# Comparing Distance Based Classifiers

## Submitted By Harsh Srivastava

## 117CS0755

### importing libraries

In [1]:

```
import numpy as np
import pandas as pd
%pylab inline
import matplotlib.pyplot as plt
from sklearn.preprocessing import normalize
```

```
Populating the interactive namespace from numpy and matplotlib
```

### function to calculate euclidean distance between two vectors

In [2]:

```
def euclidean(a, b) :
    dist = a - b
    sq_dist = np.dot(np.transpose(dist), dist)
    sq_dist = np.sqrt(sq_dist)
    return sq_dist
```

### function to calculate city block distance between two vectors

In [3]:

```
def city_block(a, b) :
    dist = np.abs(a - b)
    return np.sum(dist)
```

### function to calculate chess board distance between two vectors

In [4]:

```
def chess_board(a, b) :
    dist = np.abs(a - b)
    return max(dist)
```

### function to calculate cosine distance between two vectors

In [5]:

```
def cos_dist(a, b) :
    dot_product = np.dot(a, b.T)
    norm_a = np.linalg.norm(a)
    norm_b = np.linalg.norm(b)
    return 1 - (dot_product / (norm_a * norm_b))
```

## function to calculate bray curtis distance between two vectors

In [6]:

```
def bray_curtis(a, b) :
    d1 = np.sum(np.abs(a - b))
    d2 = np.sum(np.abs(a + b))
    return d1 / d2
```

## function to calculate canberra distance between two vectors

In [7]:

```
def canberra(a, b) :
    dist = np.abs(a - b) / (np.abs(a) + np.abs(b))
    return np.sum(dist)
```

## function to calculate mahalonobis distance between two vectors

In [8]:

```
def mahalonobis(a, b, input_space) :
    cov = np.cov(input_space.T)
    cov_inv = np.linalg.inv(cov)
    diff = a - b
    dist = np.dot(np.dot(diff.T, cov_inv), diff)
    return dist
```

## function to calculate correlation distance between two vectors

In [9]:

```
def correlation(a, b) :
    dev_a = a - np.mean(a)
    dev_b = b - np.mean(b)
    norm_a = np.linalg.norm(dev_a)
    norm_b = np.linalg.norm(dev_b)
    dist = 1 - (np.dot(dev_a, dev_b.T) / (norm_a * norm_b))
    return dist
```

## function to calculate minkowski distance between two vectors

In [10]:

```
def minkowski(a, b, p) :
    diff = np.abs(a - b)
    dist = pow(np.sum(pow(diff, p)), (1 / p))
    return dist
```

## fucntion to select a distance type

In [11]:

```
def distance(a, b, input_space, dist_type = 0) :

    if dist_type == 0 :
        return euclidean(a, b)

    elif dist_type == 1 :
        return city_block(a, b)

    elif dist_type == 2 :
        return chess_board(a, b)

    elif dist_type == 3 :
        return cos_dist(a, b)

    elif dist_type == 4 :
        return bray_curtis(a, b)

    elif dist_type == 5:
        return canberra(a, b)

    elif dist_type == 6:
        return mahalonobis(a, b, input_space)

    elif dist_type == 7:
        return correlation(a, b)

    elif dist_type == 8:
        return minkowski(a, b, np.random.randint(1, 10))
```

## distance types dictionary

In [12]:

```
dist_dict = {0: 'Euclidean',
             1: 'City Block',
             2: 'Chess Board',
             3: 'Cosine',
             4: 'Bray Curtis',
             5: 'Canberra',
             6: 'Mahalonobis',
             7: 'Correlation',
             8: 'Minkowski'}

dist_nums = len(dist_dict)
```

## splitting dataset into 3 classes

In [13]:

```
dataset = pd.read_csv('IRIS.csv')

dataset_class1 = dataset.loc[0:49, :]
dataset_class2 = dataset.loc[50:99, :]
dataset_class3 = dataset.loc[100:149, :]
```

## Iris-setosa

In [14]:

```
dataset_class1
```

Out[14]:

