PRINCETON AI4ALL FRAGILE FAMILIES PYTHON CHEAT SHEET

CREATING A DATAFRAME

1. df = pd.read\_csv(filename.csv) # read csv file with name filename.csv and put the contents into data frame df.
2. df = pd.DataFrame(data) # create a data frame using the data. Data can be a list, array, or a Pandas Series.

VIEWING A DATAFRAME

1. df # print the data frame
2. df.shape # print the number of rows and columns in the data frame
3. df.head(n=numrows) # show the first numrows of the data frame (default numrows=5)
4. df.tail(n=numrows) # show the last numrows of the data frame (default numrows=5)

COMBINING DATAFRAMES

1. pd.concat([df1,df2,df3],axis=1) # combine multiple dataframes with the same number (and name) of rows
2. pd.merge(df1,df2,left\_index=True,right\_index=True) # combine two dataframes with different number of rows but shared row names

ADDING ROWS OR COLUMNS (CREATING NEW FEATURES)

1. df[‘newcol’] = df[‘col1’] + df[‘col2’] # create a new column equal to the sum of col1 and col2
2. df['skills'] = df[['t5c13a','t5c13b','t5c13c']].sum(axis=1)/3 # create a new column equal to the average of three columns
3. df.append(df1.iloc[5,:]) # create a new row equal to row number 6 of df1

REMOVING ROWS OR COLUMNS

1. df = df.drop([‘col1’,’col2’], axis=1) # remove col1 and col2
2. df = df.drop([‘row1’,’row2’], axis=0) # remove row1 and row2

DATAFRAME STATISTICS

1. df.describe() # statistics of data frame
2. df.mean() # column averages
3. df[‘gpa’].mean() # average of column ‘gpa’
4. df.median() # columns median valued
5. df.std() # columns standard deviations
6. df.min() # min of columns
7. df.max() # max of columns

SELECTING PARTS OF A DATAFRAME

1. df.iloc[rownum,colnum] # select value in row number rownum and column number colnum

Examples

1. df.iloc[0,0] # get element in first row, first column
2. df.iloc[:,6] # get all rows in the sixth column
3. df.iloc[5,:] # get all columns in the sixth row
4. df.iloc[1:,3] # get all rows in the fourth column starting with the second row
5. df.iloc[1,3:] # get all columns in the second row starting with the fourth column
6. df.loc[rowname,colname]

Examples

1. df.loc[‘row5’,’income’] # get element in row named ‘row5’ and column named ‘income’
2. df.loc[3,’gpa’] # get element in row named 3 with column named ‘gpa’

# select columns named col1 and col2 from dataframe

1. df = dataframe[[ ‘col1’,’col2’ ]]
2. df = dataframe.loc[:,[ ‘col1’,’col2’ ]]
3. df = fff.pick\_ff\_variables(dataframe,[‘col1’,’col2’])

# select columns named col1 and col2 from dataframe and remove rows with NaNs and with negative values

1. df = fff.pick\_ff\_variables(dataframe,[‘col1’,’col2’],remove\_nans=True,remove\_negatives=True)

DATA IMPUTATION

1. Remove rows with NaNs
2. df.dropna(inplace=True) # remove rows with NaNs
3. Replace NaNs with value
4. df = df.fillna(value=0) # replace all NaNs with 0
5. df[‘gpa’] = df[‘gpa’].fillna(value=3) # replace all NaNs in column ‘gpa’ with 0
6. df[‘gpa’] = df[‘gpa’].fillna(value=df[‘gpa’].mean()) # replace all NaNs in column ‘gpa’ with the average gpa value
7. Remove rows based on value
8. df = df.loc[df<2,’gpa’] # only keep rows with gpa < 2

DATA VISUALIZATION

1. # Histogram
2. fff.plot\_histogram(df[‘col’],labels=’colname’,xlabel=’colname’,ylabel=’count’,title=’title’)
3. # Scatterplot
4. fff.scatterplot(df[‘colx’], df[‘coly’],xlabel=’colxname’,
5. ylabel=’colyname’,title=’title’)
6. # Pie Chart
7. percentage1 = df[df['gender']==’male’].shape[0] # how many males
8. percentage2 = df[df['gender']==’female’].shape[0]  # how many females
9. fff.one\_pie\_chart([percentage1,percentage2],label=’Gender’,pie\_labels=[‘Male’,’Female’])
10. # Bar plot
11. fff.one\_bar\_chart([percentage1,percentage2],labels=[‘Male’,’Female’])

ONE-HOT ENCODING

1. # create n binary columns based on one categorical variable with n categories
2. race = pd.get\_dummies(df['cm1ethrace'])
3. race.columns = ['White','Black','Hispanic','Other Race']

TRAIN-TEST SPLIT

1. **from** sklearn.model\_selection **import** train\_test\_split
2. # The following lines of code randomly splits both the outcome data and the predictor data into  a training and a testing set. The test\_size argument is a number between 0 and 1 that tells the function the percentage of data that should go into the test set.
3. x = data.loc[:,[‘pred1’,’pred2’,’pred3’]] # predictors
4. y = data[‘outcome’] # outcome
5. train\_x, test\_x, train\_y, test\_y = train\_test\_split(x, y, test\_size = 0.3, random\_state=100)
6. # Transform to numpy arrays
7. train\_x = fff.pandas\_to\_2d\_numpy(train\_x)
8. train\_y = fff.pandas\_to\_2d\_numpy(train\_y)
9. test\_x = fff.pandas\_to\_2d\_numpy(test\_x)
10. test\_y = fff.pandas\_to\_2d\_numpy(test\_y)

LINEAR REGRESSION

1. # import the linear\_model function from sklearn
2. **from** sklearn **import** linear\_model
3. # intialize the model
4. reg = linear\_model.LinearRegression()
5. # fit the model ON THE TRAINING DATA (important!)
6. reg.fit(train\_x, train\_y)
7. # Print the trained model coefficients
8. # c0 is equal to the model intercept
9. **print**("c0 = ",reg.intercept\_)
10. # the other coefficients tells us how the features affect the outcome
11. i=1
12. **for** c **in** reg.coef\_[0]:
13. **print**('c'+str(i)+'=',c)
14. i+=1
15. # predict the TESTING DATA
16. pred\_y = reg.predict(test\_x)
17. # Now, we can quantify how good our model is at predicting new values by computing R2 score
18. R2 = sklearn.metrics.r2\_score(test\_y, pred\_y)
19. **print**('mean root of square error : {}'.format(R2))