1061      X1062      X1063      X1064      X1065      X1066      X1067      X1068      X1069      X1069      X1070      X1071      X1072      X1073      X1074      X1075      X1076      X1077      X1078      X1079      X1079      X1080      X1081      X1082      X1083      X1084      X1085      X1086      X1087      X1088      X1089      X1089      X1090      X1091      X1092      X1093      X1094      X1095      X1096      X1097      X1098      X1099      X1099      X1100      X1101      X1102      X1103      X1104      X1105      X1106      X1107      X1108      X1109      X1109      X1110      X1111      X1112      X1113      X1114      X1115      X1116      X1117      X1118      X1119      X1119      X1120      X1121      X1122      X1123      X1124      X1125      X1126      X1127      X1128      X1129      X1129      X1130      X1131      X1132      X1133      X1134      X1135      X1136      X1137      X1138      X1139      X1139      X1140      X1141      X1142      X1143      X1144      X1145      X1146      X1147      X1148      X1149      X1149      X1150      X1151      X1152      X1153      X1154      X1155      X1156      X1157      X1158      X1159      X1159      X1160      X1161      X1162      X1163      X1164      X1165      X1166      X1167      X1168      X1169      X1169      X1170      X1171      X1172      X1173      X1174      X1175      X1176      X1177      X1178      X1179      X1179      X1180      X1181      X1182      X1183      X1184      X1185      X1186      X1187      X1188      X1189      X1189      X1190      X1191      X1192      X1193      X1194      X1195      X1196      X1197      X1198      X1199      X1199      X1200      X1201      X1202      X1203      X1204      X1205      X1206      X1207      X1208      X1209      X1209      X1210      X1211      X1212      X1213      X1214      X1215      X1216      X1217      X1218      X1219      X1219      X1220      X1221      X1222      X1223      X1224      X1225      X1226      X1227      X1228      X1229      X1229      X1230      X1231      X1232      X1233      X1234      X1235      X1236      X1237      X1238      X1239      X1239      X1240      X1241      X1242      X1243      X1244      X1245      X1246      X1247      X1248      X1249      X1249      X1250      X1251      X1252      X1253      X1254      X1255      X1256      X1257      X1258      X1259      X1259      X1260      X1261      X1262      X1263      X1264      X1265      X1266      X1267      X1268      X1269      X1269      X1270      X1271      X1272      X1273      X1274      X1275      X1276      X1277      X1278      X1279      X1279      X1280      X1281      X1282      X1283      X1284      X1285      X1286      X1287      X1288      X1289      X1289      X1290      X1291      X1292      X1293      X1294      X1295      X1296      X1297      X1298      X1299      X1299      X1300      X1301      X1302      X1303      X1304      X1305      X1306      X1307      X1308      X1309      X1309      X1310      X1311      X1312      X1313      X1314      X1315      X1316      X1317      X1318      X1319      X1319      X1320      X1321      X1322      X1323      X1324      X1325      X1326      X1327      X1328      X1329      X1329      X1330      X1331      X1332      X1333      X1334      X1335      X1336      X1337      X1338      X1339      X1339      X1340      X1341      X1342      X1343      X1344      X1345      X1346      X1347      X1348      X1349      X1349      X1350      X1351      X1352      X1353      X1354      X1355      X1356      X1357      X1358      X1359      X1359      X1360      X1361      X1362      X1363      X1364      X1365      X1366      X1367      X1368      X1369      X1369      X1370      X1371      X1372      X1373      X1374      X1375      X1376      X1377      X1378      X1379      X1379      X1380      X1381      X1382      X1383      X1384      X1385      X1386      X1387      X1388      X1389      X1389      X1390      X1391      X1392      X1393      X1394      X1395      X1396      X1397      X1398      X1399      X1399      X1400      X1401      X1402      X1403      X1404      X1405      X1406      X1407      X1408      X1409      X1409      X1410      X1411      X1412      X1413      X1414      X1415      X1416      X1417      X1418      X1419      X1419      X1420      X1421      X1422      X1423      X1424      X1425      X1426      X1427      X1428      