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **0** | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |
| **5** | 5.4 | 3.9 | 1.7 | 0.4 | Iris-setosa |
| **6** | 4.6 | 3.4 | 1.4 | 0.3 | Iris-setosa |
| **7** | 5.0 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| **8** | 4.4 | 2.9 | 1.4 | 0.2 | Iris-setosa |
| **9** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **10** | 5.4 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| **11** | 4.8 | 3.4 | 1.6 | 0.2 | Iris-setosa |
| **12** | 4.8 | 3.0 | 1.4 | 0.1 | Iris-setosa |
| **13** | 4.3 | 3.0 | 1.1 | 0.1 | Iris-setosa |
| **14** | 5.8 | 4.0 | 1.2 | 0.2 | Iris-setosa |
| **15** | 5.7 | 4.4 | 1.5 | 0.4 | Iris-setosa |
| **16** | 5.4 | 3.9 | 1.3 | 0.4 | Iris-setosa |
| **17** | 5.1 | 3.5 | 1.4 | 0.3 | Iris-setosa |
| **18** | 5.7 | 3.8 | 1.7 | 0.3 | Iris-setosa |
| **19** | 5.1 | 3.8 | 1.5 | 0.3 | Iris-setosa |
| **20** | 5.4 | 3.4 | 1.7 | 0.2 | Iris-setosa |
| **21** | 5.1 | 3.7 | 1.5 | 0.4 | Iris-setosa |
| **22** | 4.6 | 3.6 | 1.0 | 0.2 | Iris-setosa |
| **23** | 5.1 | 3.3 | 1.7 | 0.5 | Iris-setosa |
| **24** | 4.8 | 3.4 | 1.9 | 0.2 | Iris-setosa |
| **25** | 5.0 | 3.0 | 1.6 | 0.2 | Iris-setosa |
| **26** | 5.0 | 3.4 | 1.6 | 0.4 | Iris-setosa |
| **27** | 5.2 | 3.5 | 1.5 | 0.2 | Iris-setosa |
| **28** | 5.2 | 3.4 | 1.4 | 0.2 | Iris-setosa |
| **29** | 4.7 | 3.2 | 1.6 | 0.2 | Iris-setosa |
| **30** | 4.8 | 3.1 | 1.6 | 0.2 | Iris-setosa |
| **31** | 5.4 | 3.4 | 1.5 | 0.4 | Iris-setosa |
| **32** | 5.2 | 4.1 | 1.5 | 0.1 | Iris-setosa |
| **33** | 5.5 | 4.2 | 1.4 | 0.2 | Iris-setosa |

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **34** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **35** | 5.0 | 3.2 | 1.2 | 0.2 | Iris-setosa |
| **36** | 5.5 | 3.5 | 1.3 | 0.2 | Iris-setosa |
| **37** | 4.9 | 3.1 | 1.5 | 0.1 | Iris-setosa |
| **38** | 4.4 | 3.0 | 1.3 | 0.2 | Iris-setosa |
| **39** | 5.1 | 3.4 | 1.5 | 0.2 | Iris-setosa |
| **40** | 5.0 | 3.5 | 1.3 | 0.3 | Iris-setosa |
| **41** | 4.5 | 2.3 | 1.3 | 0.3 | Iris-setosa |
| **42** | 4.4 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **43** | 5.0 | 3.5 | 1.6 | 0.6 | Iris-setosa |
| **44** | 5.1 | 3.8 | 1.9 | 0.4 | Iris-setosa |
| **45** | 4.8 | 3.0 | 1.4 | 0.3 | Iris-setosa |
| **46** | 5.1 | 3.8 | 1.6 | 0.2 | Iris-setosa |
| **47** | 4.6 | 3.2 | 1.4 | 0.2 | Iris-setosa |
| **48** | 5.3 | 3.7 | 1.5 | 0.2 | Iris-setosa |
| **49** | 5.0 | 3.3 | 1.4 | 0.2 | Iris-setosa |

## Iris-versicolor

In [15]:

```
dataset_class2
```

Out[15]:

