Data Loading: First, you need to load the transaction data into your preferred programming environment. You can use libraries like Pandas in Python or read data from a database.

Data Cleaning: Clean the data by handling missing values, removing duplicates, and correcting any inconsistencies.

Data Transformation: Perform any necessary transformations, such as data type conversions or feature engineering.

Feature Scaling: If needed, scale numerical features to ensure they have similar magnitudes.

Data Encoding: Encode categorical variables, such as converting text categories into numerical values using techniques like one-hot encoding or label encoding.

Data Splitting: Split the data into training and testing sets to evaluate your model's performance.

Data Normalization: Normalize the data if required, often done for machine learning models like neural networks.

Data Exploration: Explore the data through statistics and visualization to gain insights into its characteristics.

Data Preprocessing Pipeline: It's common to create a data preprocessing pipeline to automate these steps, ensuring consistency and reproducibility.

The specific code for each step will depend on the programming language and libraries you're using. If you have a particular question or need help with a specific part of the process, feel free to ask!

```
Import pandas as pd

Import numpy as np


# Load the transaction data from a CSV file (you can replace 'transactions.csv' with your file)

Data = pd.read_csv('transactions.csv')


# Display the first few rows to inspect the data

Print(data.head())


# Preprocessing steps:

# 1. Handle missing values (if any)

Data = data.dropna()  # Drop rows with missing values


# 2. Convert date columns to datetime objects (if applicable)

Data['date'] = pd.to_datetime(data['date'])


# 3. Perform any necessary data transformations or calculations

Data['total_amount'] = data['quantity'] * data['unit_price']


# Now, your data is loaded and preprocessed.


# You can perform various analysis or visualization tasks on this preprocessed data.
```

Processamento de dados Python

Default (GPT-3.5)

User

Loading and preprocessing the transaction data with coding in python

ChatGPT

Sure, I can help you with loading and preprocessing transaction data in Python. To do this, you'll likely need some libraries like Pandas and NumPy. Here's a basic example of how you can load and preprocess transaction data:

Python

Copy code

```
Import pandas as pd

Import numpy as np


# Load the transaction data from a CSV file (you can replace 'transactions.csv' with your file)

Data = pd.read_csv('transactions.csv')


# Display the first few rows to inspect the data

Print(data.head())


# Preprocessing steps:

# 1. Handle missing values (if any)

Data = data.dropna()  # Drop rows with missing values


# 2. Convert date columns to datetime objects (if applicable)

Data['date'] = pd.to_datetime(data['date'])


# 3. Perform any necessary data transformations or calculations
```

Data['total_amount'] = data['quantity'] * data['unit_price']

Now, your data is loaded and preprocessed.

You can perform various analysis or visualization tasks on this preprocessed data.

DATA SCIENCE

+2

How to Preprocess Data in Python

Here's a step-by-step tutorial on data preprocessing implementation using Python, NumPy and Pandas.

Written byAfroz Chakure

Published on Jun. 10, 2022

How-to-preprocess-data-python

In this article, we'll prep a machine learning model to predict who survived the Titanic. To do that, we first have to clean up our data. I'll show you how to apply preprocessing techniques on the Titanic data set.

To get started, you'll need:

Python

NumPy

Pandas

The Titanic data set

What Is Data Preprocessing and Why Do We Need It?

For machine learning algorithms to work, it's necessary to convert raw data into a clean data set, which means we must convert the data set to numeric data. We do this by encoding all the categorical labels to column vectors with binary values. Missing values, or NaNs (not a number) in the data set is an annoying problem. You have to either drop the missing rows or fill them up with a mean or interpolated values.

Note: Kaggle provides two data sets: training data and results data. Both data sets must have the same dimensions for the model to produce accurate results.

HOW TO PREPROCESS DATA IN PYTHON STEP-BY-STEP

Load data in Pandas.

Drop columns that aren't useful.

Drop rows with missing values.

Create dummy variables.

Take care of missing data.

Convert the data frame to NumPy.

Divide the data set into training data and test data.

1. Load Data in Pandas

To work on the data, you can either load the CSV in Excel or in Pandas. For the purposes of this tutorial, we'll load the CSV data in Pandas.

Df = pd.read_csv('train.csv')

Let's take a look at the data format below:

>>> df.info()

<class 'pandas.core.frame.DataFrame'>

Int64Index: 891 entries, 0 to 890

Data columns (total 12 columns):

PassengerId 891 non-null int64

Survived 891 non-null int64

Pclass 891 non-null int64

Name 891 non-null object

Sex 891 non-null object

Age 714 non-null float64

SibSp 891 non-null int64

Parch 891 non-null int64

Ticket 891 non-null object

Fare 891 non-null float64

Cabin 204 non-null object

Embarked 889 non-null object

If you carefully observe the above summary of Pandas, there are 891 total rows but Age shows only 714 (which means we're missing some data), Embarked is missing two rows and Cabin is missing a lot as well. Object data types are non-numeric so we have to find a way to encode them to numerical values.

2. Drop Columns That Aren’t Useful

Let’s try to drop some of the columns which won’t contribute much to our machine learning model. We’ll start with Name, Ticket and Cabin.

