Programming Report

(a) Linear Regression

Formulas:

$$\mathbf{w}_{\mathrm{ML}} = \left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} \tag{1}$$

where **Φ** is the design matrix which contains 3 vectors including FTP WE and the testing variable, **t** is the HOM column vector

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j \overline{\phi_j} \tag{2}$$

where t and Φ with bars are their means

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}_n)\}^2. \tag{3}$$

Thus, we can get the errors for each variable. And the variable with the smallest error might be the third variable.

After coding and running, we can get the third variable which might be GR or LIC since they are very close to each other (GR is smaller). Below is the plot of errors for each variables .

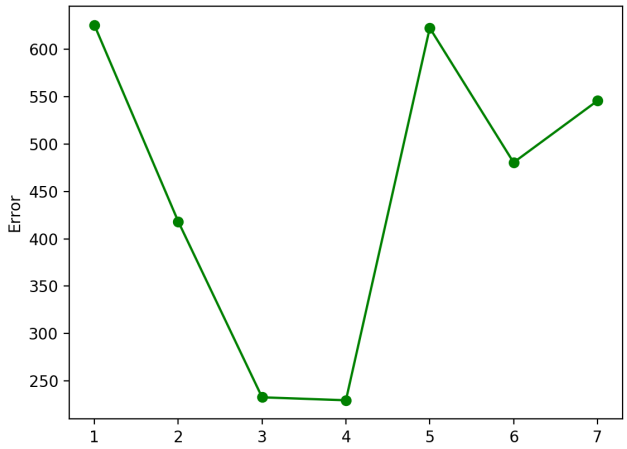

Figure 1: y-axis represents the error of each variable, x-axis represents variables as 1: UEMP, 2: MAN, 3: LIC, 4: GR, 5: NMAN, 6: GOV, 7: HE

(b) Nearest Neighbor

i.) Impute Missing Values:

For nominal features:

Replaced all "?" in feature 1 with b
Replaced all "?" in feature 4 with u
Replaced all "?" in feature 5 with g
Replaced all "?" in feature 6 with c
Replaced all "?" in feature 7 with v
Replaced all "?" in feature 9 with t
Replaced all "?" in feature 10 with f
Replaced all "?" in feature 12 with f
Replaced all "?" in feature 13 with g

For "+"labelled real-valued features:

Replaced all "?" in "+"labelled feature 2 with 33.72049180327869
Replaced all "?" in "+"labelled feature 3 with 5.904951140065147
Replaced all "?" in "+"labelled feature 8 with 3.427899022801302
Replaced all "?" in "+"labelled feature 11 with 4.605863192182411
Replaced all "?" in "+"labelled feature 14 with 164.421926910299
Replaced all "?" in "+"labelled feature 15 with 2038.85993485342

For "-"labelled real-valued features:

Replaced all "?" in "-"labelled feature 2 with 29.80823056300268
Replaced all "?" in "-"labelled feature 3 with 3.8399477806788513
Replaced all "?" in "-"labelled feature 8 with 1.2579242819843344
Replaced all "?" in "-"labelled feature 11 with 0.6318537859007833
Replaced all "?" in "-"labelled feature 14 with 199.6994680851064
Replaced all "?" in "-"labelled feature 15 with 198.60574412532637

Method:

Used Pandas module in Python to implement.

Output files:

crx.data.training.processed
crx.data.testing.processed

ii.) k-NN Algorithm:

Output files:

Distance

LabelledTesting

iii.) k-NN Accuracy:

k-value	Accuracy
1	0.8115942028985508
2	0.8115942028985508
3	0.855072463768116
4	0.8623188405797102
5	0.8478260869565217
6	0.8623188405797102
7	0.855072463768116
8	0.8623188405797102
9	0.8478260869565217
10	0.8840579710144928
11	0.855072463768116
12	0.8695652173913043
13	0.8623188405797102
14	0.8985507246376812
15	0.8695652173913043
16	0.8840579710144928
17	0.8840579710144928
18	0.8913043478260869
19	0.8840579710144928
...	.....

Table1: Accuracy Table

Conclusion:

The best k-value would be 14. (I have tested k from 1 to 137, there is no accuracy of k that is larger than the accuracy of k = 14)

Source Code

2.(a): Linear Regression

```
import pandas as pd
import numpy as np
import scipy.io as si
import matplotlib.pyplot as plt

def main():
    # Load ".mat" file, and create a DataFrame
    df = si.loadmat('/Users/qiujingye/Downloads/Homework2/detroit.mat')['data']
    df = pd.DataFrame(df)

    # Prepare the design matrix, combine FTP, WE and a test variable
    m = []
    for j in range(1, 8):
        temp = []
        for i in range(len(df)):
            row = [df[j][i],df[0][i],df[8][i]]
            temp.append(row)
        m.append(temp)

    # Test the rest 7 variables, and let t be the HOM column
    t = df[9]
    errors = []
    for i in range(7):
        # Create the design matrix p
        p = np.array(m[i])
        pb = range(3)
        tb = 0
        for j in range(len(p)):
            pb += p[j]
            tb += t[j]
        # Get the mean of Φ and t
        pb /= len(t)
        tb /= len(t)
        # Formula 1 on my report
        weights = (np.linalg.inv(p.T.dot(p))).dot(p.T).dot(t)
        # Formula 2 on my report
        wpb = 0
```

```
        for j in range(3):
            wpb += weights[j] * pb[j]
        w0 = tb - wpb
        # Formula 3 on my report
        e = 0
        for n in range(len(p)):
            wp = weights.T.dot(p[n])
            e += (t[n] - w0 - wp) ** 2
        e /= 2
        errors.append(e)

    print('\nThe errors of UEMP, MAN, LIC, GR, NMAN, GOV, HE are listed below:')
    print(errors)
    print('(The one with the smallest error is the third variable)')
    print('And the plot is as shown.')
    # Plot
    plt.plot(range(1, 8), errors, linestyle = '-', color = 'green', marker = 'o')
    plt.ylabel('Error')
    plt.show()

main()
```

2.(b) i): Impute Missing Values

```
import pandas as pd

def main(filepath):
    # Load files and create DataFrame
    df = pd.read_csv(filepath, sep=',', header=None)
    training = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.training', sep=',',
header=None)
    testing = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.testing', sep=',',
header=None)

    # Combine training file and testing file for better accuracy
    frames = [training, testing]
    result = pd.concat(frames)

    # For text features, replace "?" with modes
    num=[0, 3, 4, 5, 6, 8, 9, 11, 12]
    for i in num:
        df[i].loc[df[i] == '?'] = result[i].loc[result[i] != '?'].mode()[0]
        print('Replaced all "?" of feature ' + str(i+1) + ' with ' +
str(result[i].loc[result[i] != '?'].mode()[0]))

    # Break the dataset into 2 parts ("+" and "-")
    df1 = result.loc[result[15] == '+']
    df2 = result.loc[result[15] == '-']

    # Change feature 2 and 14 to numerical
    df[1].loc[df[1]!='?'] = pd.to_numeric(df[1].loc[df[1]!='?'])
    df[13].loc[df[13] != '?'] = pd.to_numeric(df[13].loc[df[13] != '?'])

    # Replace "?" with mean for real-valued features
    df[1].loc[(df[1] == '?') & (df[15] == '+')] = pd.to_numeric(df1[1], errors='coerce').mean()
    df[2].loc[(df[2] == '?') & (df[15] == '+')] = df1[2].loc[df1[2]!= '?'].mean()
    df[7].loc[(df[7] == '?') & (df[15] == '+')] = df1[7].loc[df1[7]!= '?'].mean()
    df[10].loc[(df[10] == '?') & (df[15] == '+')] = df1[10].loc[df1[10]!= '?'].mean()
    df[13].loc[(df[13] == '?') & (df[15] == '+')] = pd.to_numeric(df1[13],
errors='coerce').mean()
    df[14].loc[(df[14] == '?') & (df[15] == '+')] = df1[14].loc[df1[14]!= '?'].mean()
    # Ones with "-"
    df[1].loc[(df[1] == '?') & (df[15] == '-')] = pd.to_numeric(df2[1], errors='coerce').mean()
```

```
    df[2].loc[(df[2] == '?') & (df[15] == '-')] = df2[2].loc[df2[2]!= '?'].mean()
    df[7].loc[(df[7] == '?') & (df[15] == '-')] = df2[7].loc[df2[7]!= '?'].mean()
    df[10].loc[(df[10] == '?') & (df[15] == '-')] = df2[10].loc[df2[10]!= '?'].mean()
    df[13].loc[(df[13] == '?') & (df[15] == '-')] = pd.to_numeric(df2[13],
errors='coerce').mean()
    df[14].loc[(df[14] == '?') & (df[15] == '-')] = df2[14].loc[df2[14]!= '?'].mean()

    # Print action
    print('Replaced all "?" in "+"labelled feature 2 with ' + str(pd.to_numeric(df1[1],
errors='coerce').mean()))
    print('Replaced all "?" in "+"labelled feature 3 with ' + str(df1[2].loc[df1[2]!=
'?'].mean()))
    print('Replaced all "?" in "+"labelled feature 8 with ' + str(df1[7].loc[df1[7]!=
'?'].mean()))
    print('Replaced all "?" in "+"labelled feature 11 with ' + str(df1[10].loc[df1[10]!=
'?'].mean()))
    print('Replaced all "?" in "+"labelled feature 14 with ' + str(pd.to_numeric(df1[13],
errors='coerce').mean()))
    print('Replaced all "?" in "+"labelled feature 15 with ' + str(df1[14].loc[df1[14] !=
'?'].mean()))

    print('Replaced all "?" in "-"labelled feature 2 with ' + str(pd.to_numeric(df2[1],
errors='coerce').mean()))
    print('Replaced all "?" in "-"labelled feature 3 with ' + str(df2[2].loc[df2[2] !=
'?'].mean()))
    print('Replaced all "?" in "-"labelled feature 8 with ' + str(df2[7].loc[df2[7] !=
'?'].mean()))
    print('Replaced all "?" in "-"labelled feature 11 with ' + str(df2[10].loc[df2[10] !=
'?'].mean()))
    print('Replaced all "?" in "-"labelled feature 14 with ' + str(pd.to_numeric(df2[13],
errors='coerce').mean()))
    print('Replaced all "?" in "-"labelled feature 15 with ' + str(df2[14].loc[df2[14] !=
'?'].mean()))

    # Save file
    p = filepath + '.processed'
    df.to_csv(p,index=False,header=False)
    print('\nSuccess!\n\nProcessed data saved to '+p)

main('/Users/qiujingye/Downloads/credit 2019/crx.data.training')
main('/Users/qiujingye/Downloads/credit 2019/crx.data.testing')
```

2.(b) ii): k-NN Algorithm

```
import pandas as pd
from math import sqrt

def main():
    # Load files
    df0 = pd.read_csv('/Users/qiujingye/Downloads/credit
2019/crx.data.training.processed', sep=',', header=None)
    df1 = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.testing.processed',
sep=',', header=None)
    # Combine for better accuracy
    frames = [df0, df1]
    result = pd.concat(frames)
    std = result.std()     # Standard deviation for z-scaling

    d = pd.DataFrame(index = [], columns = range(len(df1)))    # Create distance DataFrame

    # Sort
    values = [1, 2, 7, 10, 13, 14]
    strings = [0, 3, 4, 5, 6, 8, 9, 11, 12]
    # Calculate distance
    for i in range(len(df1)):
        temp = []
        for k in range(len(df0)):
            terms = 0
            for j in values:
                terms += ((df0[j][k] - df1[j][i])/std.loc[j]) ** 2
            for l in strings:
                if df0[l][k] != df1[l][i]:
                    terms += 1
            temp.append(sqrt(terms))
        d[i] =  temp
        print('\r' + str(i+1) + '/' + str(len(df1)) + ' finished',end='')
    # Save distance data
    d.to_csv('/Users/qiujingye/Downloads/credit 2019/Distance', index=False, header=False)

    print('\n\nDistance data saved.\n')

    # Calculate accuracy
    v = []
```

```
    for k in range(1, len(df1)):
        check = 0
        for i in range(len(df1)):

            num = d.nsmallest(k, i)
            temp = []
            for j in num.index.values:
                temp.append(df0[15][j])
            result = max(temp, key=temp.count)
            if result == df1[15][i]:
                check += 1
            accuracy = check / len(df1)
        v.append(accuracy)

        print('When k = ' + str(k) + ', the accuracy is ' + str(accuracy))
    v = pd.DataFrame(v)
    best = v[0].idxmax() + 1

    print('\nThe best k value is ' + str(best))

    # Save additional labelled data
    sign = []
    for i in range(len(df1)):
        num = d.nsmallest(k, i)
        temp = []
        for j in num.index.values:
            temp.append(df0[15][j])
        sign.append(max(temp, key=temp.count))
    df1[16] = sign

    df1.to_csv('/Users/qiujingye/Downloads/credit 2019/LabelledTesting', index=False,
header=False)

    print('\nLabelled testing data saved.')

main()
```

2.(b) iii): k-NN Accuracy

```
import pandas as pd

def main():

    # Load files
    d = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/Distance', sep=',',
header=None)
    df0 = pd.read_csv('/Users/qiujingye/Downloads/credit
2019/crx.data.training.processed', sep=',', header=None)
    df1 = pd.read_csv('/Users/qiujingye/Downloads/credit 2019/crx.data.testing.processed',
sep=',', header=None)

    # Calculate accuracy
    v = []
    for k in range(1, len(df1)):
        check = 0
        for i in range(len(df1)):
            num = d.nsmallest(k, i)
            temp = []
            for j in num.index.values:
                temp.append(df0[15][j])
            result = max(temp, key=temp.count)
            if result == df1[15][i]:
                check += 1
            accuracy = check / len(df1)
        v.append(accuracy)

        print('When k = ' + str(k) + ', the accuracy is ' + str(accuracy))
    v = pd.DataFrame(v)
    best = v[0].idxmax() + 1

    print('\nThe best k value is ' + str(best))

    # Save additional labelled data
    sign = []
    for i in range(len(df1)):
        num = d.nsmallest(best, i)
        temp = []
        for j in num.index.values:
```

```
            temp.append(df0[15][j])
        sign.append(max(temp, key=temp.count))
    df1[16] = sign

    df1.to_csv('/Users/qiujingye/Downloads/credit 2019/LabelledTesting', index=False,
header=False)

    print('\nLabelled testing data saved.')

main()
```