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **50** | 7.0 | 3.2 | 4.7 | 1.4 | Iris-versicolor |
| **51** | 6.4 | 3.2 | 4.5 | 1.5 | Iris-versicolor |
| **52** | 6.9 | 3.1 | 4.9 | 1.5 | Iris-versicolor |
| **53** | 5.5 | 2.3 | 4.0 | 1.3 | Iris-versicolor |
| **54** | 6.5 | 2.8 | 4.6 | 1.5 | Iris-versicolor |
| **55** | 5.7 | 2.8 | 4.5 | 1.3 | Iris-versicolor |
| **56** | 6.3 | 3.3 | 4.7 | 1.6 | Iris-versicolor |
| **57** | 4.9 | 2.4 | 3.3 | 1.0 | Iris-versicolor |
| **58** | 6.6 | 2.9 | 4.6 | 1.3 | Iris-versicolor |
| **59** | 5.2 | 2.7 | 3.9 | 1.4 | Iris-versicolor |
| **60** | 5.0 | 2.0 | 3.5 | 1.0 | Iris-versicolor |
| **61** | 5.9 | 3.0 | 4.2 | 1.5 | Iris-versicolor |
| **62** | 6.0 | 2.2 | 4.0 | 1.0 | Iris-versicolor |
| **63** | 6.1 | 2.9 | 4.7 | 1.4 | Iris-versicolor |
| **64** | 5.6 | 2.9 | 3.6 | 1.3 | Iris-versicolor |
| **65** | 6.7 | 3.1 | 4.4 | 1.4 | Iris-versicolor |
| **66** | 5.6 | 3.0 | 4.5 | 1.5 | Iris-versicolor |
| **67** | 5.8 | 2.7 | 4.1 | 1.0 | Iris-versicolor |
| **68** | 6.2 | 2.2 | 4.5 | 1.5 | Iris-versicolor |
| **69** | 5.6 | 2.5 | 3.9 | 1.1 | Iris-versicolor |
| **70** | 5.9 | 3.2 | 4.8 | 1.8 | Iris-versicolor |
| **71** | 6.1 | 2.8 | 4.0 | 1.3 | Iris-versicolor |
| **72** | 6.3 | 2.5 | 4.9 | 1.5 | Iris-versicolor |
| **73** | 6.1 | 2.8 | 4.7 | 1.2 | Iris-versicolor |
| **74** | 6.4 | 2.9 | 4.3 | 1.3 | Iris-versicolor |
| **75** | 6.6 | 3.0 | 4.4 | 1.4 | Iris-versicolor |
| **76** | 6.8 | 2.8 | 4.8 | 1.4 | Iris-versicolor |
| **77** | 6.7 | 3.0 | 5.0 | 1.7 | Iris-versicolor |
| **78** | 6.0 | 2.9 | 4.5 | 1.5 | Iris-versicolor |
| **79** | 5.7 | 2.6 | 3.5 | 1.0 | Iris-versicolor |
| **80** | 5.5 | 2.4 | 3.8 | 1.1 | Iris-versicolor |
| **81** | 5.5 | 2.4 | 3.7 | 1.0 | Iris-versicolor |
| **82** | 5.8 | 2.7 | 3.9 | 1.2 | Iris-versicolor |
| **83** | 6.0 | 2.7 | 5.1 | 1.6 | Iris-versicolor |

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **84** | 5.4 | 3.0 | 4.5 | 1.5 | Iris-versicolor |
| **85** | 6.0 | 3.4 | 4.5 | 1.6 | Iris-versicolor |
| **86** | 6.7 | 3.1 | 4.7 | 1.5 | Iris-versicolor |
| **87** | 6.3 | 2.3 | 4.4 | 1.3 | Iris-versicolor |
| **88** | 5.6 | 3.0 | 4.1 | 1.3 | Iris-versicolor |
| **89** | 5.5 | 2.5 | 4.0 | 1.3 | Iris-versicolor |
| **90** | 5.5 | 2.6 | 4.4 | 1.2 | Iris-versicolor |
| **91** | 6.1 | 3.0 | 4.6 | 1.4 | Iris-versicolor |
| **92** | 5.8 | 2.6 | 4.0 | 1.2 | Iris-versicolor |
| **93** | 5.0 | 2.3 | 3.3 | 1.0 | Iris-versicolor |
| **94** | 5.6 | 2.7 | 4.2 | 1.3 | Iris-versicolor |
| **95** | 5.7 | 3.0 | 4.2 | 1.2 | Iris-versicolor |
| **96** | 5.7 | 2.9 | 4.2 | 1.3 | Iris-versicolor |
| **97** | 6.2 | 2.9 | 4.3 | 1.3 | Iris-versicolor |
| **98** | 5.1 | 2.5 | 3.0 | 1.1 | Iris-versicolor |
| **99** | 5.7 | 2.8 | 4.1 | 1.3 | Iris-versicolor |

## Iris-virginica

In [16]:

```
dataset_class3
```

Out[16]:

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **100** | 6.3 | 3.3 | 6.0 | 2.5 | Iris-virginica |
| **101** | 5.8 | 2.7 | 5.1 | 1.9 | Iris-virginica |
| **102** | 7.1 | 3.0 | 5.9 | 2.1 | Iris-virginica |
| **103** | 6.3 | 2.9 | 5.6 | 1.8 | Iris-virginica |
| **104** | 6.5 | 3.0 | 5.8 | 2.2 | Iris-virginica |
| **105** | 7.6 | 3.0 | 6.6 | 2.1 | Iris-virginica |
| **106** | 4.9 | 2.5 | 4.5 | 1.7 | Iris-virginica |
| **107** | 7.3 | 2.9 | 6.3 | 1.8 | Iris-virginica |
| **108** | 6.7 | 2.5 | 5.8 | 1.8 | Iris-virginica |
| **109** | 7.2 | 3.6 | 6.1 | 2.5 | Iris-virginica |
| **110** | 6.5 | 3.2 | 5.1 | 2.0 | Iris-virginica |
| **111** | 6.4 | 2.7 | 5.3 | 1.9 | Iris-virginica |
| **112** | 6.8 | 3.0 | 5.5 | 2.1 | Iris-virginica |
| **113** | 5.7 | 2.5 | 5.0 | 2.0 | Iris-virginica |
| **114** | 5.8 | 2.8 | 5.1 | 2.4 | Iris-virginica |
| **115** | 6.4 | 3.2 | 5.3 | 2.3 | Iris-virginica |
| **116** | 6.5 | 3.0 | 5.5 | 1.8 | Iris-virginica |
| **117** | 7.7 | 3.8 | 6.7 | 2.2 | Iris-virginica |
| **118** | 7.7 | 2.6 | 6.9 | 2.3 | Iris-virginica |
| **119** | 6.0 | 2.2 | 5.0 | 1.5 | Iris-virginica |
| **120** | 6.9 | 3.2 | 5.7 | 2.3 | Iris-virginica |
| **121** | 5.6 | 2.8 | 4.9 | 2.0 | Iris-virginica |
| **122** | 7.7 | 2.8 | 6.7 | 2.0 | Iris-virginica |
| **123** | 6.3 | 2.7 | 4.9 | 1.8 | Iris-virginica |
| **124** | 6.7 | 3.3 | 5.7 | 2.1 | Iris-virginica |
| **125** | 7.2 | 3.2 | 6.0 | 1.8 | Iris-virginica |
| **126** | 6.2 | 2.8 | 4.8 | 1.8 | Iris-virginica |
| **127** | 6.1 | 3.0 | 4.9 | 1.8 | Iris-virginica |
| **128** | 6.4 | 2.8 | 5.6 | 2.1 | Iris-virginica |
| **129** | 7.2 | 3.0 | 5.8 | 1.6 | Iris-virginica |
| **130** | 7.4 | 2.8 | 6.1 | 1.9 | Iris-virginica |
| **131** | 7.9 | 3.8 | 6.4 | 2.0 | Iris-virginica |
| **132** | 6.4 | 2.8 | 5.6 | 2.2 | Iris-virginica |
| **133** | 6.3 | 2.8 | 5.1 | 1.5 | Iris-virginica |

| | sepal_length | sepal_width | petal_length | petal_width | species |
|---|---|---|---|---|---|
| **134** | 6.1 | 2.6 | 5.6 | 1.4 | Iris-virginica |
| **135** | 7.7 | 3.0 | 6.1 | 2.3 | Iris-virginica |
| **136** | 6.3 | 3.4 | 5.6 | 2.4 | Iris-virginica |
| **137** | 6.4 | 3.1 | 5.5 | 1.8 | Iris-virginica |
| **138** | 6.0 | 3.0 | 4.8 | 1.8 | Iris-virginica |
| **139** | 6.9 | 3.1 | 5.4 | 2.1 | Iris-virginica |
| **140** | 6.7 | 3.1 | 5.6 | 2.4 | Iris-virginica |
| **141** | 6.9 | 3.1 | 5.1 | 2.3 | Iris-virginica |
| **142** | 5.8 | 2.7 | 5.1 | 1.9 | Iris-virginica |
| **143** | 6.8 | 3.2 | 5.9 | 2.3 | Iris-virginica |
| **144** | 6.7 | 3.3 | 5.7 | 2.5 | Iris-virginica |
| **145** | 6.7 | 3.0 | 5.2 | 2.3 | Iris-virginica |
| **146** | 6.3 | 2.5 | 5.0 | 1.9 | Iris-virginica |
| **147** | 6.5 | 3.0 | 5.2 | 2.0 | Iris-virginica |
| **148** | 6.2 | 3.4 | 5.4 | 2.3 | Iris-virginica |
| **149** | 5.9 | 3.0 | 5.1 | 1.8 | Iris-virginica |

## creating a dictionary to map class name with our defined labels

## 'Iris-setosa' --> 0

## 'Iris-versicolor' --> 1

## 'Iris-virginica' --> 2

In [17]:

```
iris_dict = {0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'}
iris_dict
```

