Lecture 10:
Machine Learning Classification

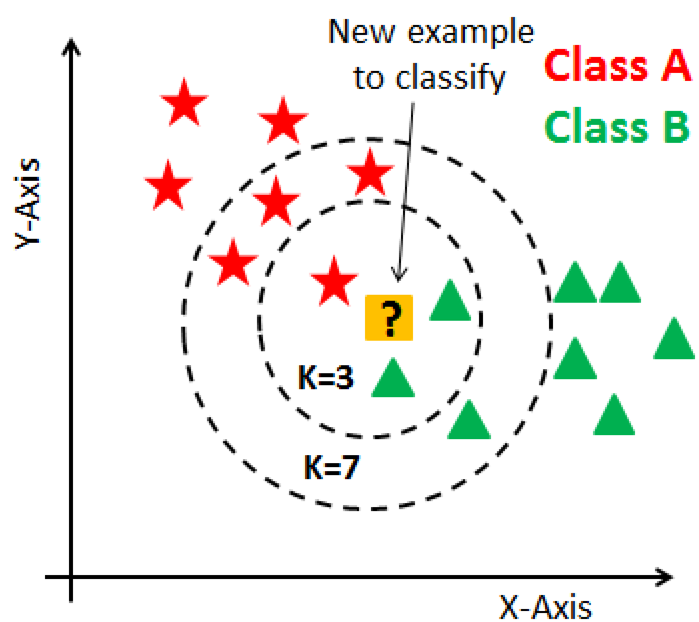
Table of Contents

Part 1

BIG DATA CLASSIFICATION

Big Data Classification

- What is KNN?
  - K-Nearest Neighbors
    - Classifies unlabeled data points by assigning them the class of similar labeled data points



Big Data Classification

- KNN applications
  - They have been used successfully for
    - Computer vision applications
      - Character recognition and facial recognition in both still images and video
    - Identifying patterns in hospital data
      - Detection of diseases
    - Predicting whether a person will enjoy a movie or music recommendation

Big Data Classification

- How KNN works?
  - Suppose that we want to predict T-shirt size of a new customer given height and weight information

Height	Weight	Size
158	58	M
158	59	M
158	63	M
160	59	M
160	60	M
163	60	M
163	61	M
160	64	L
163	64	L
165	61	L
165	62	L
165	65	L
168	62	L
168	63	L
168	66	L
170	63	L
170	64	L
170	68	L
162	62	???

Big Data Classification

- How KNN works?
  - Step 1: Determine parameter k ($k > 0$)
  - Step 2: Determine similarity by calculating the distance between a test point and all other points in the dataset
  - Step 3: Sort the dataset according to the distance values
  - Step 4: Determine the category of the k-th nearest neighbors
  - Step 5: Use simple majority of the category of the k nearest neighbors as the category of a test point

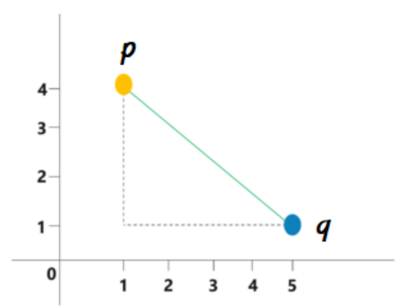
Big Data Classification

❖ How KNN works?

- Step 1: Determine parameter *k* (*k* > 0)
  - Suppose *k* = 3
- Step 2: Determine similarity by calculating the distance between a test point and all other points in the dataset
  - Euclidean distance

$$\mathrm{dist}(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \ldots + (p_n - q_n)^2}$$

– Where *p* and *q* are the data points to be compared



Big Data Classification

- How KNN works?
  - Step 2: Calculate the distance between a test point and all other points in the dataset

Height	Weight	Size	Distance
158	58	M	5.656854
158	59	M	5
158	63	M	4.123106
160	59	M	3.605551
160	60	M	2.828427
163	60	M	2.236068
163	61	M	1.414214
160	64	L	2.828427
163	64	L	2.236068
165	61	L	3.162278
165	62	L	3
165	65	L	4.242641
168	62	L	6
168	63	L	6.082763
168	66	L	7.211103
170	63	L	8.062258
170	64	L	8.246211
170	68	L	10
162	62	???	

Big Data Classification

❖ How KNN works?

- Step 3: Sort the dataset according to the distance values

Height	Weight	Size	Distance
163	61	M	1.414214
163	60	M	2.236068
163	64	L	2.236068
160	60	M	2.828427
160	64	L	2.828427
165	62	L	3
165	61	L	3.162278
160	59	M	3.605551
158	63	M	4.123106
165	65	L	4.242641
158	59	M	5
158	58	M	5.656854
168	62	L	6
168	63	L	6.082763
168	66	L	7.211103
170	63	L	8.062258
170	64	L	8.246211
170	68	L	10
162	62	???	

Big Data Classification

- How KNN works?
  - Step 4: Determine the category of the *k*-th nearest neighbors

k=3

Height	Weight	Size	Distance
163	61	M	1.414214
163	60	M	2.236068
163	64	L	2.236068
160	60	M	2.828427
160	64	L	2.828427
165	62	L	3
165	61	L	3.162278
160	59	M	3.605551
158	63	M	4.123106
165	65	L	4.242641
158	59	M	5
158	58	M	5.656854
168	62	L	6
168	63	L	6.082763
168	66	L	7.211103
170	63	L	8.062258
170	64	L	8.246211
170	68	L	10
162	62	???	

Big Data Classification

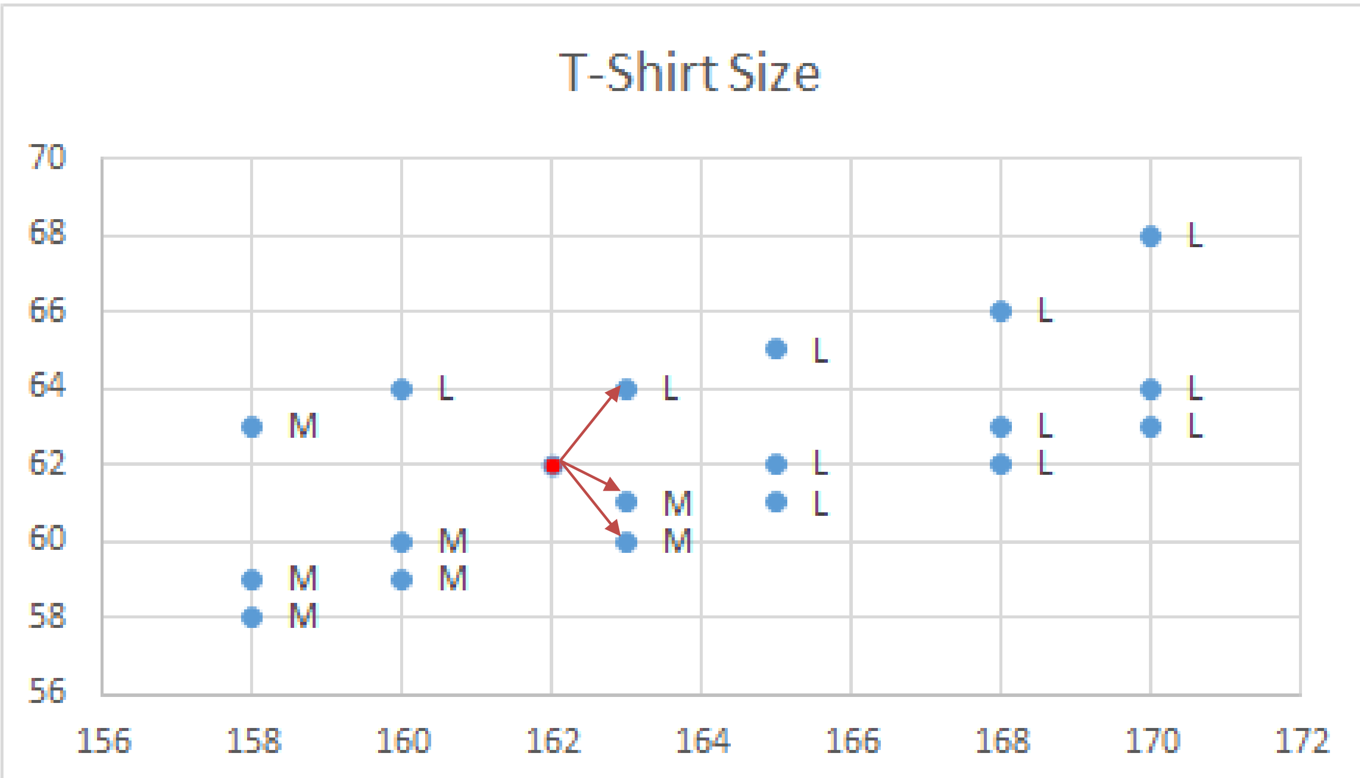
- How KNN works?
  - Step 5: Use simple majority of the category of the *k* nearest neighbors as the category of a test point

k=3

Height	Weight	Size	Distance
163	61	M	1.414214
163	60	M	2.236068
163	64	L	2.236068
160	60	M	2.828427
160	64	L	2.828427
165	62	L	3
165	61	L	3.162278
160	59	M	3.605551
158	63	M	4.123106
165	65	L	4.242641
158	59	M	5
158	58	M	5.656854
168	62	L	6
168	63	L	6.082763
168	66	L	7.211103
170	63	L	8.062258
170	64	L	8.246211
170	68	L	10
162	62	M	

Big Data Classification

- How KNN works?
  - Step 5: Use simple majority of the category of the *k* nearest neighbors as the category of a test point

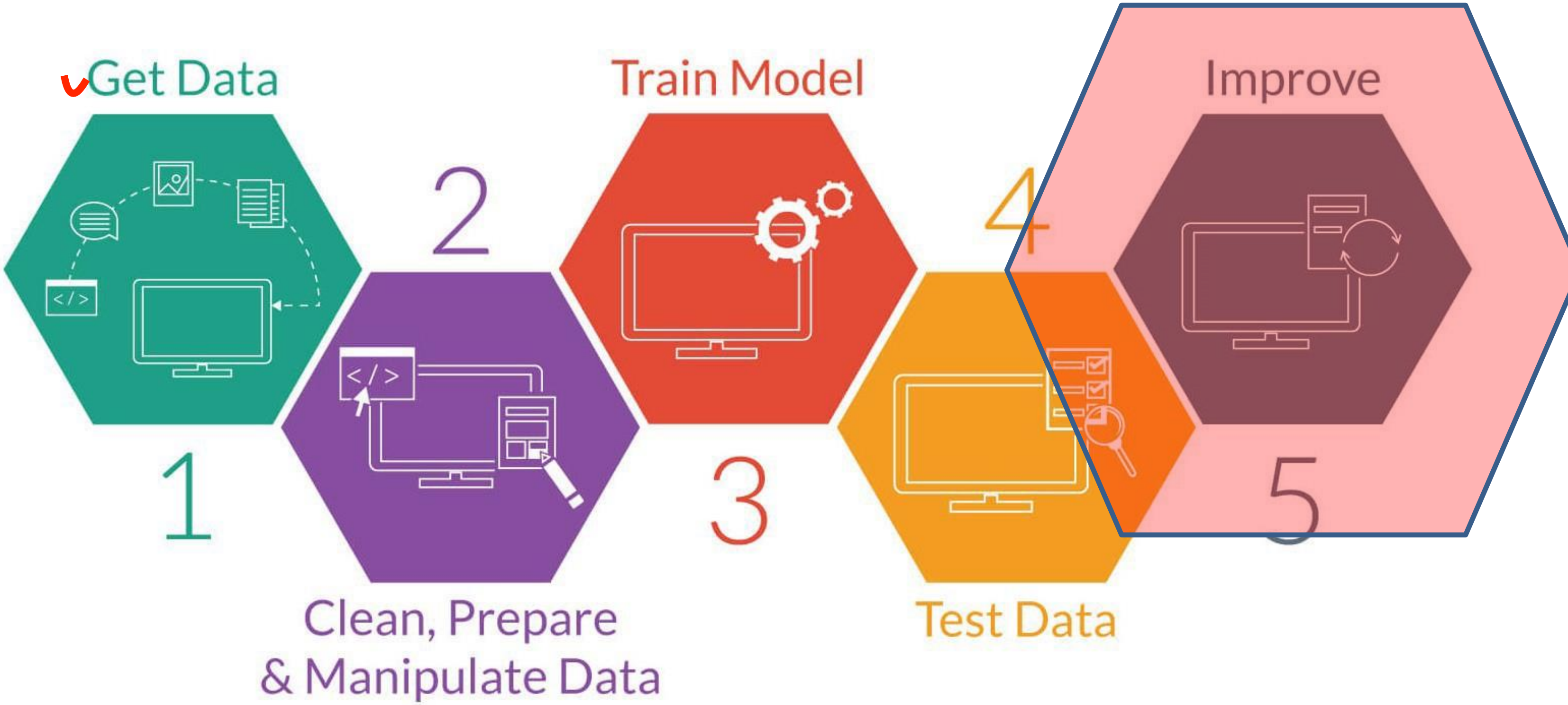


Part 2

EVALUATION OF BIG DATA ANALYTICS

In the last lecture

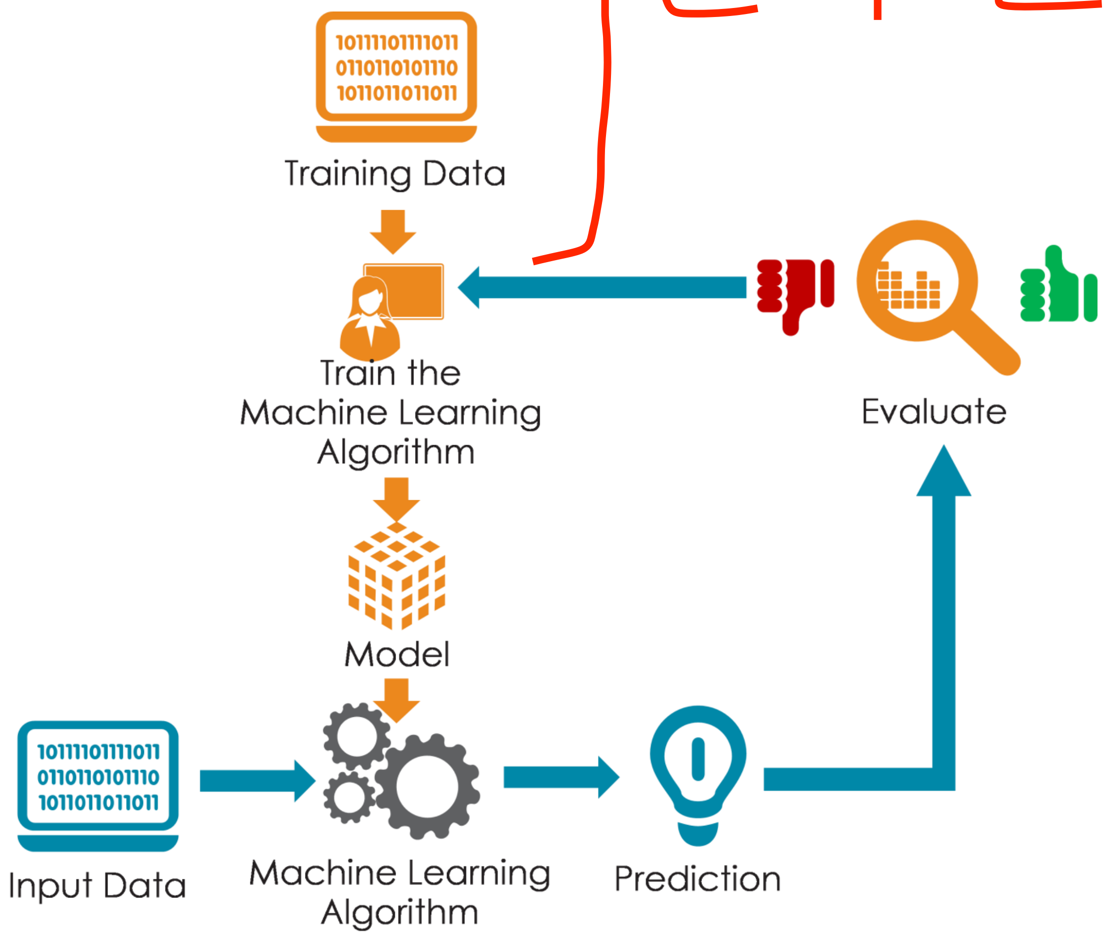
❖ Big data process



Evaluation of Big Data Analytics

❖ How Big Data analytics work?

ETL



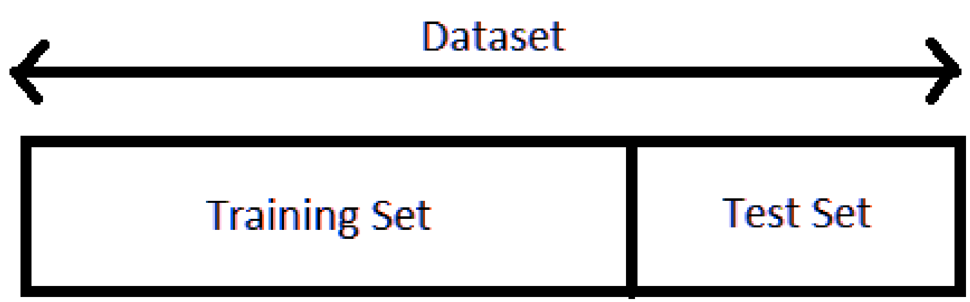
Evaluation of Big Data Analytics

- Why evaluate?
  - Building Big Data Analytics is based on the principle of continuous feedback
  - The Big Data Analytics are built and model performance is evaluated further continuously and continue until you achieve a desirable accuracy
  - Big Data Analytics evaluation metrics are used to explain the performance of metrics
  - It is important to check performance metrics before carrying out predictions

95%

Evaluation of Big Data Analytics

- Train and test split
  - Train dataset
    - The actual dataset that we use to train the model
      - The model sees and learns from this data
  - Test dataset
    - Dataset used for evaluating the model
  - We usually split the data around 20%-80% between testing and training stages



Evaluation of Big Data Analytics

- ❖ Train and test split
  - sklearn library

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test =
Train_test_split(training_points, training_labels, test_size=0.2, random_state=4)
```

  - test_size=0.2
    - Test dataset is 20% and training dataset is 80%
  - random_state=4
    - data is randomly assigned unless you use random_state hyperparameter

Part 3

EVALUATION METRICS

Evaluation Metrics

- Classification accuracy
  - Confusion matrix
  - Accuracy
  - Error rate
  - Precision
  - Recall
  - F measure
- Regression accuracy
  - Mean squared error

Evaluation Metrics

- Classification accuracy
  - sklearn libraries

```
from sklearn.metrics import confusion_matrix
from sklearn import metrics

print(confusion_matrix(y_test, guesses))

print(metrics.accuracy_score(y_test, guesses))

print(metrics.precision_score(y_test, guesses, average='binary'))

print(metrics.recall_score(y_test, guesses, average='binary'))

print(metrics.f1_score(y_test, guesses, average='binary'))
```

Evaluation Metrics

- Confusion matrix
  - Confusion matrix is a table that categorizes predictions according to whether they match the actual value

	Predicted 0	Predicted 1
Actual 0	TN	FP
Actual 1	FN	TP

Evaluation Metrics

- Confusion matrix
  - The most common performance measures consider the model's ability to discern one class versus all others
    - The class of interest is known as the **positive**
    - All others are known as **negative**

  - The relationship between the positive class and negative class predictions can be depicted as a 2 x 2 confusion matrix
    - It tabulates whether predictions fall into one of the four categories
      - **True Positive (TP):** Correctly classified as the class of interest
      - **True Negative (TN):** Correctly classified as not the class of interest
      - **False Positive (FP):** Incorrectly classified as the class of interest
      - **False Negative (FN):** Incorrectly classified as not the class of interest

Evaluation Metrics

❖ Accuracy

- With the 2 x 2 confusion matrix, we can formalize our definition of prediction accuracy (sometimes called the success rate) as:

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \qquad (1)$$

❖ Error rate

- The error rate or the proportion of the incorrectly classified examples is specified as:

$$\text{error rate} = \frac{\text{FP} + \text{FN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} = 1 - \text{accuracy} \qquad (2)$$

Evaluation Metrics

- Task 1
  - Calculate accuracy and error rate for cancer dataset
  - Accuracy
    - (29 + 71) / (29 + 71 + 9 + 5)
    - Result: 0.877
  - Error rate
    - (9 + 5) / (29 + 71 + 9 + 5)
    - Result: 0.122

Evaluation Metrics

- Precision 양품 예측
  - The precision is defined as the proportion of positive examples that are truly positive
  - In other words, when a model predicts the positive class, how often is it correct

$$\text{precision} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3}$$

- Recall 진짜 양품
  - On the other hand, recall is a measure of how complete the results are

$$\text{recall} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{4}$$

Evaluation Metrics

- Task 2
  - Calculate accuracy and error rate for cancer dataset

  - Precision
    - 71/ (71 + 6)
    - Result: 0.934

  - Recall
    - 71/ (71 + 9)
    - Result: 0.887

Evaluation Metrics

- F-measure
  - A measure that combines precision and recall into a single number is known as the F-measure
    - Sometimes called the F1 score or F-score

$$\text{F-measure} = \frac{2 \times \text{precision} \times \text{recall}}{\text{recall} + \text{precision}} = \frac{2 \times \text{TP}}{2 \times \text{TP} + \text{FP} + \text{FN}} \qquad (5)$$

- Task 3
  - F-measure
    - (2 * 0.934 * 0.887)/ (0.934 + 0.887) = 1.656/1.821
    - Result: 0.91

Part 4

ACCURACY IMPROVEMENT

Big Data Design Process

- Big data design process contains of two main steps
  - Data management, data training and continuous improving accuracy

- Steps for big data design
  1. Loading libraries
  2. Loading dataset
  3. Data observation
  4. Exploratory Data Analysis
  5. Splitting into training and testing datasets
  6. Training model and checking out accuracy
  7. Improving accuracy by tuning hyperparameters (number of k)
  8. Changing ratios of training and test datasets
  9. Rescaling

Big Data Design Process

❖ Step 1

- Loading several libraries that will be used to do the analysis in this tutorial
  - I assume that you have already installed the library

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn import metrics
```

Big Data Design Process

❖ Step 2

- Load the dataset to be used, dataset contains historical data from patients who have been examined for heart disease

```
df = pd.read_csv('D:₩My Datasets₩heart.csv')
```

❖ Step 3

- Let's see some general information from the data to be more familiar with our data

```
#print(df.head())
#print(df.shape)
#print(df.info())
```

Big Data Design Process

- Step 3
  - Result of print(df.head())
    - Shows the top five records of the dataset

```
Run:  mongodb
C:\Users\OFFICE\PycharmProjects\MongoDB\venv\Scripts\python.exe C:/Users/OFFICE/PycharmProjects/MongoDB/mongodb.py
   age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal  target
0   63    1   3       145   233    1  ...      0      2.3      0   0     1       1
1   37    1   2       130   250    0  ...      0      3.5      0   0     2       1
2   41    0   1       130   204    0  ...      0      1.4      2   0     2       1
3   56    1   1       120   236    0  ...      0      0.8      2   0     2       1
4   57    0   0       120   354    0  ...      1      0.6      2   0     2       1

[5 rows x 14 columns]

Process finished with exit code 0
```

  - Task 1
    - Check out print(df.shape) and print(df.info()) by yourself

Big Data Design Process

❖ Step 4

- Conducting Exploratory Data Analysis (EDA) to understand our data better

1. Target class distribution
   - Looks like the target feature is balanced because the number of values 0 and 1 does not differ much

```
print(df['target'].value_counts())
```

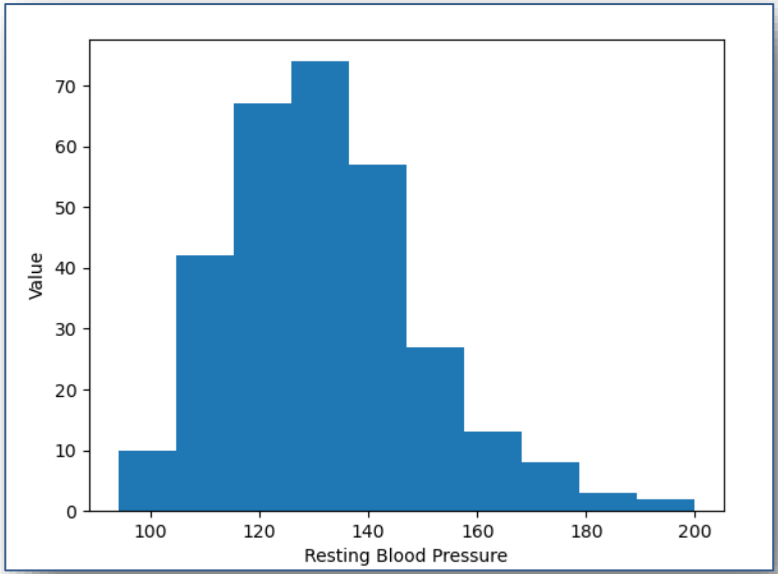
```
1    165
0    138
Name: target, dtype: int64

Process finished with exit code 0
```

Big Data Design Process

❖ Step 4

2. Histogram of trestbps attribute

```
plt.hist(df['trestbps'])
plt.xlabel('Resting Blood Pressure')
plt.ylabel('Value')
plt.show()
```

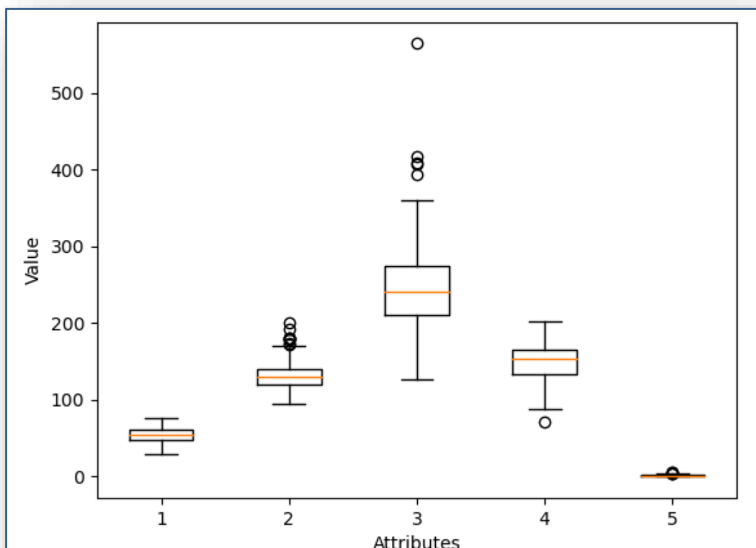


Big Data Design Process

❖ Step 4

3. Outlier of all attributes

```
data_to_boxplot = [df['age'], df['trestbps'], df['chol'],
                   df['thalach'], df['oldpeak']]
plt.boxplot(data_to_boxplot)
plt.xlabel('Attributes')
plt.ylabel('Value')
plt.show()
```



Big Data Design Process

❖ Step 4

4. Missing values

```
print(df.isnull().sum())
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64

Process finished with exit code 0
```

Big Data Design Process

- Step 5
  - Splitting into training and test datasets to check out the accuracy

```
training_points = df.drop(columns=['target'])
training_labels = df['target']

X_train, X_test, y_train, y_test = train_test_split(
            training_points,
            training_labels,
            test_size=0.3,
            random_state=4)

print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

Big Data Design Process

- Step 6
  - Training the model (k=5) and check the accuracy

```
classifier = KNeighborsClassifier(n_neighbors = 5)
classifier.fit(X_train, y_train)
guesses = classifier.predict(X_test)

print(guesses)
print(confusion_matrix(y_test, guesses))
print(metrics.accuracy_score(y_test, guesses))
```

  - **Initial classification accuracy is 0.5934065934065934**

Big Data Design Process

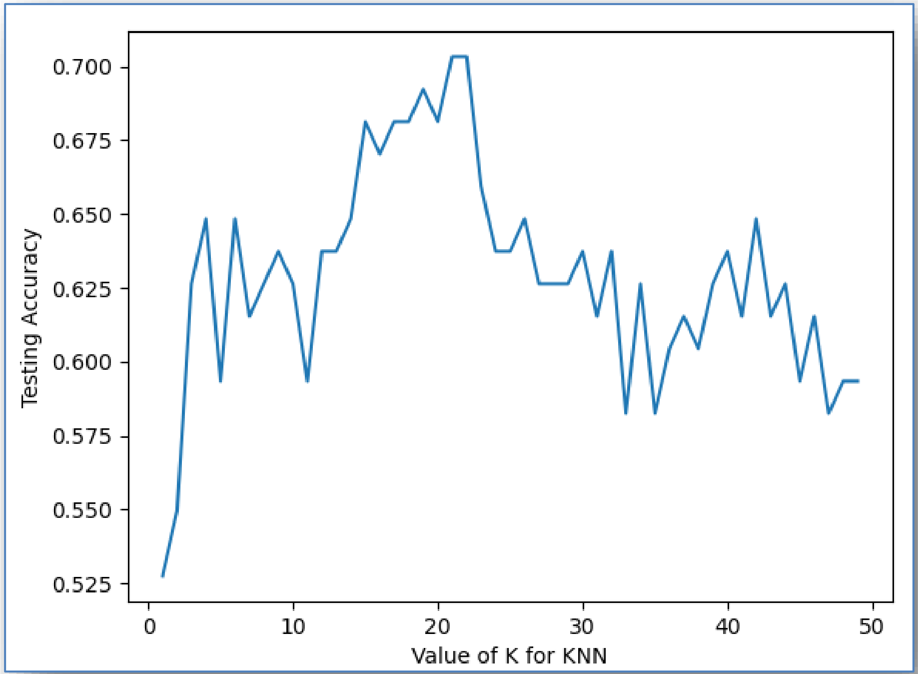
❖ Step 7

- Improving accuracy by tuning hyperparameters (number of k)

```
k_range = range(1, 50)

accuracy_scores = []

for k in k_range:
    classifier = KNeighborsClassifier(n_neighbors = k)
    classifier.fit(X_train, y_train)
    guesses = classifier.predict(X_test)
    accuracy_scores.append(metrics.accuracy_score(y_test, guesses))
print(accuracy_scores)

