Data Science

Intro to Pandas

CS202 09.25.22

Pandas

Missing Data

There are a lot of times when you'll find missing data in your given data set. What are some things you can do about it?

You can drop the null data if it is an insignificant amount.

```
df.dropna()
df.dropna(axis =1)
```

The first line will let you drop rows with null value.

The second line will let you drop the columns with null value

Pandas

Missing data

If you have a significant amount of data that are unavailable then you can fill those values with something else. By using the fillna( ) method.

```
df.fillna(value = “FILL VALUE”)
```

The line above tells you that you should fill null values with “fill value”

Often times you may want to fill in the null values with the average or the median.

```
df[‘A’].fillna(value = df[‘A’].mean())
```

The line above tells you that you should fill null values in the ‘A’ column with the mean of the column.

Pandas

Groupby

Lets say you have some recurring or nested data. (We can see an example during demo time). Groupby lets you group by categories of data or recurring data.

For example if you have recurring data in a specific column, lets say you are looking at customer info and there is a column with customer id. If you have recurring customers then you can group by their id and see the activity unique to each customer.

```
df.groupby(‘customer_id’)
```

Pandas

Groupby

With the groupby method you can calculate the mean, standard deviation, Max, min, count or description

```
grouped_df = df.groupby(‘customer_id’)
grouped_df.mean()
grouped_df.std()
grouped_df.min()
grouped_df.max()
grouped_df.describe()
```

Pandas

Merging joining and concatenating

We can Merge, join or concatenate different data frames.

Using:

```
pd.concat([df1, df2, df3])
pd.concat([df1,df2,df3], axis=1)
#this lets you concat via columns
pd.merge(df1, df2, how = ‘inner’)
#you want to merge if the data sets share a column
df1.join(df2, how = ‘outer’)
```

We can see a live demo of what this would look like during demo time.

Pandas

Operations

Remember we said we can have recurring data.

We can use the unique( ) method to find out how many unique data we have in a given column:

```
df['col'].unique()
```

We can also see how many unique values there are by using nunique( ), this will show us the number of unique values in a given column.

```
df['col'].nunique()
```

We can also see how many unique values there are and how many times they show up by using value_counts( ) method.

```
df['col'].value_counts()
```

Pandas

Operation

Selecting data:

We can select data by using criteria from multiple columns:

```
newdf = df[(df['col1']>2) & (df[‘col2']==444)]
```

The above line says select from the data frame df: from column ‘col1’ all things greater than 2 and from column ‘col2’ all things that are equal to 444.

You will be using “&” and “||” for ‘and’ and ‘or’ respectively.

Pandas

Operations

We can use the .apply( ) method to apply a function any data points in the dataset. Last time during demo we've seen how we can make use of the .apply( ) method with the lambda function.

df.columns is used to see the columns of the data frame.

df.index is used to see index names of the data frame.

df.sort_values( ) is used to sort values in a data frame.

df.isnull( ) is used to see if there are any null values in a data frame.

Pandas

Reading csv and excel files

Your data set will be given to you as a csv or excel file. One thing you can do to see some examples is go to kaggle.com to see some real data that is circulated out there.

To read a csv file:

```
df = pd.read_csv(‘<name of csv file>.csv’)
```

To read a excel file:

```
df = pd.read_excel(‘<name of csv file>.xls’)
```

Pandas supports xls, *xlsx*, *xlsm*, *xlsb*, *odf*, *ods* and *odt file extensions*

Once you have uploaded your csv or excel you can start with checking the head of the data set: (of course you can check df.tail( ) as well)

```
df.head()
```