32. Data Science – Machine Learning – XGBoost

Contents

1. Introduction 2

2. Why Use XGBoost? 2

3. Install XGBoost 2

4. Dataset explanation 3

5. Input and output from the Dataset 4

5.1. Input Variables (X): 4

5.2. Output Variables (y): 4

6. Label Encoding 15

7. Feature Importance with XGBoost and Feature Selection 24

32. Data Science – Machine Learning – XGBoost

1. Introduction

- ✓ XGBoost stands for eXtreme Gradient Boosting.
- ✓ XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework.
- ✓ It combines the predictions from two or more models.
- ✓ So that it get the better performance.

2. Why Use XGBoost?

- ✓ The two reasons to use XGBoost are also the two goals of the project:
  - Execution Speed.
  - Model Performance.

3. Install XGBoost

- ✓ pip install xgboost

4. Dataset explanation

- ✓ We are going to work with **pima-indians-diabetes.csv**
- ✓ This Dataset is related to health care domain.
- ✓ Pima Indians are a Native American group that lives in Mexico and Arizona, USA.
- ✓ It describes patient medical record data for Pima Indians and whether they had a diabetes within five years.
- ✓ The Pima Indian Diabetes dataset consisting of Pima Indian females 21 years and older is a popular benchmark dataset.
- ✓ It is a binary classification problem (onset of diabetes as 1 or not as 0).
- ✓ All of the input variables that describe each patient are numerical.

Feature	Description	Data type	Range
Preg	Number of times pregnant	Numeric	[0, 17]
Gluc	Plasma glucose concentration at 2 Hours in an oral glucose tolerance test (GTIT)	Numeric	[0, 199]
BP	Diastolic Blood Pressure (mm Hg)	Numeric	[0, 122]
Skin	Triceps skin fold thickness (mm)	Numeric	[0, 99]
Insulin	2-Hour Serum insulin (µh/ml)	Numeric	[0, 846]
BMI	Body mass index [weight in kg/(Height in m)]	Numeric	[0, 67.1]
DPF	Diabetes pedigree function	Numeric	[0.078, 2.42]
Age	Age (years)	Numeric	[21, 81]
Outcome	Binary value indicating non-diabetic /diabetic	Factor	[0,1]

5. Input and output from the Dataset

5.1. Input Variables (X):

- ✓ 1. Number of times pregnant
- ✓ 2. Plasma glucose concentration at 2 hours in an oral glucose tolerance test
- ✓ 3. Diastolic blood pressure (mm Hg)
- ✓ 4. Triceps skin fold thickness (mm)
- ✓ 5. 2-hour serum insulin (µIU/ml)
- ✓ 6. Body mass index (weight in kg/(height in m))
- ✓ 7. Diabetes pedigree function
- ✓ 8. Age (years)

5.2. Output Variables (y):

- ✓ 1. Class variable (0 or 1)

Program Name: Loading csv file
Input file: demo1.py
pima-indians-diabetes.csv

```
from numpy import loadtxt

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

print(dataset)
```

Output

```
[[  6.    148.     72.    ...   0.627  50.     1.   ]
 [  1.     85.     66.    ...   0.351  31.     0.   ]
 [  8.    183.     64.    ...   0.672  32.     1.   ]
 ...
 [  5.    121.     72.    ...   0.245  30.     0.   ]
 [  1.    126.     60.    ...   0.349  47.     1.   ]
 [  1.     93.     70.    ...   0.315  23.     0.   ]]
```

Program Name: Preparing input (X) and output (y) variables
Input file: demo2.py
pima-indians-diabetes.csv

```
from numpy import loadtxt

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

print("Input and output")
```

Output

```
Input and output
```

Program Name: Splitting data into train and test datasets
Input file: demo3.py
pima-indians-diabetes.csv

```
from sklearn.model_selection import train_test_split
from numpy import loadtxt

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

seed = 7

test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(
   X, y,
   test_size = test_size,
   random_state = seed
)

print("Train and Test dataset")
```

Output

```
Train and Test dataset
```

Program Name: Model creation
Input file: demo4.py
pima-indians-diabetes.csv

```
from sklearn.model_selection import train_test_split
from numpy import loadtxt
from xgboost import XGBClassifier

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

seed = 7

test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(
   X, y,
   test_size = test_size,
   random_state = seed
)

model = XGBClassifier()
print("Model created")
```

Output

```
Model created
```

Program Name: Train the model

Input file: demo5.py

pima-indians-diabetes.csv

```
from sklearn.model_selection import train_test_split
from numpy import loadtxt
from xgboost import XGBClassifier

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

seed = 7

test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(
   X, y,
   test_size = test_size,
   random_state = seed
)

model = XGBClassifier()
model.fit(X_train, y_train)
print("Model trained")
```