Cols = [‘Name’, ‘Ticket’, ‘Cabin’]

Df = df.drop(cols, axis=1)

We dropped three columns:

>>>df.info()

PassengerId 891 non-null int64

Survived 891 non-null int64

Pclass 891 non-null int64

Sex 891 non-null object

Age 714 non-null float64

SibSp 891 non-null int64

Parch 891 non-null int64

Fare 891 non-null float64

Embarked 889 non-null object

3. Drop Rows With Missing Values

Next we can drop all rows in the data that have missing values (NaNs). Here’s how:

>>> df = df.dropna()

>>> df.info()

Int64Index: 712 entries, 0 to 890

Data columns (total 9 columns):

PassengerId 712 non-null int64

Survived 712 non-null int64

Pclass 712 non-null int64

Sex 712 non-null object

Age 712 non-null float64

SibSp 712 non-null int64

Parch 712 non-null int64

Fare 712 non-null float64

Embarked 712 non-null object

THE PROBLEM WITH DROPPING ROWS

After dropping rows with missing values, we find the data set is reduced to 712 rows from 891, which means we are wasting data. Machine learning models need data to train and perform well. So, let's preserve the data and make use of it as much as we can. More on this below.

BUILT IN EXPERT CONTRIBUTORS CAN HELP

How to Find Residuals in Regression Analysis

4. Creating Dummy Variables

Instead of wasting our data, let's convert the Pclass, Sex and Embarked to columns in Pandas and drop them after conversion.

```
Dummies = []
Cols = ['Pclass', 'Sex', 'Embarked']
For col in cols:
  Dummies.append(pd.get_dummies(df[col]))
```

Then…

Titanic_dummies = pd.concat(dummies, axis=1)

Now we've transformed eight columns wherein 1, 2 and 3 represent the passenger class.

How-to-preprocess-data-python

Finally we concatenate to the original data frame, column-wise:

Df = pd.concat((df,titanic_dummies), axis=1)

Now that we converted Pclass, Sexand Embarked values into columns, we drop the redundant columns from the data frame.

Df = df.drop(['Pclass', 'Sex', 'Embarked'], axis=1)

Let's take a look at the new data frame:

>>>df.info()

PassengerId 891 non-null int64

Survived 891 non-null int64

Age 714 non-null float64

SibSp 891 non-null int64

Parch 891 non-null int64

Fare 891 non-null float64

1 891 non-null float64

2 891 non-null float64

3 891 non-null float64

Female 891 non-null float64

Male 891 non-null float64

C 891 non-null float64

Q 891 non-null float64

S 891 non-null float64

Data Preprocessing Using Pandas and Matplotlib

MORE ON DUMMY VARIABLES

Beware of the Dummy Variable Trap in Pandas

5. Take Care of Missing Data

Everything's clean now, except Age, which has lots of missing values. Let's compute a median or interpolate() all the ages and fill those missing age values. Pandas has an interpolate() function that will replace all the missing NaNs to interpolated values.

Df['Age'] = df['Age'].interpolate()

Now let's observe the data columns. Notice Ageis now interpolated with imputed new values.

>>>df.info()

Data columns (total 14 columns):

PassengerId 891 non-null int64

Survived 891 non-null int64

Age 891 non-null float64

SibSp 891 non-null int64

Parch 891 non-null int64

Fare 891 non-null float64

1 891 non-null float64

2 891 non-null float64

3 891 non-null float64

Female 891 non-null float64

Male 891 non-null float64

C 891 non-null float64

Q 891 non-null float64

MORE PYTHON TUTORIALS

How to Find Outliers With IQR Using Python

6. Convert the Data Frame to NumPy

Now that we’ve converted all the data to integers, it’s time to prepare the data for machine learning models. This is where scikit-learn and NumPy come into play:

X= Input set with 14 attributes

Y = Small y output, in this case Survived

Now we convert our data frame from Pandas to NumPy and we assign input and output:

X = df.values

Y = df[‘Survived’].values

X still has Survived values in it, which should not be there. So we drop in the NumPy column, which is the first column.

X = np.delete(X, 1, axis=1)

MORE TUTORIALS FROM BUILT IN EXPERTS

How to Use Float in Python (With Sample Code!)

7. Divide the Data Set Into Training Data and Test Data

Now that we’re ready with X and y, let’s split the data set: we’ll allocate 70 percent for training and 30 percent for tests using scikit model_selection.

From sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

And that's all, folks. Now you can preprocess data on your own. Go on and try it for yourself to start building your own models and making predictions.

RECENT MACHINE LEARNING ARTICLES

How Data Center Shortages Could Hinder the Future of AI

What Is VAE in Stable Diffusion?

Introduction to the Beam Search Algorithm

Data Science

Expert Contributors

Machine Learning

Expert Contributors

Built In's expert contributor network publishes thoughtful, solutions-oriented stories written by innovative tech professionals. It is the tech industry's definitive destination for sharing compelling, first-person accounts of problem-solving on the road to innovation.

libraries

Import numpy as np # used for handling numbers

Import pandas as pd # used for handling the dataset

From sklearn.impute import SimpleImputer # used for handling missing data

From sklearn.preprocessing import LabelEncoder, OneHotEncoder # used for encoding categorical data

From sklearn.model_selection import train_test_split # used for splitting training and…