Machine Learning

Practice 1

Quan Minh Phan & Ngoc Hoang Luong

University of Information Technology (UIT)

March 24, 2021

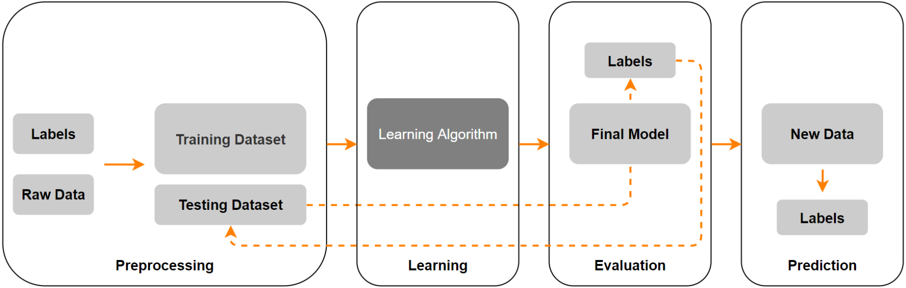
Overview

1. A roadmap for building machine learning system
2. Data Pre-processing
3. K-Nearest Neighbors
4. Linear Regression
5. Model Evaluation

Roadmap

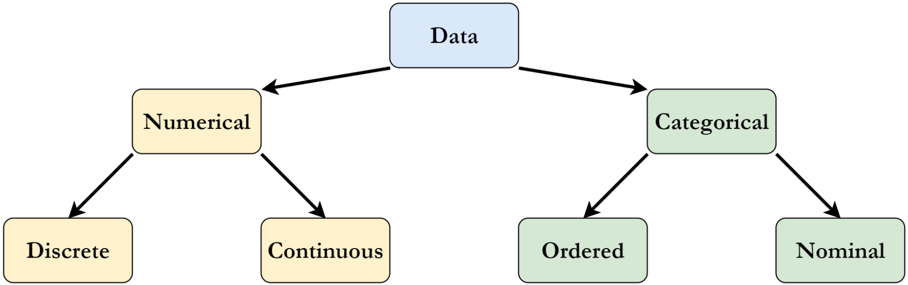
5 major steps:

- Data Pre-processing
- Model Learning
- Model Evaluation
- Prediction
- Model Deployment



Overview

Types of Data



Numerical: quantitative data

- Discrete: the number of students, the age of a person, ...
- Continuous: the height of a person, the score of a student, ...

Categorical: qualitative data

- Ordered: food ratings (excellent, good, bad), feelings (happy, not bad, bad), ...
- Nominal: the name of students, ...

How to load data?

Syntax (load)

pandas.read_csv(*filepath*)

Examples

```
>> import pandas as pd
>> df = pd.read_csv('/content/drive/MyDrive/Colab/iris.csv')
```

Syntax (show)

pandas.DataFrame.head(*n*)

Examples

```
>> df.head(n = 7)
```

Data Representation

Independent variables

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Target variable

Indepedent variables should NOT contain

- Missing or NULL values
- Outliers
- Data on different scales
- Special characters
- ...

Data Cleaning

- Including processes of detecting and correcting (or removing) missing values or outliers.
- Ensuring data is correct, consistent and usable.

Missing values

- In .csv files, missing values are usually represented as empty, 'NA', 'N/A', 'null', 'nan', 'NaN'.

```
ID,Name,Age,Grade
1,Nguyen Van A,21,Good
2,Nguyen Van B,,Excellent
3,Nguyen Van C,20,Good
4,Nguyen Van D,25,
```

a.

	ID	Name	Age	Grade
0	1	Nguyen Van A	21.0	Good
1	2	Nguyen Van B	NaN	Excellent
2	3	Nguyen Van C	20.0	Good
3	4	Nguyen Van D	25.0	NaN

b.

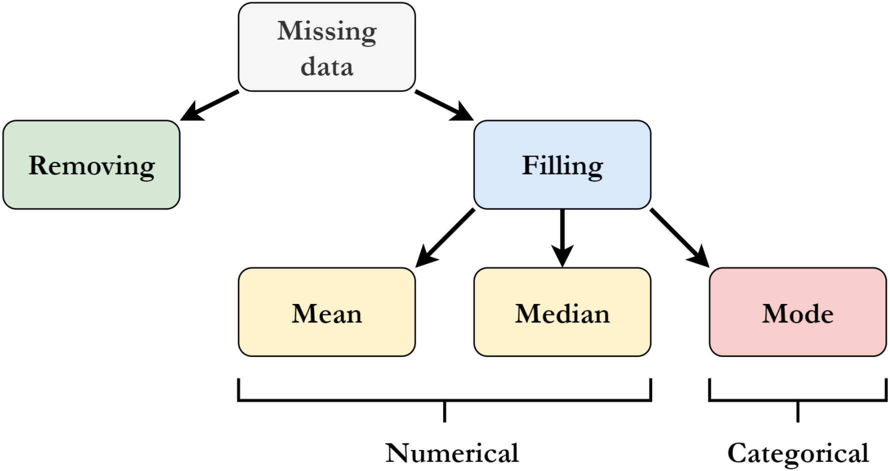
Syntax (count 'NaN')

pandas.DataFrame.isna().sum()

Examples

> countNULL = df.isna().sum()
> null_columns = countNULL[countNULL > 0]
> null_columns

How to handle?



Removing

Syntax

pandas.DataFrame.dropna($inplace$)

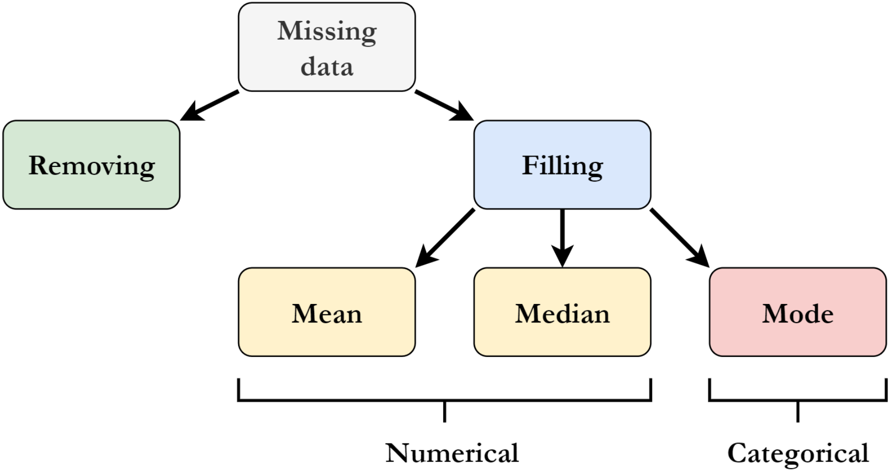
Examples

$>$ df.dropna($inplace = True$)

or

$>$ df $=$ df.dropna($inplace = False$)

How to handle?



Filling

Examples

Find the mean, median, and mode for the following list of values:
13, 18, 13, 14, 13, 16, 14, 21, 13

Mean

- $mean = (13 + 18 + 13 + 14 + 13 + 16 + 14 + 21 + 13)/9 = 15$

Median

- Sorting the list: 13, 13, 13, 13, 14, 14, 16, 18, 21
- $median = 14$

Mode

- $mode = 13$

Filling

Step 1: Calculating the filling values

Syntax (calculate the mean)

pandas.DataFrame.mean()

Examples

```
> mean_age = df['Age'].mean()
> mean_age
```

Syntax (calculate the median)

pandas.DataFrame.median()

Examples

```
> median_height = df['Height'].median()
> median_height
```

Filling

Step 1: Calculating the filling values

Syntax (calculate the mode)

pandas.DataFrame.mode()[0]

Examples

```
> mode_grade = df['Grade'].mode()[0]
> mode_grade
```

Filling

Step 2: Replacing 'NaN' by the filling values

Syntax

$$\textbf{pandas.DataFrame.fillna}(value, inplace)$$

Examples

$>$ df['Age'].fillna($value = mean\_age, inplace = True$)
$>$ df['Height'].fillna($value = median\_height, inplace = True$)
$>$ df['Grade'].fillna($value = mode\_grade, inplace = True$)

Outliers

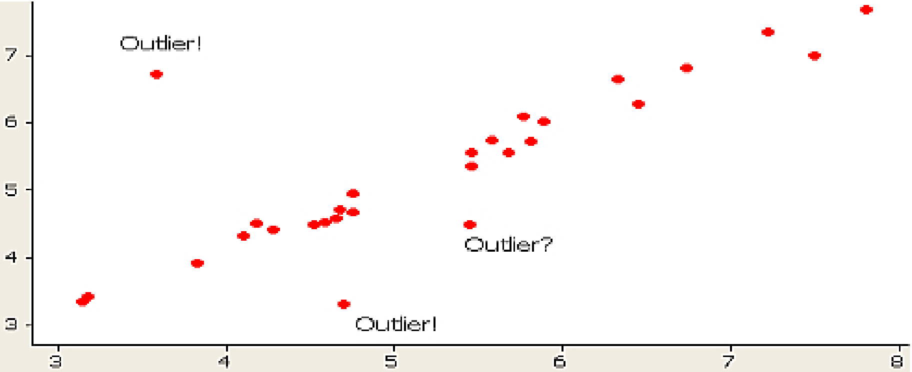


Figure: Examples of outliers

Outliers

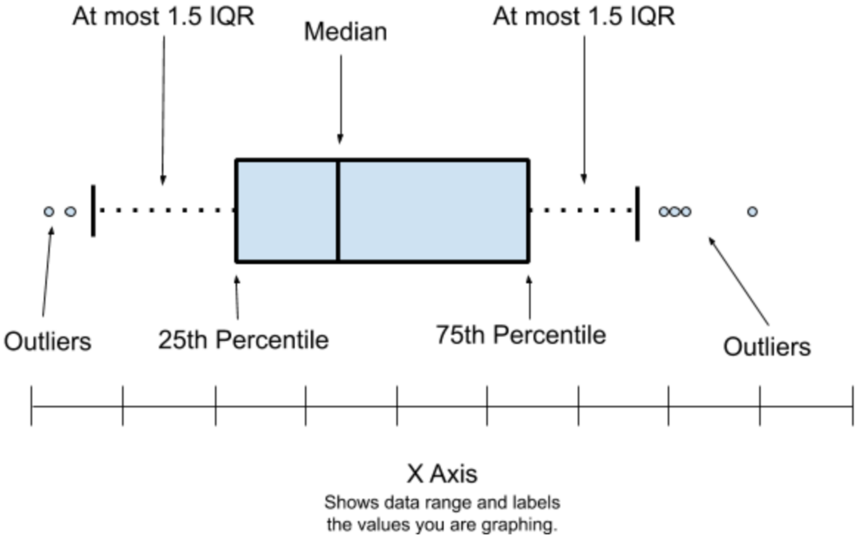
Syntax (plot the outliers)

seaborn.boxplot(*data*)

Examples

```
>> import seaborn as sbn
>> sbn.boxplot(df['Height'])
```

Outliers


At most 1.5 IQR
Median
At most 1.5 IQR
Outliers
25th Percentile
75th Percentile
Outliers
X Axis
Shows data range and labels
the values you are graphing.

Outliers

Examples

Find the outliers on 71, 70, 90, 70, 70, 60, 70, 72, 72, 320, 71, 69

Outliers

Examples

Find the outliers on 71, 70, 90, 70, 70, 60, 70, 72, 72, 320, 71, 69

Solution

- Sort the data: 60, 69, 70, 70, 70, 70, 71, 71, 72, 72, 90, 320
- Calculate the median (Q2) $\rightarrow (70 + 71)/2 = 70.5$
- Calculate the lower quartile (Q1) $\rightarrow (70 + 70)/2 = 70.0$
- Calculate the upper quartile (Q3) $\rightarrow (72 + 72)/2 = 72$
- Calculate the interquartile range (IQR) $\rightarrow$ Q3 – Q1 $= 72 - 70 = 2$
- Find the upper and lower fences.
  Lower fence = Q1 – 1.5 * IQR = 70 – 1.5 * 2 = 67
  Upper fence = Q3 + 1.5 * IQR = 71.5 + 1.5 * 2 = 74.5
- Any data points lower than the lower fence and greater than the upper fence are outliers $\rightarrow$ outliers: 60; 90; 320.

Outliers

Examples

$>>$ Q1 = df['Height'].quantile(0.25)
Q3 = df['Height'].quantile(0.75)
$IQR = Q3 - Q1$

$>>$ $low\_fence = Q1 - (1.5 * IQR)$
$up\_fence = Q3 + (1.5 * IQR)$

$>>$ df[((df['Height'] < low_fence)|(df['Height'] > up_fence))]

$>>$ df = df[∼((df['Height'] < low_fence)|(df['Heigh'] > up_fence))]

Data Transformation

Label Encoding: replacing each value in a categorical column with numbers from 0 to $N-1$

Syntax (initialize)

sklearn.preprocessing.LabelEncoder()

Examples

```
>> from sklearn.preprocessing import LabelEncoder
>> encoder = LabelEncoder()
```

Label Encoding

Syntax (fit & transform)

sklearn.preprocessing.LabelEncoder().fit_transform(X)

Examples

```
>> df['Sex'] = encoder.fit_transform(df['Sex'])
```

Data Transformation

One-hot Encoding: dividing a categorical column into n number of columns with n is the total number of unique labels in that column.

Syntax (initialize)

sklearn.preprocessing.OneHotEncoder(*sparse*)

Examples

```
>> from sklearn.preprocessing import OneHotEncoder
>> encoder = OneHotEncoder(sparse = False)
```

One-hot Encoding

Syntax (fit & transform)

sklearn.preprocessing.OneHotEncoder().fit_transform(X)

Examples

```
>> name_col = 'Grade'
>> data_new = encoder.fit_transform(df[[name_col]])
>> grade_column = pd.DataFrame(data=data_new,
   columns=encoder.get_feature_names([name_col]))
>> df = pd.concat([df.drop(columns=[name_col, 'Good-looking']),
   grade_column, df['Good-looking']], axis=1)
```

Data Splitting

Syntax

sklearn.model_selection.train_test_split($X, y, test\_size, random\_state$)

Examples

```
>> from sklearn.model_selection import train_test_split
>> X = df.drop(columns = ['Good-looking', 'ID'])
   y = df['Good-looking']
>> X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3)
```

Data Scaling

Normalization: refers to the rescaling of the features to a range of $[0, 1]$

$$x_{norm}^{(i)} = \frac{x^{(i)} - x_{min}}{x_{max} - x_{min}}$$

where:

- x_{max}: the largest value of column x
- x_{min}: the smallest value of column x

Standardization: centers the columns at the mean 0 with the standard deviation 1

$$x_{std}^{(i)} = \frac{x^{(i)} - \mu_x}{\sigma_x}$$

where:

- μ_x: the mean of column x
- σ_x: the standard deviation of column x

Note:

Be careful when scaling the data

Normalization

Syntax

sklearn.preprocessing.MinMaxScaler()

Examples

```
>> from sklearn.preprocessing import MinMaxScaler
>> scaler = MinMaxScaler()
>> X_train[['Height']] = scaler.fit_transform(X_train[['Height']])
   X_test[['Height']] = scaler.fit_transform(X_test[['Height']])
```

Standardization

Syntax

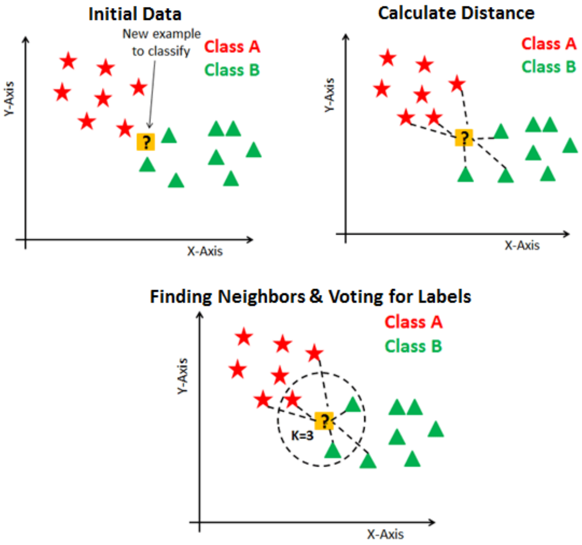
sklearn.preprocessing.StandardScaler()

Examples

```
>> from sklearn.preprocessing import StandardScaler
>> scaler = StandardScaler()
>> X_train[['Age']] = scaler.fit_transform(X_train[['Age']])
   X_test[['Age']] = scaler.fit_transform(X_test[['Age']])
```

Overview

Recall


Initial Data
New example
to classify
Class A
Class B
?
Y-Axis
X-Axis
Calculate Distance
Class A
Class B
?
Y-Axis
X-Axis
Finding Neighbors & Voting for Labels
Class A
Class B
?
K=3
Y-Axis
X-Axis

How to implement it?

Syntax (initialize)

sklearn.neighbors.KNeighborsClassifier($n\_neighbors$, p)

where:

- n_neighbors: the number of neighbors (K)
- p: power parameter for the Minkowski metric.
  - $p = 1$: Manhattan distance
  - $p = 2$: Euclidean distance
  - $p > 2$: Minkowski distance

Examples

```
>> from sklearn.neighbors import KNeighborsClassifier
>> clf = KNeighborsClassifier(n_neighbors = 3, p = 2)
```

How to implement it?

Syntax (fit)

$$\textbf{sklearn.neighbors.KNeighborsClassifier().fit}(X, y)$$

Examples

$>>$ clf.fit(X_train, y_train)

Syntax (predict)

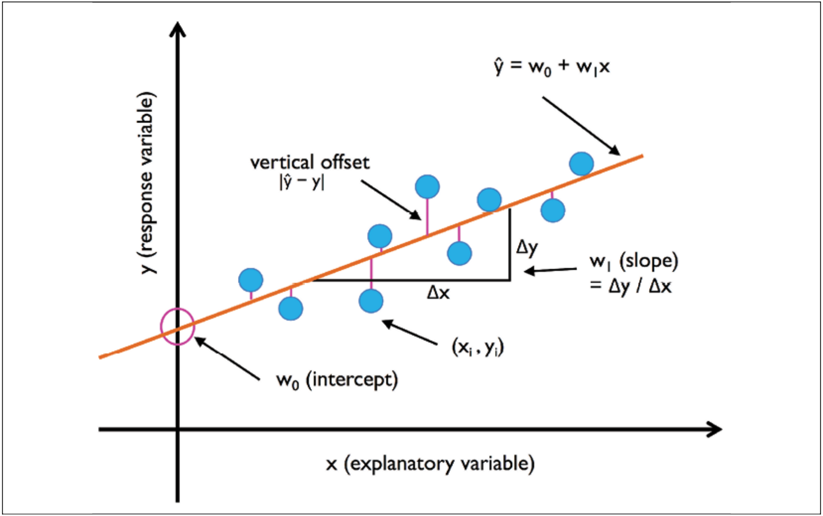
$$\textbf{sklearn.neighbors.KNeighborsClassifier().predict}(X)$$

Examples

$>>$ y_pred = clf.predict(X_test)

Overview

Recall



$\hat{y} = w_0 + w_1x$
vertical offset
$|\hat{y} - y|$
Δy
w_1 (slope)
$= \Delta y / \Delta x$
Δx
(x_i, y_i)
w_0 (intercept)
y (response variable)
x (explanatory variable)

How to implement it?

Solution 1

Syntax (initialize)

sklearn.linear_model.SGDRegression(*fit_intercept*, *random_state*)

Examples

```
>> from sklearn.linear_model import SGDRegression
>> reg = SGDRegression(fit_intercept = True, random_state = 1)
```

How to implement it?

Syntax (fit)

sklearn.linear_model.SGDRegression().fit(X, y)

Examples

$>>$ reg.fit(X_train, y_train)

Syntax (predict)

sklearn.linear_model.SGDRegression().predict(X)

Examples

$>>$ y_pred = reg.predict(X_test)

Solution 2

Syntax (initialize)

sklearn.linear_model.LinearRegression(*fit_intercept*, *random_state*)

Examples

$>>$ **from** sklearn.linear_model **import** LinearRegression

$>>$ reg = LinearRegression(*fit_intercept* = *True*, *random_state* = 1)

How to implement it?

Syntax (fit)

sklearn.linear_model.LinearRegression().fit(X, y)

Examples

$>>$ reg.fit(X_train, y_train)

Syntax (predict)

sklearn.linear_model.LinearRegression().predict(X)

Examples

$>>$ y_pred $=$ reg.predict(X_test)

Overview

Performance Metrics

Classification

- Accuracy
- Confusion matrix
- Precision and Recall
- F1 score

Regression

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)
- R-Squared

Syntax (import)

from sklearn.metrics import . . .

Examples

```
>> from sklearn.metrics import accuracy_score
>> accuracy = accuracy_score(y_pred, y_test)
   accuracy
```