Out[17]:

```
{0: 'Iris-setosa', 1: 'Iris-versicolor', 2: 'Iris-virginica'}
```

## splitting features and classes

In [18]:

```
features = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width']

X1 = dataset_class1[features]
X2 = dataset_class2[features]
X3 = dataset_class3[features]

X1 = normalize(X1, 'l2')
X2 = normalize(X2, 'l2')
X3 = normalize(X3, 'l2')
```

## splitting dataset into training and test

In [19]:

```
from sklearn.model_selection import train_test_split

x_train1_pd, x_test1_pd = train_test_split(X1, test_size=.4)
x_train2_pd, x_test2_pd = train_test_split(X2, test_size=.4)
x_train3_pd, x_test3_pd = train_test_split(X3, test_size=.4)

num_test = 20
num_train = 50 - num_test
total_num_test = 60
total_num_train = 150 - total_num_test

x_train1_pd
```

Out[19]:

```
array([[0.82307218, 0.51442011, 0.24006272, 0.01714734],
       [0.81609427, 0.5336001 , 0.21971769, 0.03138824],
       [0.78010936, 0.57660257, 0.23742459, 0.0508767 ],
       [0.86093857, 0.44003527, 0.24871559, 0.0573959 ],
       [0.80597792, 0.52151512, 0.26865931, 0.07901744],
       [0.81803119, 0.51752994, 0.25041771, 0.01669451],
       [0.81120865, 0.55945424, 0.16783627, 0.02797271],
       [0.776114  , 0.54974742, 0.30721179, 0.03233808],
       [0.80373519, 0.55070744, 0.22325977, 0.02976797],
       [0.80212413, 0.54690282, 0.23699122, 0.03646019],
       [0.786991  , 0.55745196, 0.26233033, 0.03279129],
       [0.79837025, 0.55735281, 0.22595384, 0.03012718],
       [0.78889479, 0.55222635, 0.25244633, 0.09466737],
       [0.82225028, 0.51771314, 0.22840286, 0.06090743],
       [0.77867447, 0.59462414, 0.19820805, 0.02831544],
       [0.79524064, 0.54144043, 0.27072022, 0.03384003],
       [0.80642366, 0.5315065 , 0.25658935, 0.03665562],
       [0.76693897, 0.57144472, 0.28572236, 0.06015208],
       [0.77381111, 0.59732787, 0.2036345 , 0.05430253],
       [0.81803119, 0.51752994, 0.25041771, 0.01669451],
       [0.79778206, 0.5424918 , 0.25529026, 0.06382256],
       [0.81228363, 0.5361072 , 0.22743942, 0.03249135],
       [0.77577075, 0.60712493, 0.16864581, 0.03372916],
       [0.78591858, 0.57017622, 0.23115252, 0.06164067],
       [0.82699754, 0.52627116, 0.19547215, 0.03007264],
       [0.80218492, 0.54548574, 0.24065548, 0.0320874 ],
       [0.80641965, 0.54278246, 0.23262105, 0.03101614],
       [0.80033301, 0.56023311, 0.20808658, 0.04801998],
       [0.80779568, 0.53853046, 0.23758697, 0.03167826],
       [0.8068282 , 0.53788547, 0.24063297, 0.04246464]])
```

## converting training and test datasets to numpy arrays for easy calculations

In [20]:

```
x_train = []
x_test = []

x_train.append(x_train1_pd)#.to_numpy())
x_train.append(x_train2_pd)#.to_numpy())
x_train.append(x_train3_pd)#.to_numpy())

x_test.append(x_test1_pd)#.to_numpy())
x_test.append(x_test2_pd)#.to_numpy())
x_test.append(x_test3_pd)#.to_numpy())

print(x_train[0])
print(x_test[0])

print(x_train[0].shape)
print(x_test[0].shape)
```

```
[[0.82307218 0.51442011 0.24006272 0.01714734]
 [0.81609427 0.5336001  0.21971769 0.03138824]
 [0.78010936 0.57660257 0.23742459 0.0508767 ]
 [0.86093857 0.44003527 0.24871559 0.0573959 ]
 [0.80597792 0.52151512 0.26865931 0.07901744]
 [0.81803119 0.51752994 0.25041771 0.01669451]
 [0.81120865 0.55945424 0.16783627 0.02797271]
 [0.776114   0.54974742 0.30721179 0.03233808]
 [0.80373519 0.55070744 0.22325977 0.02976797]
 [0.80212413 0.54690282 0.23699122 0.03646019]
 [0.786991   0.55745196 0.26233033 0.03279129]
 [0.79837025 0.55735281 0.22595384 0.03012718]
 [0.78889479 0.55222635 0.25244633 0.09466737]
 [0.82225028 0.51771314 0.22840286 0.06090743]
 [0.77867447 0.59462414 0.19820805 0.02831544]
 [0.79524064 0.54144043 0.27072022 0.03384003]
 [0.80642366 0.5315065  0.25658935 0.03665562]
 [0.76693897 0.57144472 0.28572236 0.06015208]
 [0.77381111 0.59732787 0.2036345  0.05430253]
 [0.81803119 0.51752994 0.25041771 0.01669451]
 [0.79778206 0.5424918  0.25529026 0.06382256]
 [0.81228363 0.5361072  0.22743942 0.03249135]
 [0.77577075 0.60712493 0.16864581 0.03372916]
 [0.78591858 0.57017622 0.23115252 0.06164067]
 [0.82699754 0.52627116 0.19547215 0.03007264]
 [0.80218492 0.54548574 0.24065548 0.0320874 ]
 [0.80641965 0.54278246 0.23262105 0.03101614]
 [0.80033301 0.56023311 0.20808658 0.04801998]
 [0.80779568 0.53853046 0.23758697 0.03167826]
 [0.8068282  0.53788547 0.24063297 0.04246464]]
[[0.80377277 0.55160877 0.22064351 0.0315205 ]
 [0.790965   0.5694948  0.2214702  0.0316386 ]
 [0.82647451 0.4958847  0.26447184 0.03305898]
 [0.78609038 0.57170209 0.23225397 0.03573138]
 [0.80327412 0.55126656 0.22050662 0.04725142]
 [0.8173379  0.51462016 0.25731008 0.03027177]
 [0.82210585 0.51381615 0.23978087 0.05138162]
 [0.77729093 0.57915795 0.24385598 0.030482  ]
 [0.80846584 0.52213419 0.26948861 0.03368608]
 [0.76578311 0.60379053 0.22089897 0.0147266 ]
```

```
 [0.80533308 0.54831188 0.2227517  0.03426949]
 [0.82512295 0.52807869 0.19802951 0.03300492]
 [0.8025126  0.55989251 0.20529392 0.01866308]
 [0.81803119 0.51752994 0.25041771 0.01669451]
 [0.79428944 0.57365349 0.19121783 0.05883625]
 [0.80003025 0.53915082 0.26087943 0.03478392]
 [0.82813287 0.50702013 0.23660939 0.03380134]
 [0.77964883 0.58091482 0.22930848 0.0458617 ]
 [0.78417499 0.5663486  0.2468699  0.05808704]
 [0.79594782 0.55370283 0.24224499 0.03460643]]
(30, 4)
(20, 4)
```

## calculating mean of features of classes in training dataset

In [21]:

```
mean = []

mean.append(np.mean(x_train[0], axis = 0))
mean.append(np.mean(x_train[1], axis = 0))
mean.append(np.mean(x_train[2], axis = 0))

print(mean[0])
print(mean[1])
print(mean[2])
```

```
[0.80184486 0.54520738 0.23574351 0.04115118]
[0.74726389 0.34832392 0.53835635 0.16803758]
[0.70542075 0.31784356 0.59273109 0.21819911]
```

## predicting classes for iris-setosa test data using euclidean distance

In [22]:

```
def prediction_distance_types(dist_type = 0) :

    correct_count = 0 # variable to count number of training examples correctly classified
    pred = []
    error = []

    for i in range(3) :
        train = x_train[i] # for covariance matrix
        test = x_test[i]

        for j in range(num_test) :
            min_dist = distance(test[j], mean[0], train, dist_type)
            min_idx = 0

            for k in range(len(mean)) :
                dist = distance(test[j], mean[k], train, dist_type)
                if dist < min_dist :
                    min_dist = dist
                    min_idx = k

            pred.append(iris_dict[min_idx])
            error.append(min_dist)

            if min_idx == i :
                correct_count += 1


    MER = 1.0 - (correct_count / total_num_test)

    return MER, correct_count, pred, error
```