Output

```
Model created
```

Program Name: Model prediction
Input file: demo6.py
pima-indians-diabetes.csv

```
from sklearn.model_selection import train_test_split
from numpy import loadtxt
from xgboost import XGBClassifier

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

seed = 7

test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(
   X, y,
   test_size = test_size,
   random_state = seed
)

model = XGBClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

print(X_test)
print(predictions)
```

Output

```
[[  1.     90.     62.    ...  27.2     0.58   24.   ]
 [  7.    181.     84.    ...  35.9     0.586  51.   ]
 [ 13.    152.     90.    ...  26.8     0.731  43.   ]
 ...
 [  4.    118.     70.    ...  44.5     0.904  26.   ]
 [  7.    152.     88.    ...  50.      0.337  36.   ]
 [  7.    168.     88.    ...  38.2     0.787  40.   ]]
[0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0
 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 1 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0
 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 0 1
 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1
 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1]
```

Program Name: Model prediction
Input file: demo7.py
pima-indians-diabetes.csv

```
from sklearn.model_selection import train_test_split
from numpy import loadtxt
from xgboost import XGBClassifier

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

seed = 7

test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(
   X, y,
   test_size = test_size,
   random_state = seed
)

model = XGBClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

print(X_test)
print(predictions)
```

Output

```
[[  1.     90.     62.    ...  27.2     0.58   24.   ]
 [  7.    181.     84.    ...  35.9     0.586  51.   ]
 [ 13.    152.     90.    ...  26.8     0.731  43.   ]
 ...
 [  4.    118.     70.    ...  44.5     0.904  26.   ]
 [  7.    152.     88.    ...  50.      0.337  36.   ]
 [  7.    168.     88.    ...  38.2     0.787  40.   ]]
[0 1 1 0 1 1 0 0 1 0 1 0 1 1 0 0 0 1 0 0 0 0 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0
 0 1 0 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 1 1 1 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 1 0 1 0 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 0
 1 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0
 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 1 0 0 0 1
 1 0 0 0 1 1 0 1 1 0 0 1 0 0 1 0 0 1 1 1 0 1 0 0 0 1 1 0 0 1 0 0 1 0 0 0 1
 1 0 1 1 0 1 1 0 1 1 0 0 0 0 0 1 1 0 0 1 0 0 1 0 1 0 0 0 0 1 0 1]
```

Program Name: Check model accuracy
Input file: demo8.py
pima-indians-diabetes.csv

```
from sklearn.model_selection import train_test_split
from numpy import loadtxt
from xgboost import XGBClassifier
from sklearn.metrics import accuracy_score

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

seed = 7

test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(
   X, y,
   test_size = test_size,
   random_state = seed
)

model = XGBClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print(accuracy)
```

Output

```
0.740157
```

6. Label Encoding

- ✓ The iris flowers classification problem is an example of a problem that has a string class value.
- ✓ XGBoost cannot model this problem because output variables are strings, we need to convert output variables be numeric.
- ✓ We can easily convert the string values to integer values using the LabelEncoder.
- ✓ The three class values
  - (Iris-setosa, Iris-versicolor, Iris-virginica) are mapped to the integer values (0, 1, 2).

Program Name: Loading csv file
Input file: demo1.py
iris.csv

```
from pandas import read_csv

data = read_csv('iris.csv', header = None)
dataset = data.values

print(dataset[0:5])
```

Output

```
[[5.1 3.5 1.4 0.2 'Iris-setosa']
 [4.9 3.0 1.4 0.2 'Iris-setosa']
 [4.7 3.2 1.3 0.2 'Iris-setosa']
 [4.6 3.1 1.5 0.2 'Iris-setosa']
 [5.0 3.6 1.4 0.2 'Iris-setosa']]
```

Program Name: Data preparation
Input file: demo2.py
iris.csv

```
from pandas import read_csv

data = read_csv('iris.csv', header = None)
dataset = data.values

X = dataset[:,0:4]
Y = dataset[:,4]

print(X[0:5])
print()
print(Y[0:5])
```

Output

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.0 1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.0 3.6 1.4 0.2]]

['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa']
```

Program	Encoding flower names
Name	demo3.py
Input file	iris.csv

```python
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder

data = read_csv('iris.csv', header = None)
dataset = data.values

X = dataset[:,0:4]
Y = dataset[:,4]

label_encoder = LabelEncoder()
label_encoder = label_encoder.fit(Y)
label_encoded_y = label_encoder.transform(Y)

print(label_encoded_y)
```

Output

```
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

Program Name: Splitting the data
Input file: demo4.py
iris.csv

```
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

data = read_csv('iris.csv', header = None)
dataset = data.values

X = dataset[:,0:4]
Y = dataset[:,4]

label_encoder = LabelEncoder()
label_encoder = label_encoder.fit(Y)
label_encoded_y = label_encoder.transform(Y)

seed = 7
test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(X,
label_encoded_y, test_size=test_size, random_state = seed)

print("Data splitting done")
```

Output

```
Data splitting done
```

Program Name: Model creation
Input file: demo5.py
iris.csv

```
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

data = read_csv('iris.csv', header = None)
dataset = data.values

X = dataset[:,0:4]
Y = dataset[:,4]

label_encoder = LabelEncoder()
label_encoder = label_encoder.fit(Y)
label_encoded_y = label_encoder.transform(Y)

seed = 7
test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(X,
label_encoded_y, test_size=test_size, random_state = seed)

model = XGBClassifier()

print("Model created")
```

Output

```
Model created
```

Program Name: Model training
Input file: demo6.py
iris.csv

```
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from xgboost import XGBClassifier

data = read_csv('iris.csv', header = None)
dataset = data.values

X = dataset[:,0:4]
Y = dataset[:,4]

label_encoder = LabelEncoder()
label_encoder = label_encoder.fit(Y)
label_encoded_y = label_encoder.transform(Y)

seed = 7
test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(X,
label_encoded_y, test_size=test_size, random_state = seed)

model = XGBClassifier()
model.fit(X_train, y_train)

print("Model trained")
```

Output

```
Model trained
```

Program Name: Prediction
Input file: demo7.py
iris.csv

```
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier

data = read_csv('iris.csv', header = None)
dataset = data.values

X = dataset[:,0:4]
Y = dataset[:,4]

label_encoder = LabelEncoder()
label_encoder = label_encoder.fit(Y)
label_encoded_y = label_encoder.transform(Y)

seed = 7
test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(X,
label_encoded_y, test_size=test_size, random_state = seed)

model = XGBClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

print(X_test[0:5])
print()
print(predictions[0:5])
```

Output

```
[[5.9 3.0 5.1 1.8]
 [5.4 3.0 4.5 1.5]
 [5.0 3.5 1.3 0.3]
 [5.6 3.0 4.5 1.5]
 [4.9 2.5 4.5 1.7]]

[2 1 0 1 1]
```

Program Name: Evaluate predictions
Input file: demo8.py
iris.csv

```
from pandas import read_csv
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from xgboost import XGBClassifier

data = read_csv('iris.csv', header = None)
dataset = data.values

X = dataset[:,0:4]
Y = dataset[:,4]

label_encoder = LabelEncoder()
label_encoder = label_encoder.fit(Y)
label_encoded_y = label_encoder.transform(Y)

seed = 7
test_size = 0.33

X_train, X_test, y_train, y_test = train_test_split(X,
label_encoded_y, test_size=test_size, random_state = seed)

model = XGBClassifier()
model.fit(X_train, y_train)

predictions = model.predict(X_test)

accuracy = accuracy_score(y_test, predictions)
print("Accuracy: %.2f%%" % (accuracy * 100.0))
```

Output

```
Accuracy: 92.00%
```

7. Feature Importance with XGBoost and Feature Selection

- ✓ XGBoost is helping to get important features from the existing features
- ✓ We can see like every features how much score it having.

Program Name: Feature Importance with XGBoost
Input file: demo1.py
pima-indians-diabetes.csv

```
from numpy import loadtxt
from xgboost import XGBClassifier
from matplotlib import pyplot

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

model = XGBClassifier()

model.fit(X, y)

print(model.feature_importances_)
```

Output

```
[0.10621195 0.24240209 0.08803371 0.07818192 0.10381888
 0.14867325 0.10059208 0.13208608]
```

Program Name: Plotting the Feature Importance with XGBoost
Input file: demo2.py
pima-indians-diabetes.csv

```
from numpy import loadtxt
from xgboost import XGBClassifier
from matplotlib import pyplot

dataset = loadtxt('pima-indians-diabetes.csv', delimiter = ',')

X = dataset[:, 0:8]
y = dataset[:, 8]

model = XGBClassifier()

model.fit(X, y)

print(model.feature_importances_)

r = range(len(model.feature_importances_))
f_imp = model.feature_importances_

pyplot.bar(r, f_imp)
pyplot.show()
```

Output

[0.10621195 0.24240209 0.08803371 0.07818192 0.10381888
0.14867325 0.10059208 0.13208608]