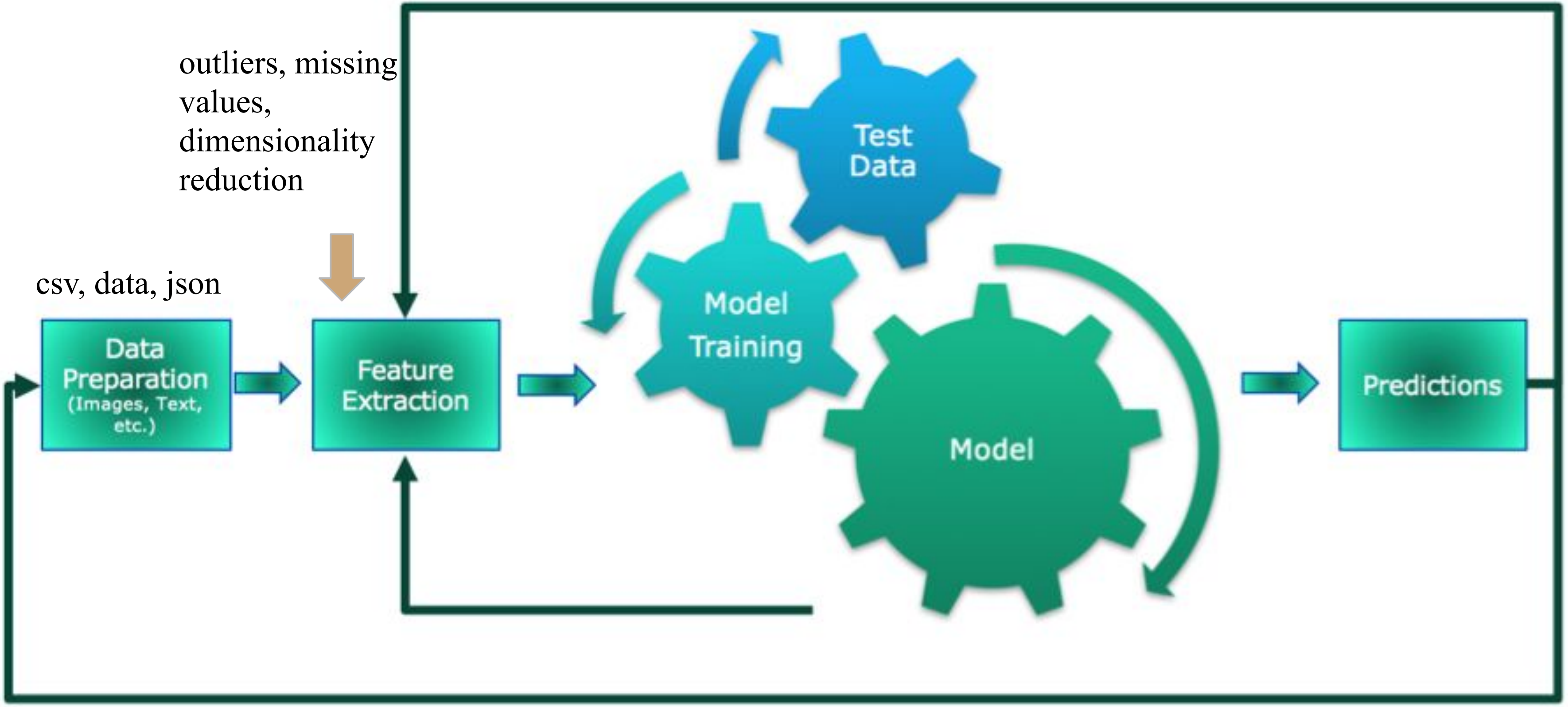
Machine Learning Algorithms

Topgyal Gurung

A Standard Machine Learning Pipeline



Types of machine Learning

1. Supervised
2. Unsupervised
3. Semi-supervised and
4. Reinforcement

Algorithms

1. **K Nearest Neighbor** : supervised
2. **Decision Tree** : supervised
3. **K means** : unsupervised