X1429      X1429      X1430      X1431      X1432      X1433      X1434      X1435      X1436      X1437      X1438      X1439      X1439      X1440      X1441      X1442      X1443      X1444      X1445      X1446      X1447      X1448      X1449      X1449      X1450      X1451      X1452      X1453      X1454      X1455      X1456      X1457      X1458      X1459      X1459      X1460      X1461      X1462      X1463      X1464      X1465      X1466      X1467      X1468      X1469      X1469      X1470      X1471      X1472      X1473      X1474      X1475      X1476      X1477      X1478      X1479      X1479      X1480      X1481      X1482      X1483      X1484      X1485      X1486      X1487      X1488      X1489      X1489      X1490      X1491      X1492      X1493      X1494      X1495      X1496      X1497      X1498      X1499      X1499      X1500      X1501      X1502      X1503      X1504      X1505      X1506      X1507      X1508      X1509      X1509      X1510      X1511      X1512      X1513      X1514      X1515      X1516      X1517      X1518      X1519      X1519      X1520      X1521      X1522      X1523      X1524      X1525      X1526      X1527      X1528      X1529      X1529      X1530      X1531      X1532      X1533      X1534      X1535      X1536      X1537      X1538      X1539      X1539      X1540      X1541      X1542      X1543      X1544      X1545      X1546      X1547      X1548      X1549      X1549      X1550      X1551      X1552      X1553      X1554      X1555      X1556      X1557      X1558      X1559      X1559      X1560      X1561      X1562      X1563      X1564      X1565      X1566      X1567      X1568      X1569      X1569      X1570      X1571      X1572      X1573      X1574      X1575      X1576      X1577      X1578      X1579      X1579      X1580      X1581      X1582      X1583      X1584      X1585      X1586      X1587      X1588      X1589      X1589      X1590      X1591      X1592      X1593      X1594      X1595      X1596      X1597      X1598      X1599      X1599      X1600      X1601      X1602      X1603      X1604      X1605      X1606      X1607      X1608      X1609      X1609      X1610      X1611      X1612      X1613      X1614      X1615      X1616      X1617      X1618      X1619      X1619      X1620      X1621      X1622      X1623      X1624      X1625      X1626      X1627      X1628      X1629      X1629      X1630      X1631      X1632      X1633      X1634      X1635      X1636      X1637      X1638      X1639      X1639      X1640      X1641      X1642      X1643      X1644      X1645      X1646      X1647      X1648      X1649      X1649      X1650      X1651      X1652      X1653      X1654      X1655      X1656      X1657      X1658      X1659      X1659      X1660      X1661      X1662      X1663      X1664      X1665      X1666      X1667      X1668      X1669      X1669      X1670      X1671      X1672      X1673      X1674      X1675      X1676      X1677      X1678      X1679      X1679      X1680      X1681      X1682      X1683      X1684      X1685      X1686      X1687      X1688      X1689      X1689      X1690      X1691      X1692      X1693      X1694      X1695      X1696      X1697      X1698      X1699      X1699      X1700      X1701      X1702      X1703      X1704      X1705      X1706      X1707      X1708      X1709      X1709      X1710      X1711      X1712      X1713      X1714      X1715      X1716      X1717      X1718      X1719      X1719      X1720      X1721      X1722      X1723      X1724      X1725      X1726      X1727      X1728      X1729      X1729      X1730      X1731      X1732      X1733      X1734      X1735      X1736      X1737      X1738      X1739      X1739      X1740      X1741      X1742      X1743      X1744      X1745      X1746      X1747      X1748      X1749      X1749      X1750      X1751      X1752      X1753      X1754      X1755      X1756      X1757      X1758      X1759      X1759      X1760      X1761      X1762      X1763      X1764      X1765      X1766      X1767      X1768      X1769      X1769      X1770      X1771      X1772      X1773      X1774      X1775      X1776      X1777      X1778      X1779      X1779      X1780      X1781      X1782      X1783      X1784      X1785      X1786      X1787      X1788      X1789      X1789      X1790      X1791      X1792      X1793      X1794      X1795      X179
```

