CZ1115 Lab Test Reference

What to take note of from Exercises 1 - 3

Exercise 1

Problem 1

Let labData be our dataset

- **labData.shape:** get row/column dimensions of dataset
- **labData.describe():** basic statistics for Numeric Variables
  - Careful, a variable that *looks* numeric may actually be categorical, as levels of categorical variables may often be encoded as numbers
  - labData.describe() != labData.info()
- **labData.info():** Prints information about a DataFrame (index dtype & columns, non-null values and memory usage)

Problem 2

Let labHTML be our dataset and labTable be our target table

- **labHTML = pd.read_html('...'):** import dataset from html page
- **print(type(labHTML)):** prints the type of the dataset (usually <class 'list'>)
- **print(len(labHTML)):** prints the number of tables
- **labHTML[0]:** Check each table in the dataset to identify the one that we want to extract; vary the index from 0 to 1, 2, 3 etc. to check each table parsed from the HTML document
- Ways to extract for example, **top 20 rows** of the dataframe
  - labTable.iloc[:20]
  - labTable.head(20)

Exercise 2

Problem 1

Let labDataNum be our Numeric Dataset

- **labDataNum = labData.loc[:, labData.dtypes == np.int64]:** pythonic way of extracting from labData all the int datatype variables
    - NOTE: loc and iloc is different! Refer to link for explanation and example:
        - https://stackoverflow.com/questions/31593201/how-are-iloc-and-loc-different
- **labDataNum = labData.select_dtypes(include = np.int64):** cleaner Pandas way to extracting from labData all the int datatype variables
- Read the description txt to check if numeric values is really numeric
    - After checking drop the non-Numeric variables (axis = 1) from the dataframe using the following code:

      labDataNum = labDataNum.drop(['MSSubClass',...,'YrSold'], axis = 1)

Problem 2

Let labItem be our first variable & labItem2 be our second variable

- **labItem = pd.DataFrame(labDataNum['lab_item'])**
- **Box Plot:**
  - f = plt.figure(figsize=(24, 4))
  - sb.boxplot(data = labItem, orient = "h")
- **Histogram:**
  - f = plt.figure(figsize=(24, 12))
  - sb.histplot(data = labItem)
- **Violin Plot:**
  - f = plt.figure(figsize=(24, 12))
  - sb.violinplot(data = labItem, orient = "h")
- **For visual comparison of 2 variables**

```
# Set up matplotlib figure with three subplots
        f, axes = plt.subplots(2, 3, figsize=(24, 12))
# Plot the basic uni-variate figures for labItem
        sb.boxplot(data = labItem, orient = "h", ax = axes[0,0])
        sb.histplot(data = labItem, ax = axes[0,1])
        sb.violinplot(data = labItem, orient = "h", ax = axes[0,2])
# Plot the basic uni-variate figures for labItem2
        sb.boxplot(data = labItem2, orient = "h", ax = axes[1,0])
        sb.histplot(data =labItem2, ax = axes[1,1])
        sb.violinplot(data = labItem2, orient = "h", ax = axes[1,2])
# Create a joint dataframe by concatenating the two variables
        jointDF = pd.concat([labItem2, labItem], axis = 1).reindex(labItem2.index)
# Draw jointplot of the two variables in the joined dataframe
        sb.jointplot(data = jointDF, x = "lab_item", y = "lab_item2", height = 12)
# Calculate the correlation between the two columns/variables
        jointDF.corr()
        sb.heatmap(jointDF.corr(), vmin = -1, vmax = 1, annot = True, fmt=".2f")
```

Exercise 3

Problem 1

Let labData be our dataset

- **Extract required variables from dataset**
    - labNumData = pd.DataFrame(labData[['labitem1', ... , 'labitemn']])
- **For Loop to create statistical graphs for each variable**

```
# Draw the distributions of all variables
f, axes = plt.subplots(5, 3, figsize=(18, 20))

count = 0
for var in labNumData:
    sb.boxplot(data = labNumData[var], orient = "h", ax = axes[count,0])
    sb.histplot(data = labNumData[var], ax = axes[count,1])
    sb.violinplot(data = labNumData[var], orient = "h", ax = axes[count,2])
    count += 1
```

- **Formula for the box-and-whiskers plot end-points to find the outliers**

```
# Calculate the quartiles
Q1 = labNumData.quantile(0.25)
Q3 = labNumData.quantile(0.75)

# Rule to identify outliers
rule = ((labNumData < (Q1 - 1.5 * (Q3 - Q1))) | (labNumData > (Q3 + 1.5 * (Q3 - Q1))))

# Count the number of outliers
rule.sum()
```

- **Correlation between the variables, followed by all bi-variate jointplots**

```
# Correlation Matrix
print(houseNumData.corr())

# Heatmap of the Correlation Matrix
f = plt.figure(figsize=(12, 12))
sb.heatmap(houseNumData.corr(), vmin = -1, vmax = 1, linewidths = 1,
    annot = True, fmt = ".2f", annot_kws = {"size": 18}, cmap = "RdBu")

# Draw pairs of variables against one another
sb.pairplot(data = houseNumData)
```

- **Best Predictor** has the Highest Correlation & Strong Linearity

Problem 2

Let labCatData be our extracted categorical variables from dataset

- Fix the data types of the first four variables to convert them to categorical.

```
labCatData['MSSubClass'] = labCatData['MSSubClass'].astype('category')
labCatData['Neighborhood'] = labCatData['Neighborhood'].astype('category')
labCatData['BldgType'] = labCatData['BldgType'].astype('category')
labCatData['OverallQual'] = labCatData['OverallQual'].astype('category')
```

- **Check the Variables Independently**
    - Summary Statistics + Statistical Visualisations

```
labCatData.describe()

# apply to all the different variables
sb.catplot(y = 'MSSubClass', data = labCatData, kind = "count", height = 8)
...
```

- **Joint heatmaps** of the important bi-variate relationships

```
# Distribution of BldgType across MSSubClass
f = plt.figure(figsize=(20, 8))
sb.heatmap(labCatData.groupby(['BldgType', 'MSSubClass']).size().unstack(),
linewidths = 1, annot = True, fmt = 'g', annot_kws = {"size": 18}, cmap = "BuGn")
```

- **Check the effect of the Variables on Main Comparison Variable (e.g. SalePrice)**
    - Create a joint DataFrame by concatenating SalePrice to labCatData

```
# create a saleprice dataframe
saleprice = pd.DataFrame(labData['SalePrice'])

# concatenate saleprice with labCatData
labCatSale = pd.concat([labCatData, saleprice],
sort = False, axis=1).reindex(index=labCatData.index)
```

- **Check the distribution** of SalePrice across different MSSubClass

```
f = plt.figure(figsize=(16, 8))
sb.boxplot(x = 'MSSubClass', y = 'SalePrice', data = labCatSale)

# rotate the x axis labels
plt.xticks(rotation=90);
```

- **Best Predictor** has the Highest Variation in the comparison variable across levels

Exercise 4 & 5

I feel that for these 2 exercises, best to refer to actual .ipynb files, compare prof's answer with yours :)

Feel free to edit here though!

Checklist

What you should have in your thumbdrive :)

Filename	Check?✓
Exercise1_Solution.ipynb	
Exercise2_Solution.ipynb	
Exercise3_Solution.ipynb	
Exercise4_Solution.ipynb	
Exercise5_Solution.ipynb	
M1 DataAcquisition.ipynb	
M2 BasicStatistics.ipynb	
M2 ExploratoryAnalysis.ipynb	
M3 LinearRegression.ipynb	
M4 ClassificationTree.ipynb	
Exercise 1 - 3 Notes.pdf	
train.csv (in case you need to run the solution files)	