Predicting **Regression** (**(**real number) or **Classification (**category)

Project Plan

1. Research on algorithms
2. Built it from scratch using python with minimum use of library
3. Implement on datasets from UCI Machine Learning Repository
4. Test and evaluate algorithms

UCI Machine Learning Dataset

1. **Breast Cancer**:
   - Benign or Malignant cancer (binary classification)
   - KNN and KMeans
2. **Monk's Problem:**
   - Which of three problems most difficult for a decision tree algorithm to learn?
   - Binary classification problem (class 0 or 1)
   - Monk 1, Monk 2, Monk 3 (5% misclassified)
   - Decision Tree

Attributes description

An artificial robot domain described by ***six attributes***

a1: head_shape =round, square or octagon (1,2,3)

a2: body_shape=round,square or octagon (1,2,3)

a3: is_smiling = yes or no (1,2)

a4: holding= sword, balloon, flag (1,2,3)

a5: jacket_color=red,yellow, green, blue

a6: has_tie=yes or no

ML Concepts

Normalization: actual value into standard range of values [-1,1] or [0,1]

min-max formula $$\bar{x}^{j} = \frac{x^{j} - min^{j}}{max^{j} - min^{j}}$$ to ensure equal weight

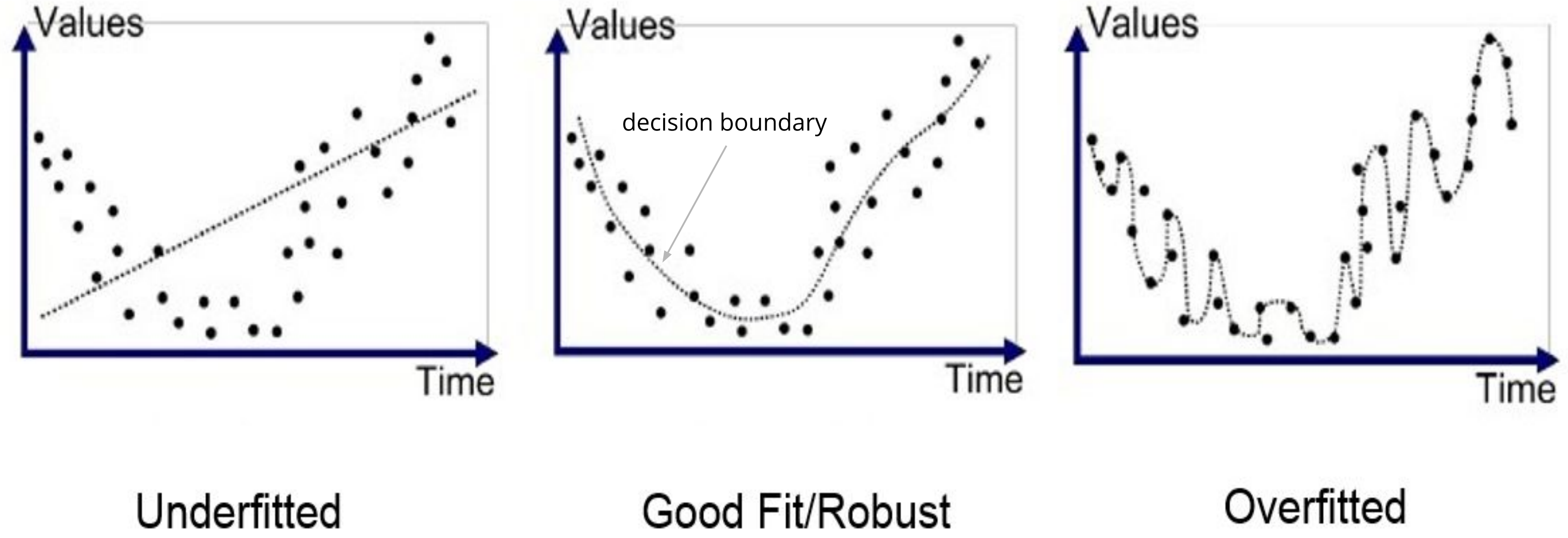
Outliers: very different from example in dataset

hyperparameter tuning: influences how algorithm works

Cross validation: less data, split train set into several subsets

e.g 5-fold CV

ML Concepts: Underfitting and overfitting



Bias and Variance

Regularization (less complex model) to prevent overfitting

i.e. **bias-variance tradeoff**: high bias but reduces variance (reasonable)

- **Low** Bias if predicts well train data, if make mistakes **high** bias
- **Variance (how spread out)**: High variance too sensitive to outliers (overfits)

Model Performance Evaluation using test set

Confusion matrix: actual vs predicted

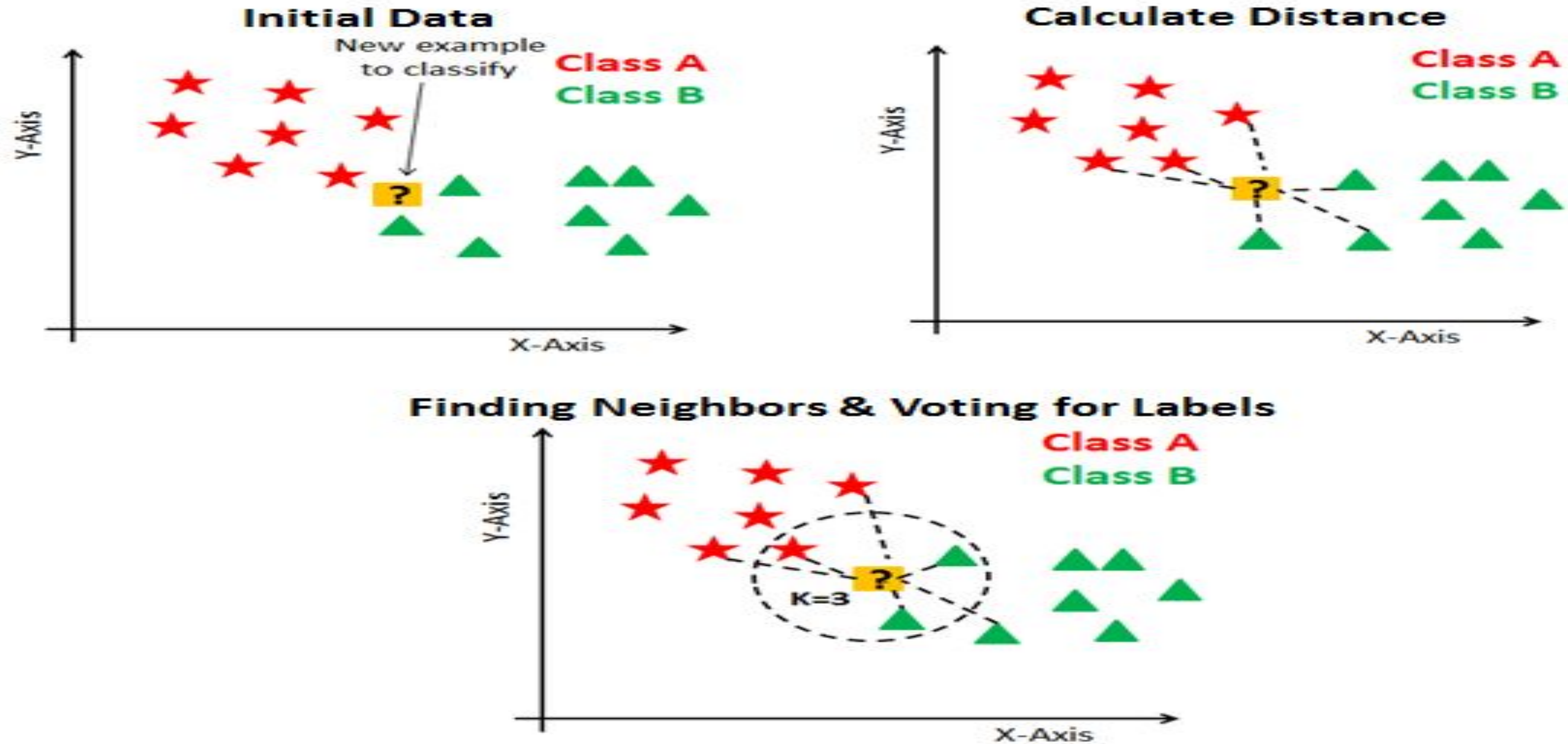
Accuracy: correctly classified divided by

by total no of classified examples

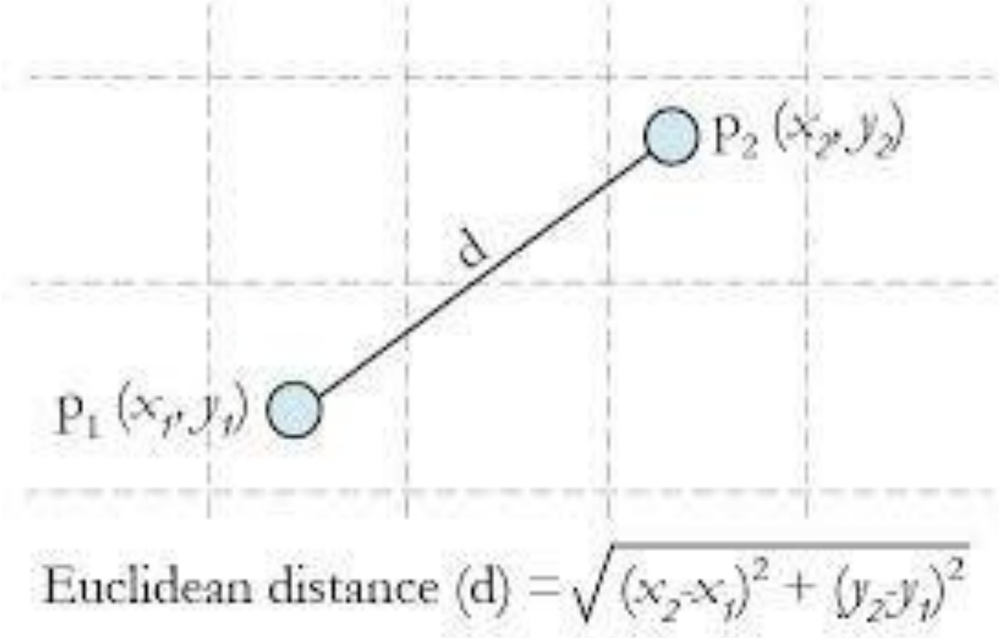
$=(TP+TN)/TP+FP+TN+FN$

		PREDICTED LABELS	
		n' (Predicted)	p' (Predicted)
TRUE LABELS	n (True)	**True Negative** (Number of instances of negative class 'n' correctly predicted)	**False Positive** *(Number of instances of negative class 'n' incorrectly predicted as the positive class 'p')*
	p (True)	**False Negative** *(Number of instances of positive class 'p' incorrectly predicted as the negative class 'n')*	**True Positive** *(Number of instances of positive class 'p' correctly predicted)*

1. K Nearest Neighbor



Euclidean distance



Steps

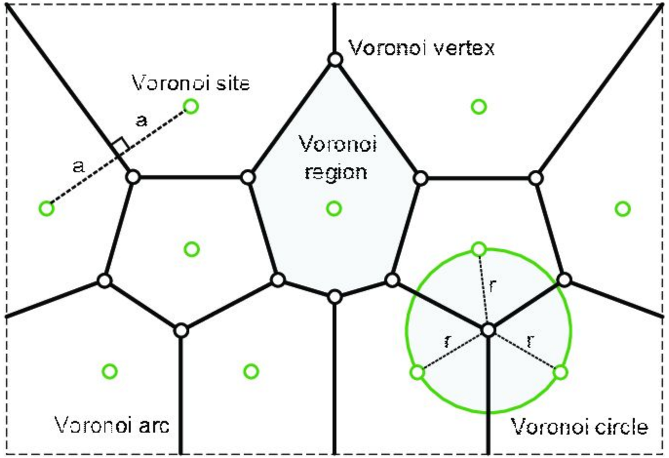
- train **70%**, validation set **20%** and test set **10%**
- k= 1 to n and compare accuracy on validation set to find optimal k
- use **optimal k** to test on test_set

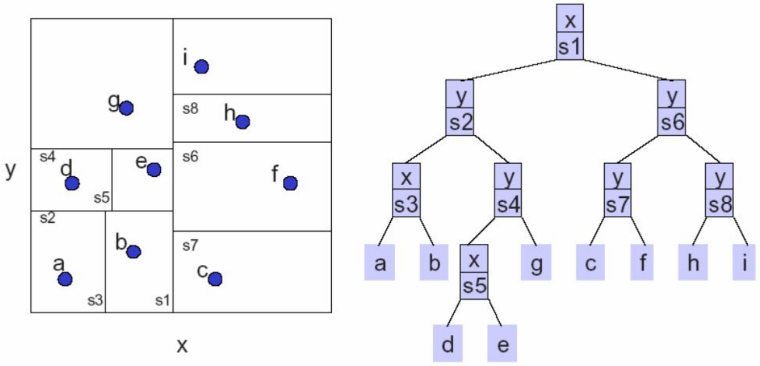
KNN using euclidean

```
#data to train, new data to predict and a value for k
def k_nearest_neighbors(data,predict,k):
    distances=[]  # list of distances
    for group in data: # each class in data
        for features in data[group]: # iterating through features
            euclidean_distance = math.sqrt( (features[0]-predict[0])**2 + (features[1]-predict[1])**2 )
            distances.append([euclidean_distance,group])
    votes=[i[1] for i in sorted(distances)[:k]]
    vote_result=Counter(votes).most_common(1)[0][0]
    return vote_result
```

Further research

- Voronoi diagrams (O(nlogn))
- C 5.0
- K-D trees (log2n depth) to make KNN fast



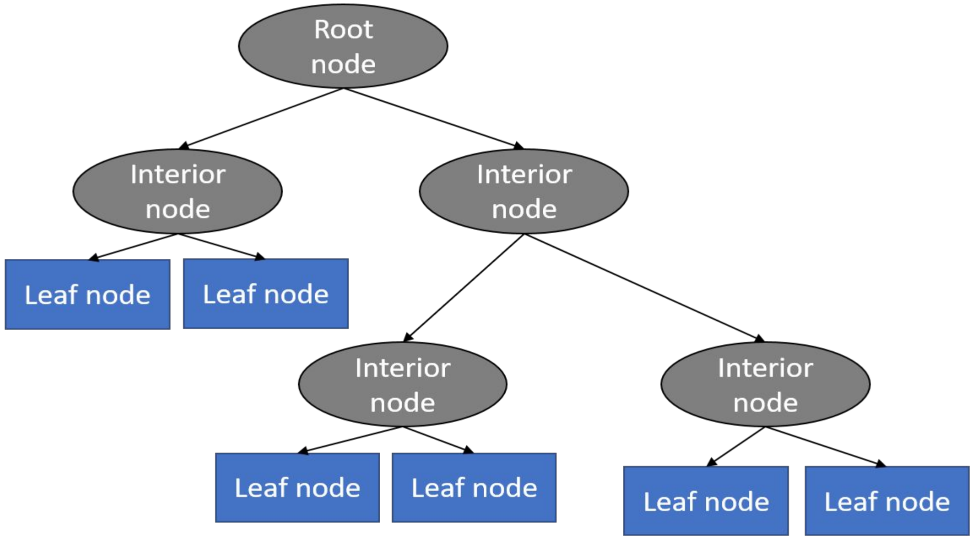


2. Decision Tree

- Tree based classifier

Two phase:

1. Tree construction
2. Tree pruning



Tree Construction

Shannon's Information Theory (ID3)

- Attribute selection at root node determined by **Information Gain**
- Split Criteria: goodness of split by **Entropy**
- High Entropy, more information gain
- Infogain= Entropy of parents- sum of entropy of children of that node

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Tree pruning : improve model

Stop Criteria

1. Pre pruning: stop growing tree when data split insignificant
2. Post pruning: grow full tree then: backtracking during search

Pseudocode

1. Compute the entropy for data-set: entropy(s)
2. For every attribute or feature:
    a. Calculate entropy for all other value: entropy(a)
    b. Take average information entropy for current attribute
    c. Calculate information gain for the current attribute
3. Pick highest gain attribute
4. Repeat until we get the tree we desired.

In [22]:
```
# entropy: measure of uncertainty
def entropy(data):
    label_column=data[:,0]
    _,counts=np.unique(label_column,return_counts=True)

    probability=counts/counts.sum()
    entropy=sum(probability*-np.log2(probability))
    return entropy
```

In [23]:
```
label_column=monk1_data[:,0] # first column=class
_,counts=np.unique(label_column,return_counts=True)
print(counts)
```

```
[62 62]
```

In [24]:
```
entropy(monk1_data)
```

Out[24]:
```
1.0
```

In [25]:
```
entropy(monk2_data)
```

Out[25]:
```
0.957117428264771
```

In [26]:
```
entropy(monk3_data)
```

Out[26]:
```
0.999806132804711
```

Complexity

Time Complexity - depth of tree or total no of level

Space complexity - number of nodes

C5.0 is , improvement to C4.5 and ID3

C5.0 saves memory and adopts Binomial Confidence Limit

Further Research

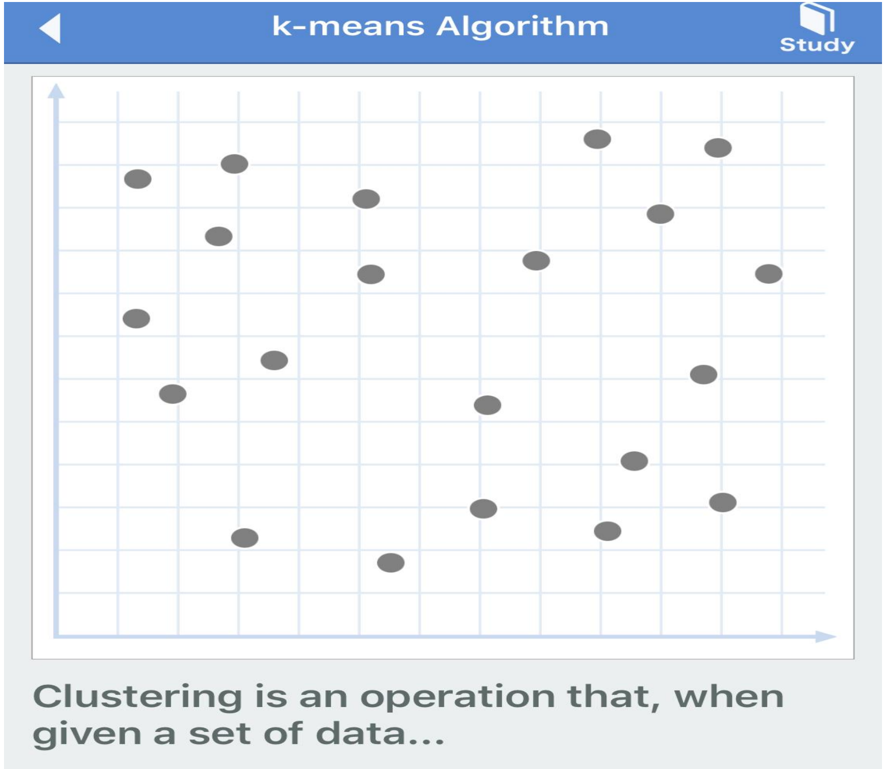
ROC Curve: tools to predict probability of binary income.

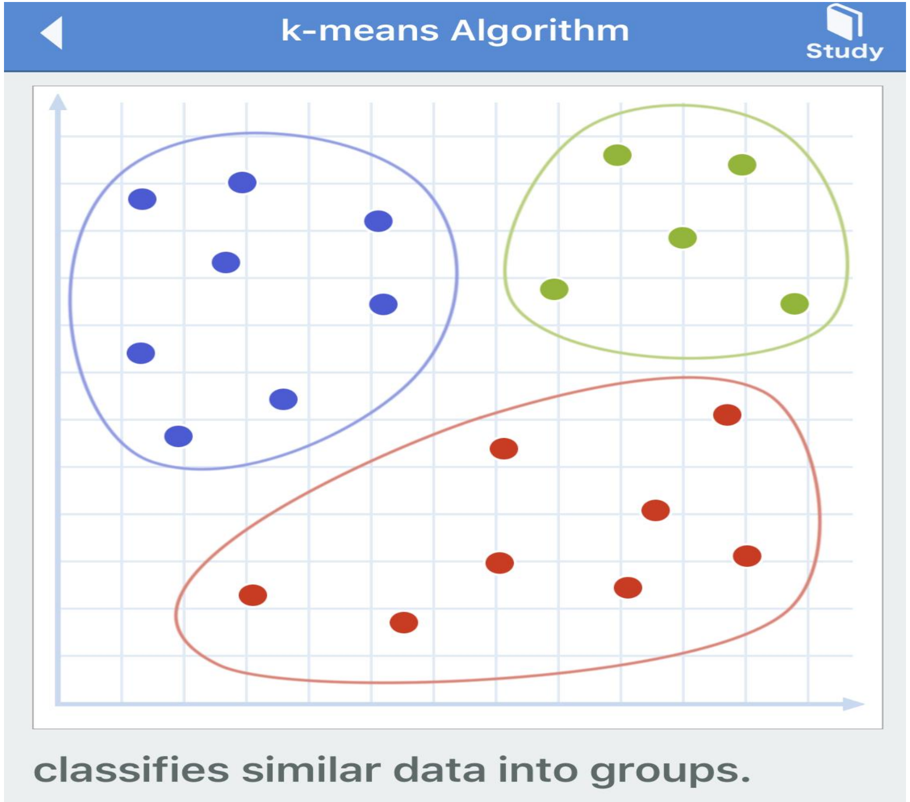
Decision tree provide foundation for Bagging, Random Forest and Gradient Boosting