```

[83] ziptrain.shape
Python
... (7291, 257)

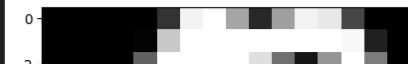
[84] ziptrain[ziptrain["Y"] == 2].shape
Python
... (731, 257)

[85] ziptrain[ziptrain["Y"] == 7].shape
Python
... (645, 257)

[86] sns.heatmap(ziptrain27.corr(), vmin = -1, vmax = 1, cmap="coolwarm")
plt.show()
Python
... 

```

```

[87] image_row = 1
image_arr = ziptrain27.iloc[image_row, 1:]
image_arr = image_arr.values.reshape((16, 16))
plt.gray()
plt.imshow(image_arr)
print(ziptrain27["Y"].iloc[image_row])
Python
... 7
... 

```





Training Errors

Linear Regression

```
[98] y = ziptrain27["Y"]
x = ziptrain27.drop(columns=["Y"])
mod1 = LinearRegression().fit(x,y)
pred1 = mod1.predict(x)
y1pred = 2 + 5 * (pred1 >= 4.5)
train_err = np.mean(y1pred != y)
print("Training Error for Linear Regression = %0.05f"%(train_err))
... Training Error for Linear Regression = 0.00073
... Python
```

KNN

```
[92] kk = list(range(1,16,2))
ypred2 = []
knn_mod = []
for k in kk:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(x,y)
    kpred = knn.predict(x)
    t_err = np.mean(kpred != y)
... Python
```

## KNN

```
[92] >     kk = list(range(1,16,2))
      ypred2 = []
      knn_mod = []
    <for k in kk:
        knn = KNeighborsClassifier(n_neighbors=k)
        knn.fit(x,y)
        kpred = knn.predict(x)
        t_err = np.mean(kpred != y)
        ypred2.append(t_err)
        knn_mod.append(knn)

knn_names = ["KNN%d"%(k) for k in kk]
ypred2 = pd.DataFrame(ypred2, index=knn_names, columns=["Training Error"])
ypred2
```

Python

Training Error	
KNN1	0.000000
KNN3	0.010174
KNN5	0.012355
KNN7	0.014535
KNN9	0.015988
KNN11	0.015988
KNN13	0.017442
KNN15	0.017442

## Testing Errors

### Linear Regression

```
[93] >     test_err_y = ziptest27["Y"]
      test_err_x = ziptest27.drop(columns=["Y"])
      test_pred1 = mod1.predict(test_err_x)
      test_y1pred = 2 + 5 * (test_pred1 >= 4.5)
      test_err = np.mean(test_y1pred != test_err_y)
      print("Test Error for Linear Regression = %0.05f%(test_err)")

... Test Error for Linear Regression = 0.01739
```

Python

## KNN

```
[94] >     test_err_knn = []
      for knn in knn_mod:
        tknn_pred = knn.predict(test_err_x)
        tknn_err = np.mean(tknn_pred != test_err_y)
        test_err_knn.append(tknn_err)

      test_err_knn = pd.DataFrame(test_err_knn, index=knn_names, columns=["Test Error"])
      test_err_knn
```

Python

## Testing Errors

### Linear Regression

```
[93] test_err_y = ziptest27["Y"]
      test_err_x = ziptest27.drop(columns=["Y"])
      test_pred1 = mod1.predict(test_err_x)
      test_y1pred = 2 + 5 * (test_pred1 >= 4.5)
      test_err = np.mean(test_y1pred != test_err_y)
      print("Test Error for Linear Regression = %0.05f"%(test_err))
```

Python

```
... Test Error for Linear Regression = 0.01739
```

### KNN

```
[94] test_err_knn = []
      for knn in knn_mod:
          tknn_pred = knn.predict(test_err_x)
          tknn_err = np.mean(tknn_pred != test_err_y)
          test_err_knn.append(tknn_err)

      test_err_knn = pd.DataFrame(test_err_knn, index=knn_names, columns=["Test Error"])
      test_err_knn
```

Python

```
...    Test Error
KNN1    0.017391
KNN3    0.014493
KNN5    0.014493
KNN7    0.017391
KNN9    0.017391
KNN11   0.017391
KNN13   0.020290
KNN15   0.020290
```