Python and Data Engineering Fundamentals

Comment

# Text	Comment one line
‘’’ Text Text ‘’’	Comment multiple lines

Variable Data Types

type(x)	Returns data type of the variable x
int(x)	Convert variable type of x to integer
float(x)	Convert variable type of x to float
str(x)	Convert variable type of x to string
bool(x)	Convert variable type of x to boolean
list()	Create an empty list
mylist = [var1, var2, var3]	Create mylist of type list with the elements : var1, var2, var3
mylist.append(x)	Add the variable to the list “mylist”
mylist.extend(x)	Concatenate the variable to the end of list “mylist”
dict()	Create an empty dictionary
Mydict={key:value}	Create a filled dictionary with key , value pair

Comparison operators

==	Equal to
Is	Equal to (Boolean evaluation)
!=	Not equal to
Is not	Not equal to (Boolean evaluation)
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

If Statement

```
if (test expression):
        Body_of_If
elif (test expression):
        Body_of_elIf
else :
        Body_of_else
```

While Loop

```
while (test expression):
        Loop_Body
```

For Loop

```
for var in sequence:
        Loop_Body
```

Range function

range (stop)	Create sequence of numbers from 0 to (stop-1)
range (start,stop)	Create sequence of numbers from start to (stop-1)
range (start, stop, step_size)	Create sequence of numbers fromstart to (stop-1) with a step size of (step_size)

Function Creation

```
def function_name (arguments):
     #statement/s
     Return value/var
```

Function Calling

```
var=function_name(arguments)
```

Special Function parameters/statements

*var_name	flexible number of parameters
Attr = value	Default attribute value
pass	Define empty function body without compilation error

Load data

pd.read_excel (“Filepath”)	Read excel file
pd.read_csv (“Filepath”)	Read comma-separated values file

Excel File reading options

sheet_name=“Sheet1”	Specify the excel sheet name
index_col=n	Assign the n^{th} column as the index column
skiprows=n	Skip the first n rows
nrows=n	Read only the first n rows
usecols=n	Read the n^{th} column only
usecols=[1,2]	Read the 2^{nd} and 3^{rd} columns only
usecols=[A:C,F]	Read the specified range (as in excel) of columns only
usecols= [“ColName”,”ColName”]	Read the specified columns name only

Dataframe

My_Dataframe	Display the Dataframe
My_Dataframe[:]	
My_Dataframe.head()	Display the first 5 rows
My_Dataframe.head(n)	Display the first n rows
My_Dataframe.tail()	Display the last 5 rows
My_Dataframe.tail(n)	Display the last n rows
My_Dataframe.loc[n]	Display the n^{th} row
My_Dataframe.loc[start:stop]	Display the specified range of rows
My_Dataframe.columns	Retrieve the Data-frame columns names
My_Dataframe.ColName	Retrieve specific column data
My_Dataframe[“ColName”]	
My_Dataframe[Condition]	Conditional selection
My_Dataframe.max()	Retrieve the maximum for all the columns
My_Dataframe.ColName.max()	Retrieve the maximum for a specific column
My_Dataframe.ColName.min()	Retrieve the minimum for a specific column
My_Dataframe.ColName.mean()	Retrieve the mean for a specific column
My_Dataframe.ColName.mode()	Retrieve the mode for a specific column
My_Dataframe.ColName.std()	Retrieve the standard deviation for a specific column
My_Dataframe.append(ExtraRow)	Append the Dataframe with a single row
My_Dataframe.copy()	Create a copy

Python and Data Engineering Fundamentals

Data cleaning

My_Dataframe.drop_duplicates()	Drop duplicated rows
My_Dataframe.isnull().sum()	Count the null values under the entire Dataframe
My_Dataframe.ColName.isnull().sum()	Count the null values under a single column
My_Dataframe[“ColName”].isnull().sum()	
My_Dataframe.dropna()	Drop the rows where at least one element is missing
My_Dataframe.fillna(value)	Fill the NaN elements with value
My_Dataframe.fillna(My_Dataframe.mode().loc[0],inplace=True)	Replace the missing elements with the most frequent element in the column
My_Dataframe.fillna(My_Dataframe ['ColName'].mean(),inplace=True)	Replace the missing elements with the column mean value

Input / output separation

Input=My_Datafram.iloc[:,:-1]	All the rows and all columns except the last one
Output=My_Datafram.iloc[:,-1]	All the rows of the last column only

Training / testing separation

Input_train, Input_test, Output_train, Output_test = train_test_split (Input, Output, test_size=value)

Ordinal Encoding

oe=OrdinalEncoder(categories=[cat])	Create ordinal encoder object, the cat is a list that defines the order of the ordinal categories
oe.fit_transform()	Map and replace each category in the attribute to its corresponding numerical value

Dummy Encoding

Enc_col=pd.get_dummies(df [['attribute_name']])	Create new attributes for encoding nominal data
df=df.drop(''attribute_name”, axis=1)	Drop the categorical feature
df=pd.concat([df, Enc_col], axis=1)	Append the new attributes that represents those categories in the column to our dataframe

Label Encoding

le = LabelEncoder()	Create label encoding object
df['att']= le.fit_transform(df['att'])	Convert the categories to numeral form

Normalization

MinMaxScaler().fit_transform (df)	Normalize(Scale) the numerical feature/ dataframe
StandardScaler().fit_transform(df)	Standardize features by removing the mean and scaling to unit variance

Data Reduction : Variance Threshold

X_variance = VarianceThreshold(threshold=.7).get_support()	Outputs a list of low/high representing the variance of each feature
PCA(n_components=0.99, whiten=True).fit_transform(X)	Create Principle Component Analysis vectors

Undersampling to Handle Imbalanced Classes

rus = RandomUnderSampler(random_state=0).fit(x,y)	Undersample the dataset with input x, and output y
X_resampled, y_resampled = rus.fit_resample(X, y)	Create undersampled data with resampled x, y

Oversampling to Handle Imbalanced Classes

ros= RandomOverSampler(random_state=0).fit(X, y)	Oversample the dataset with input x, and output y
X_resampled, y_resampled = ros.fit_resample(X, y)	Create Oversampled data with resampled x, y

Data Visualization

plt.show()	Show the plotted figure
plt.title(“The Title”)	The plot title
plt.xlabel(“X-axis label”)	The plot X-axis label
plt.ylabel(“Y-axis label”)	The plot Y-axis label
plt.figure(figsize=(value,value))	Specify the figure size
PlotData.plot(Kind='bar')	Creates a vertical bar plot
PlotData.plot(Kind='barh')	Creates a horizontal bar plot
PlotData.plot(Kind='bar', rot=degree)	Rotate the x tick labels as the provided degree
PlotData.plot(Kind='bar', color=“ColorName”)	Define the bars color
PlotData.plot(Kind='bar', color=“ColorName”, alpha=Value)	Define the bars' color intensity
PlotData.plot(Kind='Pie')	Creates a Pie chart
PlotData.plot(Kind='Pie', labeldistance=Value)	Move the pie chart labels as the value provided
PlotData.plot()	Creates a line chart
PlotData.plot(color=“ColorName”)	Define the line color
PlotData.plot(marker=“markershape”)	Define the line label marker shape
sns.heatmap(PlotData)	Creates a heatmap
sns.heatmap(PlotData, annot=True)	Add annotations to the heatmap
sns.heatmap(PlotData, cmap=“ColorName”)	Define the map color