#Visualize the result of KNN accuracy with matplotlib
plt.plot(k_range, accuracy_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Testing Accuracy')
plt.show()
```

Big Data Design Process

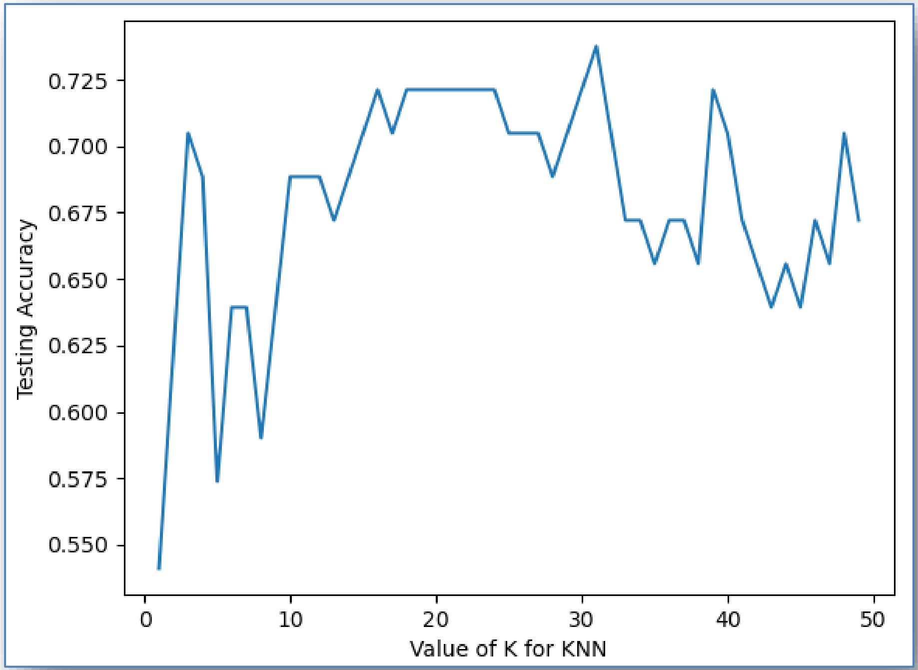
- Step 7
  - Result of tuning hyperparameters
    - **Highest accuracy: 0.7032967032967034**



Big Data Design Process

❖ Step 8

- Changing ratios of training and test datasets
  - Training dataset -> 80% and test dataset -> 20%
- **Highest accuracy: 0.7377049180327869**



Big Data Design Process

- Rescaling
  - KNN is a Distance-Based algorithm where KNN classifies data based on proximity to K-Neighbors
  - Then, often we find that the features of the data we used are not at the same scale/units
    - An example is when we have features age and height
    - Obviously these two features have different units, the feature age is in year and the height is in centimeter
  - This unit difference causes Distance-Based algorithms such as KNN to not perform optimally

Big Data Design Process

- Rescaling
  - It is necessary to rescaling features that have different units to have same scale/units

  - Rescaling methods
    - Min-Max Scaling
      - Min-Max Scaling uses the minimum and maximum values of a feature to rescale values within a range

    - Standard Scaling
      - Rescale features to be approximately standard normally distributed

    - Robust Scaling
      - Rescale the feature using the median and quartile range

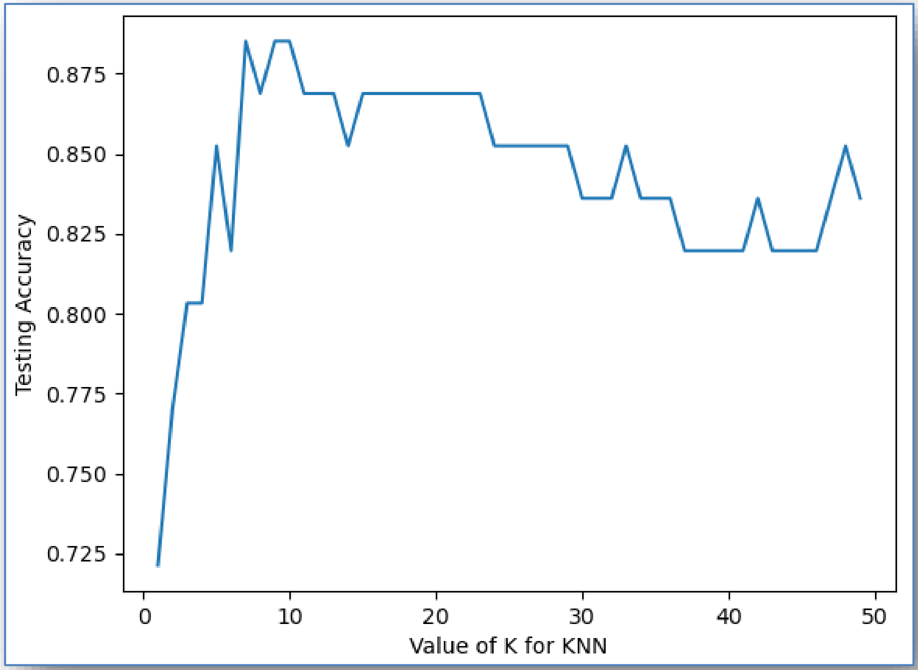
Big Data Design Process

- Step 9
  - Standard Scaling

```
from sklearn.preprocessing import StandardScaler

#Create copy of dataset.
df_model = df.copy()

#Rescaling features age, trestbps, chol, thalach, oldpeak.
scaler = StandardScaler()

features = [['age', 'trestbps', 'chol', 'thalach', 'oldpeak']]
for feature in features:
    df_model[feature] = scaler.fit_transform(df_model[feature])

training_points = df_model.drop(columns=['target'])
training_labels = df_model['target']
```

Big Data Design Process

- Step 9
  - Standard Scaling
    - **Highest accuracy: 0.8852459016393442**



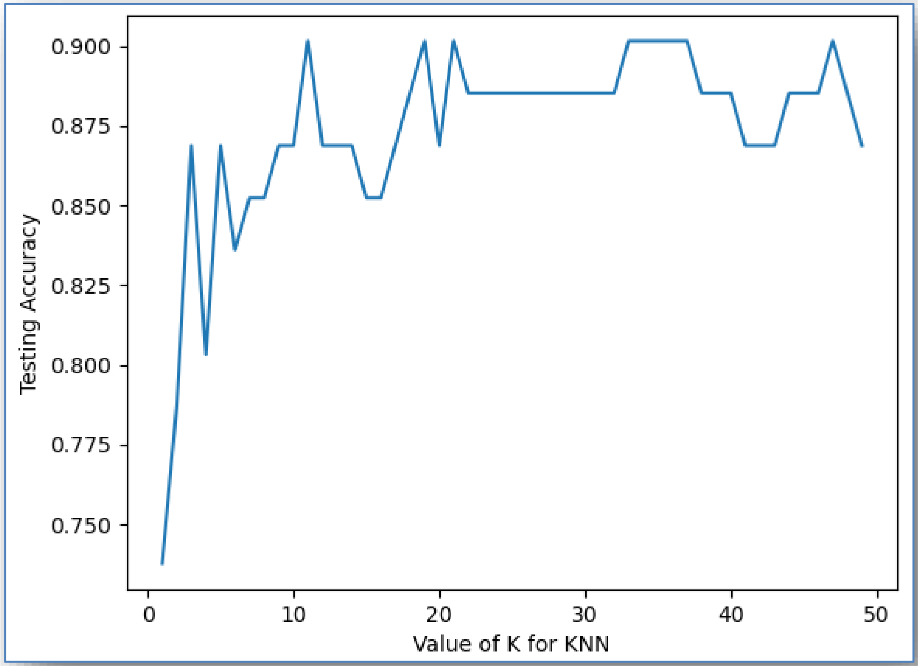
Big Data Design Process

- Step 9
  - Min-Max Scaling

```
from sklearn.preprocessing import MinMaxScaler

#Create copy of dataset.
df_model = df.copy()

#Rescaling features age, trestbps, chol, thalach, oldpeak.
scaler = MinMaxScaler()

features = [['age', 'trestbps', 'chol', 'thalach', 'oldpeak']]
for feature in features:
    df_model[feature] = scaler.fit_transform(df_model[feature])

training_points = df_model.drop(columns=['target'])
training_labels = df_model['target']
```

Big Data Design Process

- Step 9
  - Min-Max Scaling
    - **Highest accuracy: 0.9016393442622951**



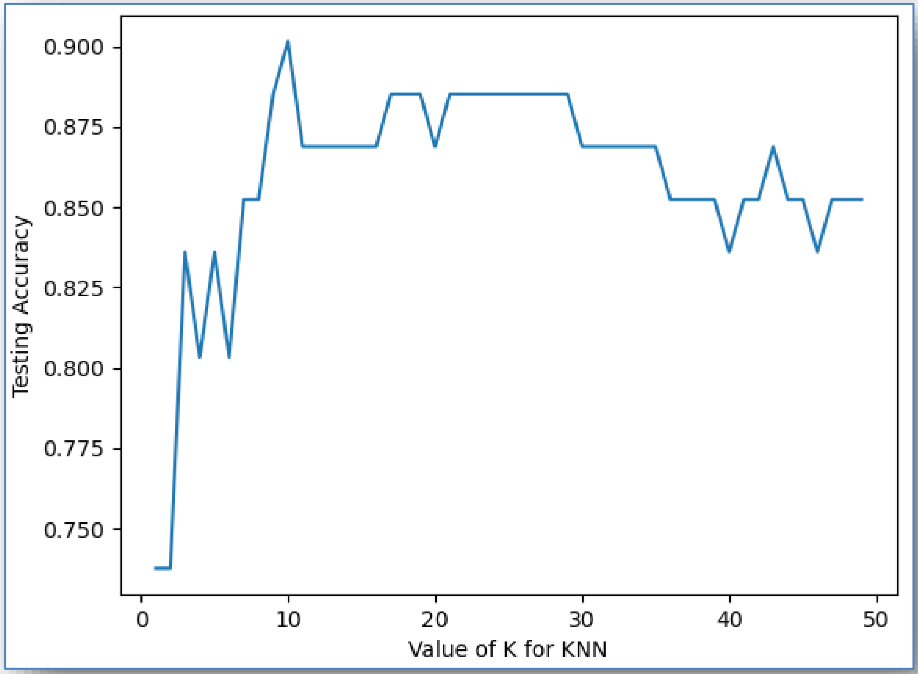
Big Data Design Process

- Step 9
  - Robust Scaling

```
from sklearn.preprocessing import RobustScaler

#Create copy of dataset.
df_model = df.copy()

#Rescaling features age, trestbps, chol, thalach, oldpeak.
scaler = RobustScaler()

features = [['age', 'trestbps', 'chol', 'thalach', 'oldpeak']]
for feature in features:
    df_model[feature] = scaler.fit_transform(df_model[feature])

training_points = df_model.drop(columns=['target'])
training_labels = df_model['target']
```

Big Data Design Process

- Step 9
  - Robust Scaling
    - **Highest accuracy: 0.9016393442622951**



Homework for Lecture 10

- Submit your source code for the following task:
  1. Try all source code in the lecture

- Submission: source code and result screenshots