K- fold Cross validation

Occam’s Razor- prefer simple and avoid unnecessary assumptions

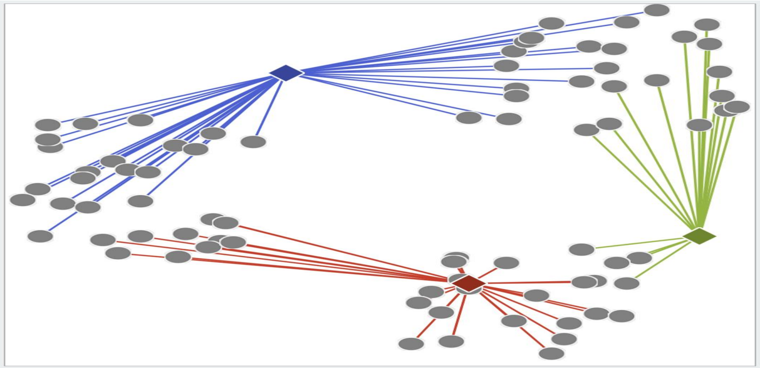
3. K Means Clustering



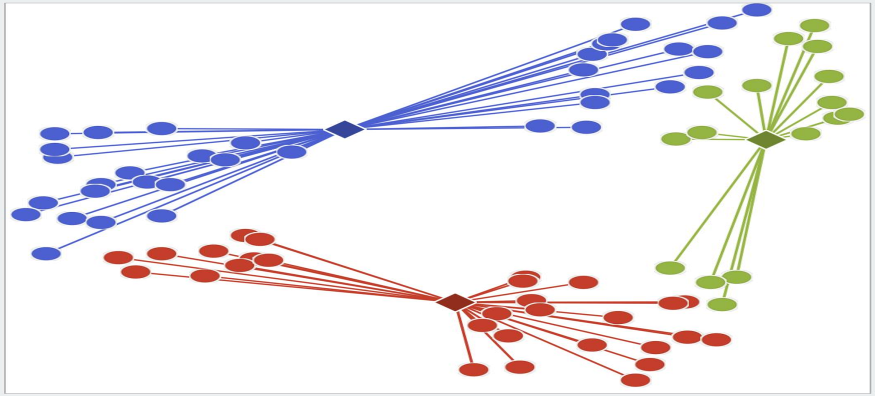


Two basic assumptions

1. cluster center is arithmetic mean of all points belonging to the cluster
2. each point is closer to its own center than to other cluster centers



From each data point, we calculate the distance to the nearest cluster's center.



The mean of the data points in each cluster is calculated, and the center point of the cluster is moved there.

Pseudocode

- initialize k means (centroids) with random values
- iterate through every row of data
- find closest mean to it by a distance metric
- assign it to the mean
- update mean

Search Optimal K

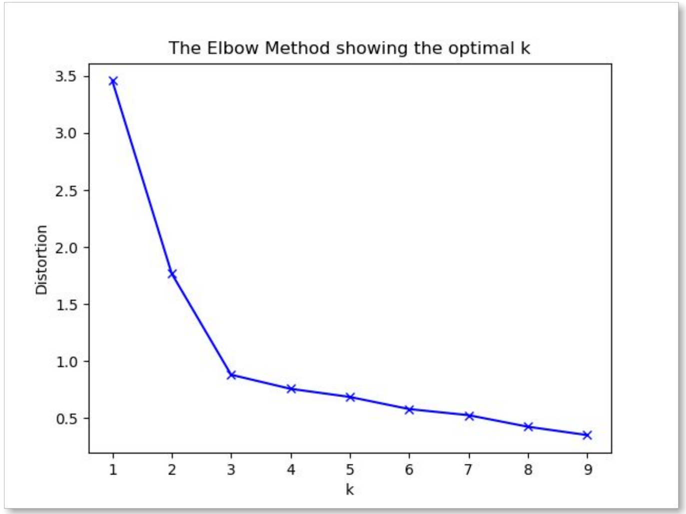
Expectation-Maximization iterative procedure

- guess cluster center and repeat until converges

a. E- step: assign points to nearest cluster center
   estimate the values of the missing data
b. M- step: set the cluster centers to the mean
   update the parameters

NOTES

- Because each iteration of k-means must access every points in the dataset, the algorithm can be relatively slow as the number of sample grows O(n^2)
-
- elbow method to decide optimal k



Further research

- consider ball trees or KD-trees to speed new centroid computation
- Minimum Distance Length (MDL) stopping criteria
- kernel trick - into high dimension

Conclusion

- ML Basic concepts and foundation
- Three algorithms, supervised & unsupervised
- Python and libraries
- Weka
- Further study on other algorithms

References

1. Machine Learning Mastery Blog
2. 100 page Machine Learning Book - Andriy Burkov
3. Pattern classification, 2nd Edition - Duda, Richard et al
4. Geeks for Geeks
5. Vidhya Analytics
6. KDNuggets
7. Towards Data Science - Medium articles
8. CIS 520 Machine Learning, UPenn Notes 2018
9. Tom Mitchell Machine Learning Course - Carnegie Mellon University