## predicting using different distance methods

In [23]:

```
MER_vals = []
correct_count_vals = []
error_vals = []

for i in range(dist_nums) :
    MER, correct_count, pred, error = prediction_distance_types(i)
    print("MER for {} distance = {}".format(dist_dict[i], MER))
    print("Number of Correct classifications out of {} = {}\n".format(total_num_test, corre
    MER_vals.append(MER)
    correct_count_vals.append(correct_count)
    error_vals.append(error)
```

```
MER for Euclidean distance = 0.050000000000000044
Number of Correct classifications out of 60 = 57

MER for City Block distance = 0.050000000000000044
Number of Correct classifications out of 60 = 57

MER for Chess Board distance = 0.050000000000000044
Number of Correct classifications out of 60 = 57

MER for Cosine distance = 0.050000000000000044
Number of Correct classifications out of 60 = 57

MER for Bray Curtis distance = 0.050000000000000044
Number of Correct classifications out of 60 = 57

MER for Canberra distance = 0.06666666666666665
Number of Correct classifications out of 60 = 56

MER for Mahalonobis distance = 0.033333333333333326
Number of Correct classifications out of 60 = 58

MER for Correlation distance = 0.050000000000000044
Number of Correct classifications out of 60 = 57

MER for Minkowski distance = 0.08333333333333337
Number of Correct classifications out of 60 = 55
```

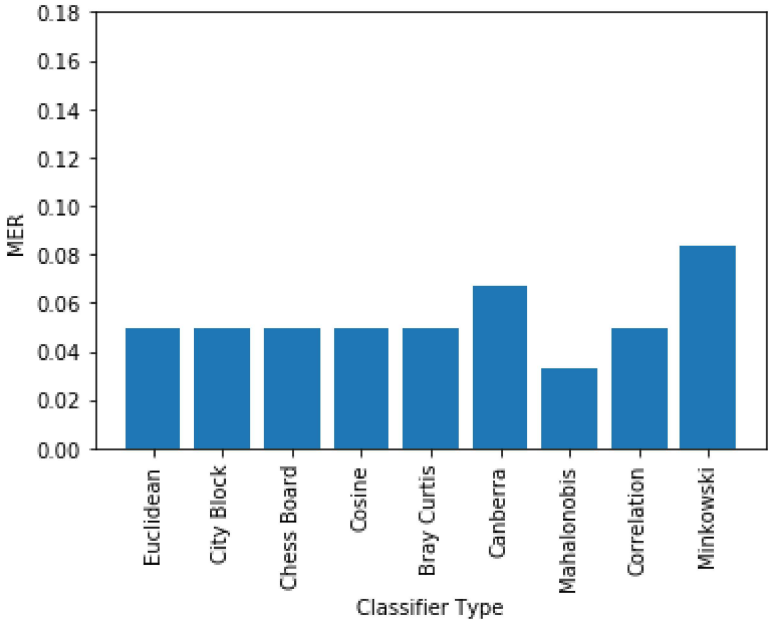
## Plotting MER vs distance type classifier

In [24]:

```
print("Plotting MER vs Classifier Type")
plt.bar(list(dist_dict.values()), MER_vals)
plt.xlabel('Classifier Type')
plt.ylabel('MER')

plt.xticks(rotation=90)
plt.yticks(np.arange(0, 0.2, 0.02))
plt.show()
```

```
Plotting MER vs Classifier Type
```




## Mean Error and plot

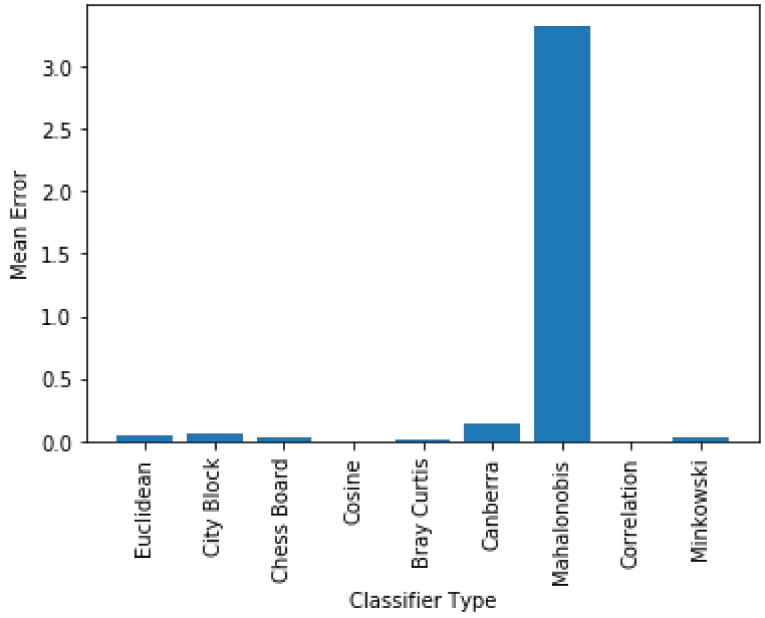
In [25]:

```
error_vals = np.array(error_vals)
mean_error = np.mean(error_vals, axis = 1)

print("Plotting Mean Error vs Classifier Type")
plt.bar(list(dist_dict.values()), mean_error)
plt.xlabel('Classifier Type')
plt.ylabel('Mean Error')

plt.xticks(rotation=90)
plt.show()
```

```
Plotting Mean Error vs Classifier Type
```




## Mean Squared Error and plot

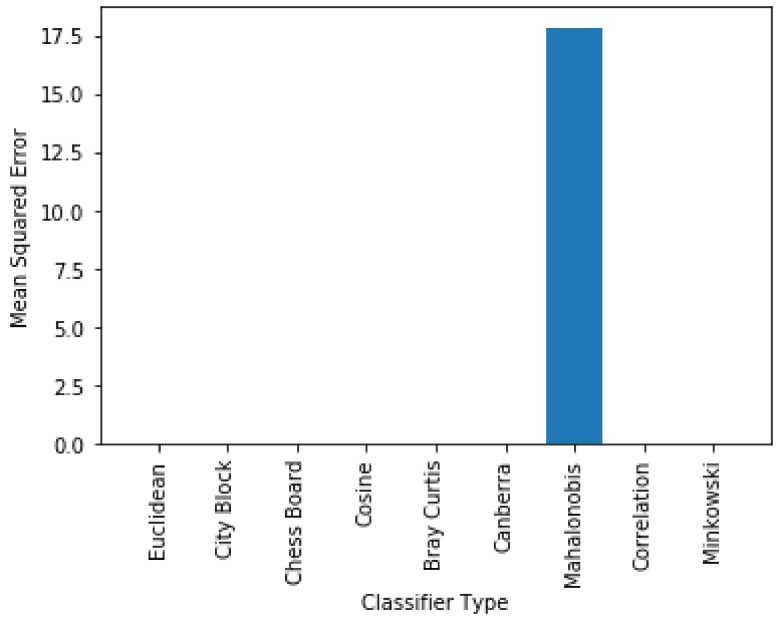
In [26]:

```
mean_squared_error = np.mean(np.square(error_vals), axis = 1)

print("Plotting Mean Squared Error vs Classifier Type")
plt.bar(list(dist_dict.values()), mean_squared_error)
plt.xlabel('Classifier Type')
plt.ylabel('Mean Squared Error')

plt.xticks(rotation=90)
plt.show()
```

```
Plotting Mean Squared Error vs Classifier Type
```




## Mean absolute Error and plot

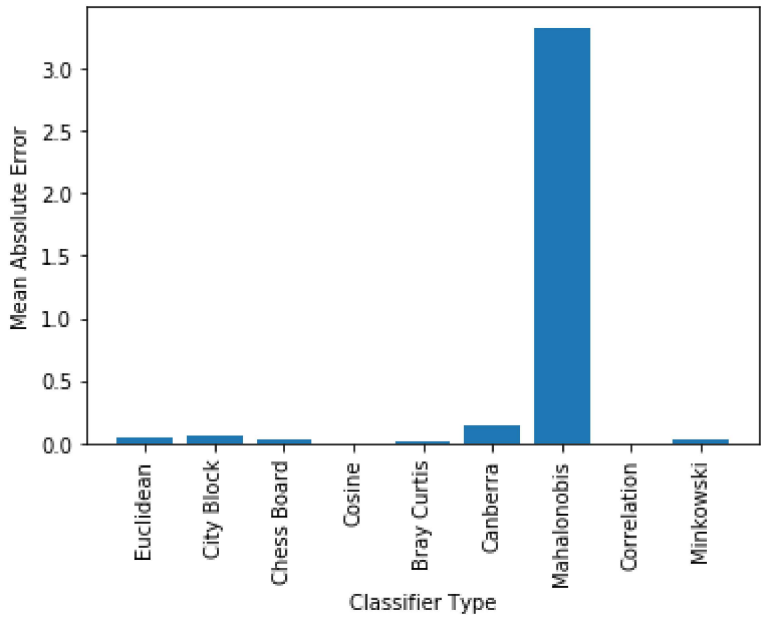
In [27]:

```
mean_absolute_error = np.mean(np.abs(error_vals), axis = 1)

print("Plotting Mean Absolute Error vs Classifier Type")
plt.bar(list(dist_dict.values()), mean_absolute_error)
plt.xlabel('Classifier Type')
plt.ylabel('Mean Absolute Error')

plt.xticks(rotation=90)
plt.show()
```

```
Plotting Mean Absolute Error vs Classifier Type
```




In [ ]: