HW-Topic-3

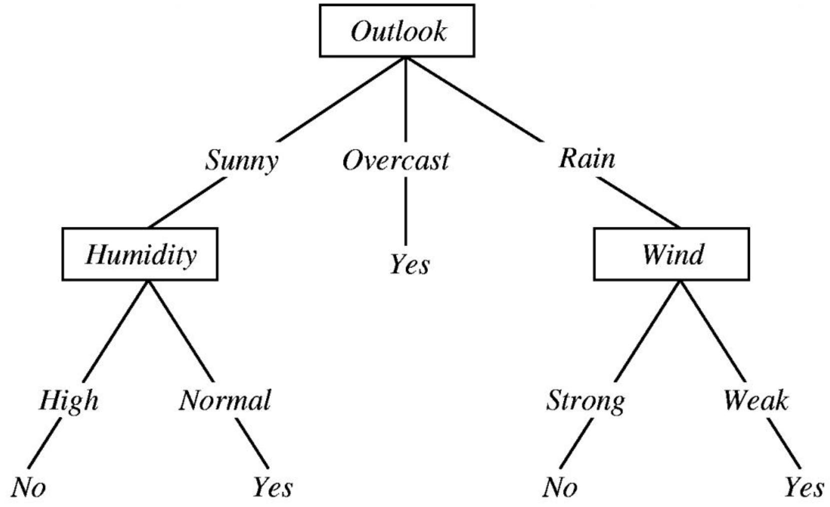
Data Acquisition, Modeling and Analysis: Big Data Analytics

Submitted By – **Sudhanshu Kakkar**
CWID – **20036779**

DLT – Decision Tree Learning

WHAT IS IT?

- A **supervised machine learning algorithm** used for **classification and regression**.
- Represents decisions as a **tree structure.**



KEY CONCEPTS

- **Splitting:** Dividing dataset based on feature values.
- **Entropy:** Measure of randomness or impurity.
- **Information Gain:** Reduction in entropy after a split.
- **Pruning:** Removing branches to reduce overfitting.

TYPES

- **Classification Tree:** Predicts discrete labels (e.g., Yes/No)
- **Regression Tree:** Predicts continuous values

Pros	Cons
• Easy to understand & visualize • Handles categorical & numerical data	• Prone to overfitting • Sensitive to small changes in data

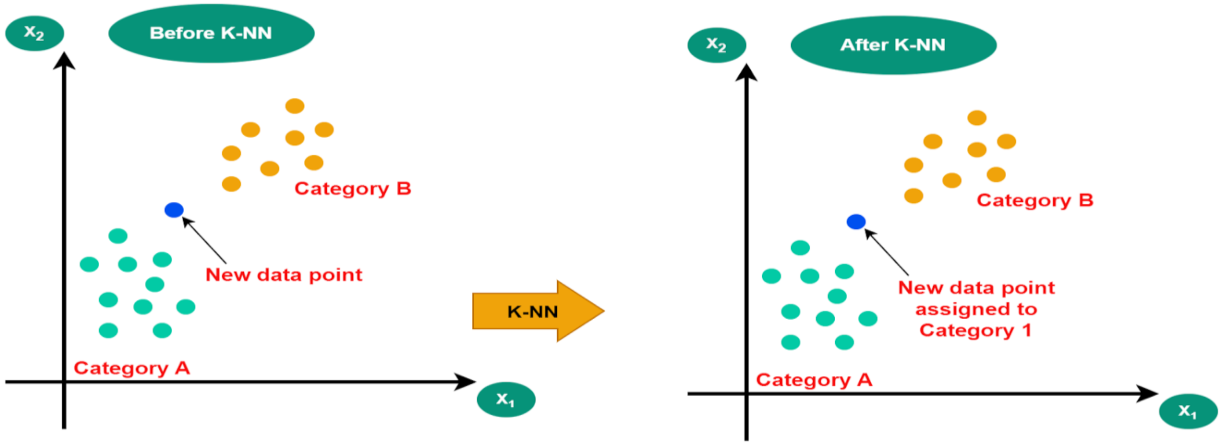
KNN – K-Nearest Neighbours

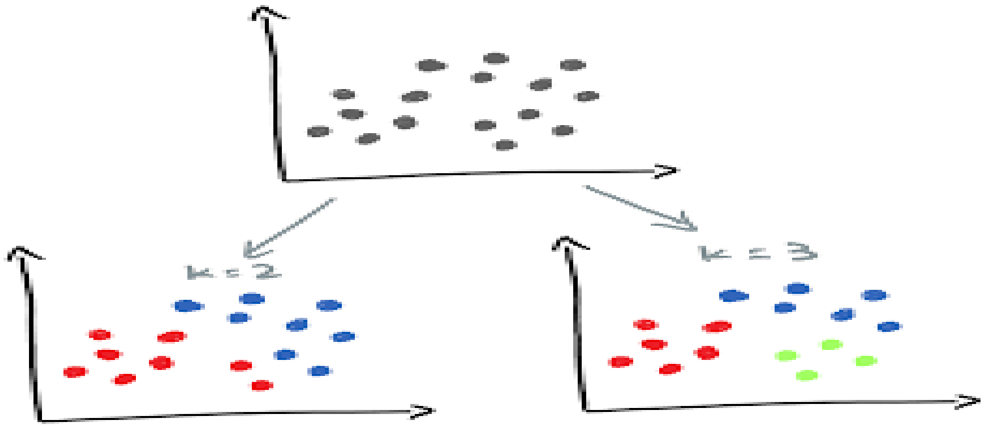
WHAT IS IT?

- A **supervised machine learning algorithm** used for **classification and regression**.
- Classifies a data point based on the **majority class of its K nearest neighbors**

KEY CONCEPTS

- **K Value:** Number of neighbors considered
- **Distance Metrics:** How similarity is measured
  - **Euclidean Distance**
  - **Manhattan Distance**





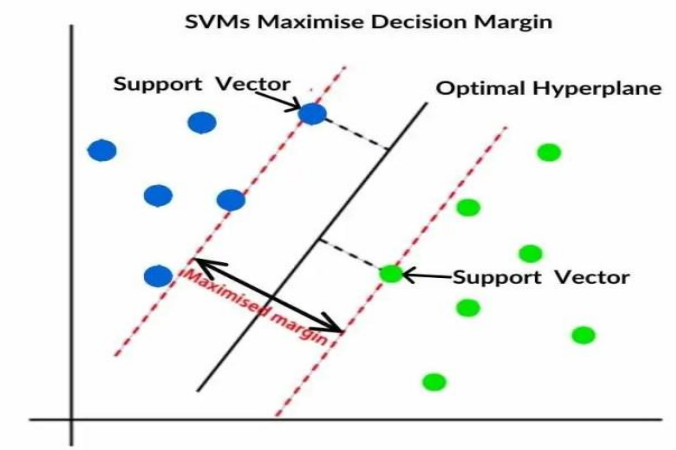
Pros	Cons
• Simple and easy to understand • Works well with small datasets	• Computationally expensive for large datasets • Poor performance on high-dimensional data

SVM – Support Vector Machine

WHAT IS IT?

- A **supervised machine learning algorithm** used for **classification and regression**.
- Finds the **best hyperplane** that separates classes with the **maximum margin**.
- Works well for **linear and non-linear** data using kernel functions.

KEY CONCEPTS



- **Hyperplane:** Decision boundary separating classes
- **Support Vectors:** Data points closest to the hyperplane
- **Margin:** Distance between support vectors of different classes (maximizing this improves generalization)

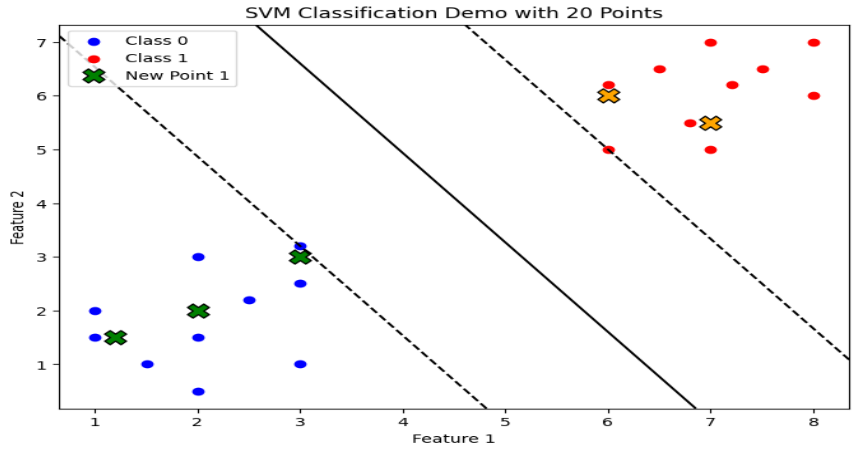
TYPES

- **Linear SVM:** Data is linearly separable
- **Non-linear SVM:** Uses kernels for complex datasets

Pros	Cons
• Effective in high-dimensional spaces • Works well for linearly and non-linearly separable data	• Computationally expensive for very large datasets • Choosing the right kernel can be tricky

SVM Code Demo (python)

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

# declaring points for training
X = np.array([
    [1,2], [2,3], [3,1], [2,1.5], [3,2.5],
    [1.5,1], [2.5,2.2], [3,3.2], [2,0.5], [1,1.5],
    [6,5], [7,7], [8,6], [7,5], [6.5,6.5],
    [8,7], [7.2,6.2], [6.8,5.5], [7.5,6.5], [6,6.2]
])
# declaring labels for points
y = np.array([0]*10 + [1]*10)  # 0 = Class A, 1 = Class B

# Train SVM
model = svm.SVC(kernel='linear')
model.fit(X, y)

# New points to classify
new_points = np.array([
    [2,2], [3,3], [6,6], [7,5.5], [1.2,1.5]
])
predictions = model.predict(new_points)

# Print classifications
for point, label in zip(new_points, predictions):
    print(f"Point {point} is classified as Class {label}")

# Plot
plt.figure(figsize=(8,6))

# Plot original points
plt.scatter(X[:10,0], X[:10,1], color='blue', label='Class
0')
plt.scatter(X[10:,0], X[10:,1], color='red', label='Class
1')
```




```python
# Plot new points with classification
for i, point in enumerate(new_points):
    plt.scatter(point[0], point[1], color='green' if predictions[i]==0 else 'orange',
                edgecolors='k', s=150, marker='X', label=f'New Point {i+1}' if i==0 else
"")


# Plot decision boundary
ax = plt.gca()
xlim = ax.get_xlim()
ylim = ax.get_ylim()
xx = np.linspace(xlim[0], xlim[1], 50)
yy = np.linspace(ylim[0], ylim[1], 50)
YY, XX = np.meshgrid(yy, xx)
Z = model.decision_function(np.c_[XX.ravel(), YY.ravel()]).reshape(XX.shape)
ax.contour(XX, YY, Z, levels=[0], colors='k')      # hyperplane
ax.contour(XX, YY, Z, levels=[-1,1], colors='k', linestyles='--')  # margins


plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.title("SVM Classification Demo with 20 Points")
plt.legend()
plt.show()
```