In [12]:
```
import pandas as pd
import numpy as np
from scipy.stats import mode
```

Problem 1

In [49]:
```
train = open('pa1train.txt', 'r')
X_train = []
y_train = []
for line in train.readlines():
    vec = line.split(' ')
    X_train.append(vec[:-1])
    y_train.append(vec[-1][0])

X_train = pd.DataFrame(X_train)
y_train = pd.DataFrame(y_train)
```

In [50]:
```
val = open('pa1validate.txt', 'r')
X_val = []
y_val = []
for line in val.readlines():
    vec = line.split(' ')
    X_val.append(vec[:-1])
    y_val.append(vec[-1][0])

X_val = pd.DataFrame(X_val)
y_val = pd.DataFrame(y_val)
```

In [51]:
```
test = open('pa1test.txt', 'r')
X_test = []
y_test = []
for line in test.readlines():
    vec = line.split(' ')
    X_test.append(vec[:-1])
    y_test.append(vec[-1][0])

X_test = pd.DataFrame(X_test)
y_test = pd.DataFrame(y_test)
```

In [52]:
```
def eucdist(v1, v2):
    return np.linalg.norm(v1 - v2)
```

In [53]:
```
for i in X_train.columns:
    X_train[i] = X_train[i].astype(int)
```

In [54]:
```
for i in X_val.columns:
    X_val[i] = X_val[i].astype(int)
```

In [55]:
```
for i in X_test.columns:
    X_test[i] = X_test[i].astype(int)
```

In [56]:
```
X_train = X_train.to_numpy()
X_val = X_val.to_numpy()
X_test = X_test.to_numpy()

y_train = y_train.iloc[:, 0].to_numpy()
y_val = y_val.iloc[:, 0].to_numpy()
y_test = y_test.iloc[:, 0].to_numpy()
```

In [61]:
```
# Calculate NNs
def NNs_pred(training_set, labels, x, k):
    preds = []
    for i in range(len(x)):
        distances = []
        for j in range(len(training_set)):
            distances.append((labels[j], eucdist(x[i], training_set[j
])))
        distances.sort(key=lambda item: item[1])
        distances = distances[:k]
        labs = [x[0] for x in distances]
        preds.append(mode(labs)[0][0])
    return np.array(preds)
```

In [62]:
```
# Training Error Calculation
training_errors = {}
for i in [1,3,5,9,15]:
    training_errors[i] = np.mean(NNs_pred(X_train, y_train, X_train, i)!
=y_train)
    print(i)
training_errors
```

```
1
3
5
9
15
```

Out[62]:
```
{1: 0.0, 3: 0.0435, 5: 0.0565, 9: 0.0685, 15: 0.0925}
```

In [67]:
```
# Validation Error Calculation
validation_errors = {}
for i in [1,3,5,9,15]:
    validation_errors[i] = np.mean(NNs_pred(X_train, y_train, X_val, i)!
=y_val)
    print(i)
validation_errors
```

```
1
3
5
9
15
```

Out[67]:
```
{1: 0.082, 3: 0.098, 5: 0.095, 9: 0.104, 15: 0.108}
```

In [68]:
```
# Test Error Calculation
test_errors = {}
for i in [1,3,5,9,15]:
    test_errors[i] = np.mean(NNs_pred(X_train, y_train, X_test, i)!=y_te
st)
    print(i)
test_errors
```

```
1
3
5
9
15
```

Out[68]:
```
{1: 0.094, 3: 0.092, 5: 0.098, 9: 0.101, 15: 0.114}
```

The classifier with the lowest validation error would be acheived by using the k = 1 parameter. This would mean that the test error of using this parameter is 9.4%.

Problem 2

In [98]:
```
projections = open('projection.txt', 'r')
proj = []
for line in projections.readlines():
    vec = line.split(' ')
    vec[-1] = vec[-1][:-1]
    vec = np.array(vec).astype(float)
    proj.append(vec)
```

In [100]:
```
train_proj = np.matmul(X_train, proj)
val_proj = np.matmul(X_val, proj)
test_proj = np.matmul(X_test, proj)
```

In [104]:
```
# Training Error Calculation
training_errors = {}
for i in [1,3,5,9,15]:
    training_errors[i] = np.mean(NNs_pred(train_proj, y_train, train_pro
j, i)!=y_train)
    print(i)
training_errors
```

```
1
3
5
9
15
```

Out[104]:
```
{1: 0.0, 3: 0.1605, 5: 0.1945, 9: 0.2305, 15: 0.257}
```

In [105]:
```
# Validation Error Calculation
validation_errors = {}
for i in [1,3,5,9,15]:
    validation_errors[i] = np.mean(NNs_pred(train_proj, y_train, val_pro
j, i)!=y_val)
    print(i)
validation_errors
```

```
1
3
5
9
15
```

Out[105]:
```
{1: 0.32, 3: 0.31, 5: 0.299, 9: 0.302, 15: 0.289}
```

In [106]:
```
# Test Error Calculation
test_errors = {}
for i in [1,3,5,9,15]:
    test_errors[i] = np.mean(NNs_pred(train_proj, y_train, test_proj, i)
!=y_test)
    print(i)
test_errors
```

```
1
3
5
9
15
```

Out[106]:
```
{1: 0.314, 3: 0.297, 5: 0.301, 9: 0.293, 15: 0.296}
```

We can see that the classification accuracy of the projected matrix is significantly lower than without projecting each vector. The benefits to projecting each vector is that the runtime is significantly lower however the accuracy has significantly decreased, which makes this a tradeoff between runtime and accuracy.

In [ ]: