ECE M148, Homework 5

Damien Ha, UID 905539967

Question 1

Part A

In [1]:

```
import pandas as pd

data = {
    'Day': list(range(1, 15)),
    'Humidity': ['High', 'High', 'Normal', 'Normal', 'High',
'Normal', 'High', 'Normal', 'High', 'High', 'High', 'High', 'High',
'Normal'],
    'Wind': ['Weak', 'Strong', 'Weak', 'Weak', 'Strong', 'Strong',
'Weak', 'Weak', 'Strong', 'Strong', 'Weak', 'Weak', 'Strong',
'Strong'],
    'Play Tennis': ['Yes', 'No', 'Yes', 'Yes', 'No', 'Yes', 'Yes',
'Yes', 'Yes', 'No', 'Yes', 'No', 'No', 'No']
}

df = pd.DataFrame(data)

def calculate_gini_index(groups, classes):
    total_instances = sum(len(group) for group in groups)
    gini_index = 0.0
    for group in groups:
        size = len(group)
        if size == 0:
            continue
        score = 0.0
        for class_val in classes:
            proportion = [row[-1] for row in group].count(class_val)
/ size
            score += proportion * proportion
        gini_index += (1.0 - score) * (size / total_instances)
    return gini_index
```

```
def calculate_gini_index_gain(data, feature):
    current_gini_index = calculate_gini_index([data.values], df['Play
Tennis'].unique())

    unique_values = df[feature].unique()
    feature_index = df.columns.get_loc(feature)

    weighted_gini_index = 0.0
    for value in unique_values:
        subset = data[data[feature] == value]
        subset_gini_index = calculate_gini_index([subset.values],
df['Play Tennis'].unique())
        weighted_gini_index += len(subset) / len(data) *
subset_gini_index

    gini_index_gain = current_gini_index - weighted_gini_index
    return gini_index_gain

gini_index_humidity = calculate_gini_index_gain(df, 'Humidity')
gini_index_wind = calculate_gini_index_gain(df, 'Wind')

print("Gini Index Gain (Humidity):", gini_index_humidity)
print("Gini Index Gain (Wind):", gini_index_wind)
```

```
Gini Index Gain (Humidity): 0.05804988662131538
Gini Index Gain (Wind): 0.16326530612244905
```

Part B

Wind, based on the results above, provides the best Gini Index Gain

Part C

In [2]:

```
import math

def calculate_entropy(data):
    total_instances = len(data)
    class_counts = data['Play Tennis'].value_counts()
    entropy = 0.0
    for count in class_counts:
        proportion = count / total_instances
        entropy -= proportion * math.log2(proportion)
```

```
    return entropy

def calculate_information_gain(data, feature):
    current_entropy = calculate_entropy(data)

    unique_values = df[feature].unique()
    feature_index = df.columns.get_loc(feature)

    weighted_entropy = 0.0
    for value in unique_values:
        subset = data[data[feature] == value]
        subset_entropy = calculate_entropy(subset)
        weighted_entropy += len(subset) / len(data) * subset_entropy

    information_gain = current_entropy - weighted_entropy
    return information_gain

information_gain_humidity = calculate_information_gain(df,
'Humidity')
information_gain_wind = calculate_information_gain(df, 'Wind')

print("Information Gain (Humidity):", information_gain_humidity)
print("Information Gain (Wind):", information_gain_wind)
```

```
Information Gain (Humidity): 0.09027634939276485
Information Gain (Wind): 0.2578314624597723
```

Part D

Wind is still the best feature

Question 2

In [4]:

```
import numpy as np

data = np.array([[1, 1], [2, 4], [4, 1], [6, 3], [5, 7], [8, 1], [4,
4], [3, 6], [3, 3]])
centers = np.array([[2, 2], [5, 5]])

distances = np.zeros((data.shape[0], centers.shape[0]))
for i in range(centers.shape[0]):
    distances[:, i] = np.linalg.norm(data - centers[i], axis=1)
```

```
cluster_assignments = np.argmin(distances, axis=1)
print("Cluster Assignments (after one iteration):",
cluster_assignments)

new_centers = np.zeros_like(centers)
for i in range(centers.shape[0]):
    new_centers[i] = np.mean(data[cluster_assignments == i], axis=0)

print("New Cluster Centers (after one iteration):\n", new_centers)
```

```
Cluster Assignments (after one iteration): [0 0 0 1 1 1 1 1 0]
New Cluster Centers (after one iteration):
 [[2 2]
 [5 4]]
```

Question 3

Part A

The circled points appear to be the support vectors as we can visually see that the circled points are where + and - points are seperated. Removing one of the non-circled points therefore shouldn't have any effect on the decision boundary, none of them are close to it and the support vectors would remain.

Part B

A hard margin SVM seeks a perfect separation of classes and assumes they can be seperated linearly, while a soft margin SVM allows for some misclassifications and includes a margin of error to handle overlapping or noisy data points

Part C

5, the remaining 3 +'s would become support vectors as they'd be equally close to the decision boundary, and the orginal 2 -'s would remain giving us 5

Question 4

Part A

For the first expression:

$$P(Y=i \mid X)=\frac{e^{\beta_{0i}+\beta_{1i}X}}{1+\sum_{j=1}^{K-1} e^{\beta_{0i}+\beta_{1i}X}}, 1 \leq i \leq K-1$$

$$=\frac{e^{\beta_{0i}+\beta_{1i}X} e^{\beta_{0K}+\beta_{1K}X}}{e^{\beta_{0K}+\beta_{1K}X}+\sum_{j=1}^{K-1} e^{\beta_{0i}+\beta_{1i}X}}$$

$$= e^{\beta_{0i}+\beta_{1i}X} \cdot \frac{e^{\beta_{0K}+\beta_{1K}X}}{e^{\beta_{0K}+\beta_{1K}X}+\sum_{j=1}^{K-1} e^{\beta_{0i}+\beta_{1i}X}}$$

$$= e^{\beta_{0i}+\beta_{1i}X} \cdot a\ constant$$

For the second expression:

$$P(Y = i \mid X) = \frac{e^{\hat{\beta_{0i}}+\hat{\beta_{1i}}X}}{1+\sum_{j=1}^{K-1} e^{\hat{\beta_{0i}}+\hat{\beta_{1i}}X}}, 1 \leq i \leq K$$

$$= \frac{e^{\hat{\beta_{0i}}+\hat{\beta_{1i}}X}}{e^{\hat{\beta_{0K}}+\hat{\beta_{1K}}X \cdot C}}$$

Subbing in and canceling terms we get

$$= \frac{e^{\hat{\beta_{0i}}+\hat{\beta_{1i}}X} e^{\beta_{0i}+\beta_{1i}X}}{e^{\hat{\beta_{0K}}+\hat{\beta_{1K}}X}}$$

$$= e^{\hat{\beta_{0i}}+\hat{\beta_{1i}}X} e^{\beta_{0i}+\beta_{1i}X}$$

$$= e^{\hat{\beta_{0i}}+\hat{\beta_{1i}}X+\beta_{0i}+\beta_{1i}X}$$

$$= e^{\beta_{0i}+\beta_{1i}X}$$

The two are equivalent

Part B

$$P(Y = 1 \mid X) = \frac{e^{-0.2+0.06 \cdot 5}}{1+e^{-0.2+0.06 \cdot 5}+e^{-0.2+0.04 \cdot 5}} = 0.146$$

$$P(Y = 2 \mid X) = \frac{e^{0.2+0.04 \cdot 5}}{1+e^{-0.2+0.06 \cdot 5}+e^{-0.2+0.04 \cdot 5}} = 0.252$$

$$P(Y = 3 \mid X) = \frac{e^{-0.3+0.5 \cdot 5}}{1+e^{-0.2+0.06 \cdot 5}+e^{-0.2+0.04 \cdot 5}} = 0.602$$

Class 3, as it has the highest probability

Question 5

Part A

False, we would actually want to pick the feature whose split minimizes MSE, so there is a lower error

Part B

False, K-means is sensitive to the initial points/centroids, so different initializations can lead to drastically different clusters and different solutions

Part C

False, its goal is to combine similar clusters and then create a hierarchy of clusters which is not necessarily minimizing distortion

Part D

False, larger λ implies a narrower margin due to a higher penalty

Part E

True, by bootstrapping and considering a subset of features for each node, we reduce the variance and generalize the performance, avoiding overfitting