# Discussion_1

September 27, 2022

## 1 ECS 171 Discussion 1

`Date`: Sep 27, 2022

`TA`: Dechen Gao (dcgao@ucdavis.edu)

The goal of our discussion:

- Coding of ML algorithms related to homeworks and lectures
- This week: how to set up environment for your homeworks
- Next week: linear regression

---

### 1.1 Outlines

- What is Jupyter?
- Why we need Jupyter?
- Installation: pip & conda
- Colab & Kaggle usage
- Notebook demos
- Export your notebooks

### 1.2 Jupyter Notebook

#### 1.2.1 Pros & Cons

Interactive (Jupyter) vs. non-interactive.

1. Pros
   - Easy debugging and data exploration.
   - Rich visualization and presentation.
2. Cons
   - Bad version control.
   - Not modular.

#### 1.2.2 Use Jupyter Locally

- **Installation**
  - pip install (docs)
  - conda (docs)
- **Configuration (Optional)**

  - Generate and modify configuration files, or use flags when calling Jupyter
    - e.g., `--notebook-dir`, where to locate your root directory
    - e.g., `--port`, specify a port number for Jupyter server
- **Client**
  - `VS Code`: with Jupyter extension that pulls up Jupyter automatically
  - `Web browser`: you need to pull up Jupyter manually

### 1.2.3 Use Jupyter Remotely

**Remote options**

- Colab
  - Web browser is all you need to access Notebook.
  - File system is a different (talk about this later).
  - CPU/GPU and RAM limited.
- Kaggle
  - A competition platform that provides notebook/algorithm resources, datasets.
  - Can access Kaggle datasets easily or you can upload and construct your own.
- SSH
  - expose a Jupyter port on server that can be accessed via your SSH client.
  - One option when you need better performance.

**Colab File System** Here are file access demos and our discussion demo.

By default you only have temporary file access during each session, and therefore you can only upload again when the session restarts.

However, you can mount Google Drive and upload dataset and Python modules to Google Drive.

```
# Mount Google Drive
from google.colab import drive
drive.mount('/content/gdrive')
# Optional. Change root directory. Use !pwd to verify
import os
os.chdir("/content/gdrive/My Drive/Colab Notebooks/")
# Optional. Append to system paths for interpreter to search for modules
import sys
sys.path.append('/content/gdrive/')
```

**Kaggle Demo**

- Explore public datasets and manage your own Kaggle Datasets
- Access datasets in your notebook

### 1.2.4 Homework Submission (Tentative)

Upload both your Jupyter source and exported PDFs (recommended for HW1) to Gradescope.

- **Required**: source code `.ipynb`
- **Optional**: exported PDF (extra bonus for HW1) plus sources

Why PDF? We grade on Gradescope where PDFs are well supported. We can view comments, and see where points are deducted in PDFs.

**Solution 1: Export to PDFs** One line command

```
jupyter nbconvert --execute --to pdf your_notebook.ipynb
```

PDF rendering requires pandoc, nbconvert, and LaTex support.

```
# Install pandoc
choco install pandoc    # For Windows
brew install pandoc     # For MacOS
# Install converting tool
pip install nbconvert   # or conda install
```

**Solution 2: If PDF is not working**

- Use the following commands to convert iPython file to HTML first

```
jupyter nbconvert --execute --to html your_notebook.ipynb
```

Open HTML with browser and "Print" in PDF format.

- For Jupyter Notebook opened in web browser, select from top bar `File` and `Download as` PDF.

**If PDF format is messed up**, please first download HTML and then print HTML to PDF in browser.

## 1.3 GitHub

### 1.3.1 Client

- Git shell
- GUI client, e.g., GitHub Desktop

### 1.3.2 Basic Usage

- Create your account and repo
- Syncronize your local code sources with remote Git repo
- Create & merge branches (`git branch`)

---

## 1.4 Markdown Cell Demo

## 1.5 Heading 1

### 1.5.1 Heading 2

**Heading 3**

#### Heading 4

I'm **bold text**.

I'm *italicized text*.

> Blockquotes are here!

Hyperlink: Canvas Page.

```
# Code highlighting
print('Hello, world')
```

Formula:

- $\sum_{x=a}^{b} f(x)$
- $\left(\int_{a}^{b} f(x)\, dx\right)$

---

### 1.5.2 Code Section

```
[1]: print('Hello, world!')
```

```
Hello, world!
```

```
[2]: !echo "Hello, world!"
```

```
"Hello, world!"
```

## 1.6 Python Packages

- We will use `pandas` a lot in homeworks to manipulat and and analyse datasets.
- For visualization, we have `matplotlib`, `seaborn`, and builtin in packages like `pandas`.

### 1.6.1 Usage

Use whatever you prefer and look into usage and docs online.

We will also have more demos coming in the next discussions for data manipulation and visualization.

- pandas docs
- matplotlib docs
- seaborn docs

```
[3]: import pandas as pd
     data = pd.read_csv('data.csv')
     data
```

```
[3]:     cache-misses_1s  node-loads_1s  branch-misses_1s  branch-load-misses_1s  \
     0          37514691        2822421          10070509               10559671
     1          49901539        3999875          15983066               15905409
     2          50400281        3256093          15329054               15610305
     3          33857600        2281202          12144856               12142687
```

```
4            48650176        3510185          16656488               16676601
…               …            …               …                    …
2995         30437560        2326933          12585063               12241054
2996         35121159        2576033          14087715               13595659
2997         34968122        2696992          14058112               13734181
2998         29050123        2382281          11980752               12213550
2999         31203289        2482180          12597138               12576162

      LLC-store-misses_1s  branch-loads_1s  L1-dcache-stores_1s  \
0                 4972315        875088331            661114306
1                 7812096        897853137            756436026
2                 7440166        909703981            734120225
3                 6133250        898523534            722133401
4                 6020216        938655692            790604661
…                   …              …                  …
2995              5041214        961290381            807913282
2996              5959816        993575253            861158899
2997              4591435        955564749            820924827
2998              5229134        965928905            788854210
2999              4960137       1000022539            835397449

      L1-icache-load-misses_1s  branch-instructions_1s  iTLB-loads_1s  …  \
0                     57139716               825641176        1274640  …
1                     73999248               983561267        1721115  …
2                     70138841               948103570        1738119  …
3                     58023478               924673885        1429445  …
4                     69916162              1039331799        1558873  …
…                        …                     …             …  …
2995                  50286201              1012821501        1057812  …
2996                  54518991              1050605365         895227  …
2997                  61481334              1047588424        1282742  …
2998                  62116174               960454788        1293510  …
2999                  53078400              1000095628         963997  …

      L1-icache-load-misses_5s  branch-instructions_5s  iTLB-loads_5s  \
0                       869593                 1966467          10449
1                       923435                48598163          20217
2                       454348                 1932445           8099
3                      1445012                 9134928          10191
4                      1011623                42037945          29691
…                        …                     …             …
2995                   4508220                 6894462          70212
2996                   5187666                 9620870          97990
2997                   2892833                 4103961          48325
2998                  17715975               122466994         103075
2999                   4866974                 6441573          78765
```

```
      iTLB-load-misses_5s  dTLB-store-misses_5s  dTLB-load-misses_5s  \
0                   23199                  2277                26126
1                   23926                  9173                36467
2                   11133                  1097                16018
3                   14334                  2200                 9918
4                   32148                  5706                23413
…                     …                    …                    …
2995                98023                  9604               111084
2996               117748                 10511               132867
2997                59845                  5794                72812
2998               249578                 86510               321995
2999               100181                  9085               115750

      dTLB-stores_5s  node-stores_5s  L1-dcache-load-misses_5s  label
0            1212879           16430                    261616      1
1           18616934          116077                   5774113      1
2            1322034           11231                    169059      1
3            5576265           21731                   1755962      1
4           25445021          119196                   7101670      1
…                …               …                        …     …
2995         5819375           78739                   1210270     30
2996         9296473           74737                   1528661     30
2997         3867988           43216                    728741     30
2998       100615883         1018327                  11113398     30
2999         6439933           78603                   1215627     30

[3000 rows x 81 columns]
```

```
[4]: # Select a few columns
data = data[['cache-misses_1s', 'branch-misses_1s', 'LLC-store-misses_1s',␣
 ↪'branch-instructions_1s', 'iTLB-loads_1s', 'label']]
data
```

```
[4]:       cache-misses_1s  branch-misses_1s  LLC-store-misses_1s  \
0            37514691          10070509              4972315
1            49901539          15983066              7812096
2            50400281          15329054              7440166
3            33857600          12144856              6133250
4            48650176          16656488              6020216
…                 …                 …                   …
2995         30437560          12585063              5041214
2996         35121159          14087715              5959816
2997         34968122          14058112              4591435
2998         29050123          11980752              5229134
2999         31203289          12597138              4960137

      branch-instructions_1s  iTLB-loads_1s  label
```

```
0                825641176      1274640     1
1                983561267      1721115     1
2                948103570      1738119     1
3                924673885      1429445     1
4               1039331799      1558873     1
…                      …            …     …
2995            1012821501      1057812    30
2996            1050605365       895227    30
2997            1047588424      1282742    30
2998             960454788      1293510    30
2999            1000095628       963997    30

[3000 rows x 6 columns]
```

[5]:
```
data.shape
```

[5]:
```
(3000, 6)
```

[6]:
```
# What's in data?
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 6 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   cache-misses_1s        3000 non-null   int64
 1   branch-misses_1s       3000 non-null   int64
 2   LLC-store-misses_1s    3000 non-null   int64
 3   branch-instructions_1s 3000 non-null   int64
 4   iTLB-loads_1s          3000 non-null   int64
 5   label                  3000 non-null   int64
dtypes: int64(6)
memory usage: 140.8 KB
```

[7]:
```
data.describe()
```

[7]:
```
       cache-misses_1s  branch-misses_1s  LLC-store-misses_1s  \
count     3.000000e+03      3.000000e+03         3.000000e+03
mean      3.574264e+07      1.320227e+07         5.776213e+06
std       7.392098e+06      2.180458e+06         1.223957e+06
min       6.707911e+06      5.306539e+06         9.561360e+05
25%       3.119621e+07      1.185055e+07         4.980937e+06
50%       3.365164e+07      1.256955e+07         5.609685e+06
75%       3.777566e+07      1.393846e+07         6.375148e+06
max       6.644830e+07      2.228168e+07         1.157133e+07
```

```
       branch-instructions_1s  iTLB-loads_1s        label
count            3.000000e+03   3.000000e+03  3000.000000
mean             9.891446e+08   1.098288e+06    15.500000
std              1.098516e+08   3.104263e+05     8.656884
min              3.090067e+08   2.007580e+05     1.000000
25%              9.139577e+08   8.916890e+05     8.000000
50%              9.727026e+08   1.039418e+06    15.500000
75%              1.044031e+09   1.216392e+06    23.000000
max              1.374149e+09   2.516269e+06    30.000000
```

[8]:

```python
import seaborn as sns
from matplotlib import pyplot as plt
# Color style
sns.set_style('whitegrid')
# Set figure size
plt.figure(figsize=(8, 6))
sns.scatterplot(data=data, x='branch-instructions_1s', y='cache-misses_1s')
```

[8]:

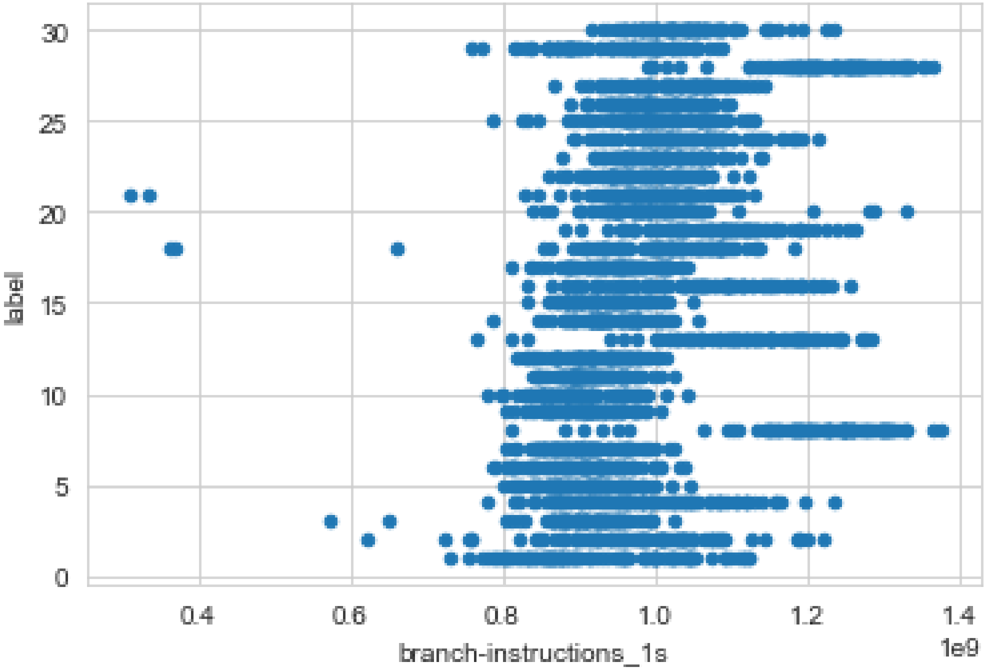
```
<matplotlib.axes._subplots.AxesSubplot at 0x24f7fc65cf8>
```

[9]:

```python
# Or you can use pandas builtin to plot
data.plot.scatter(x='branch-instructions_1s', y='label')
```

[9]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x24f7ff59f60>
```

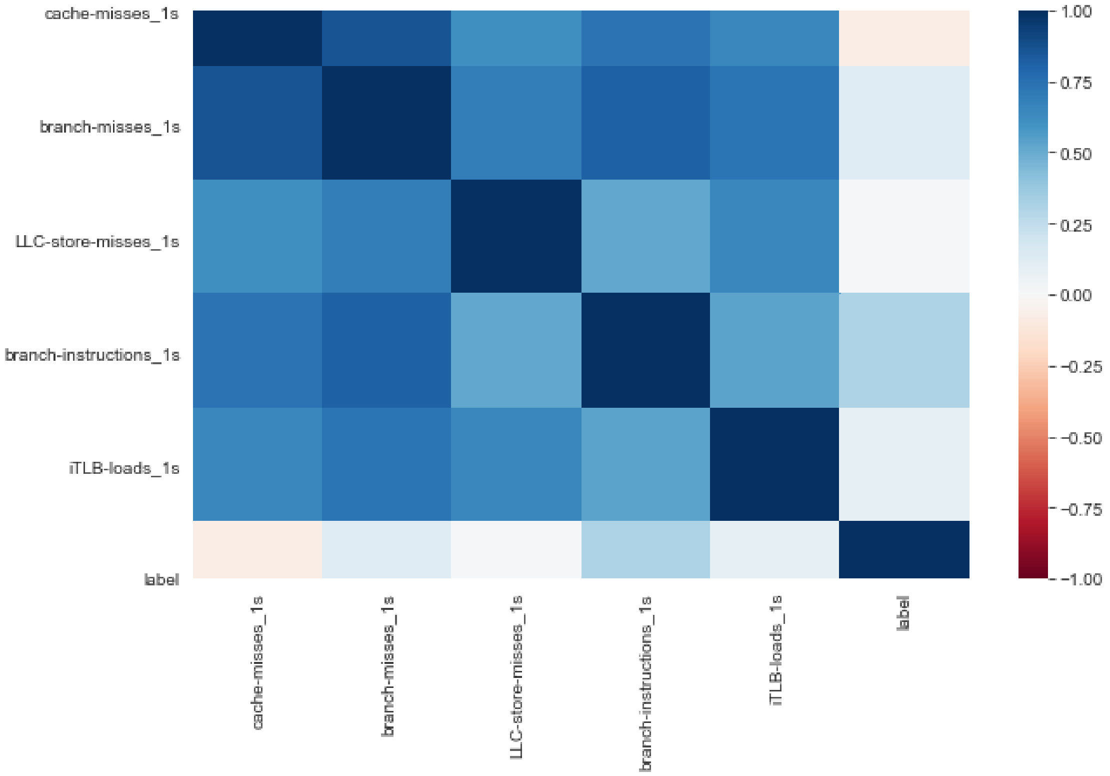


```
[10]: # Pearson correlation by default in pandas
      corr = data.corr()
      # This is an example using pandas built-in visualization
      # Determine the background colors with values in cells
      corr.style.set_precision(3).background_gradient(cmap='RdBu', vmin=-1, vmax=1)
```

```
[10]: <pandas.io.formats.style.Styler at 0x24f3cfed860>
```

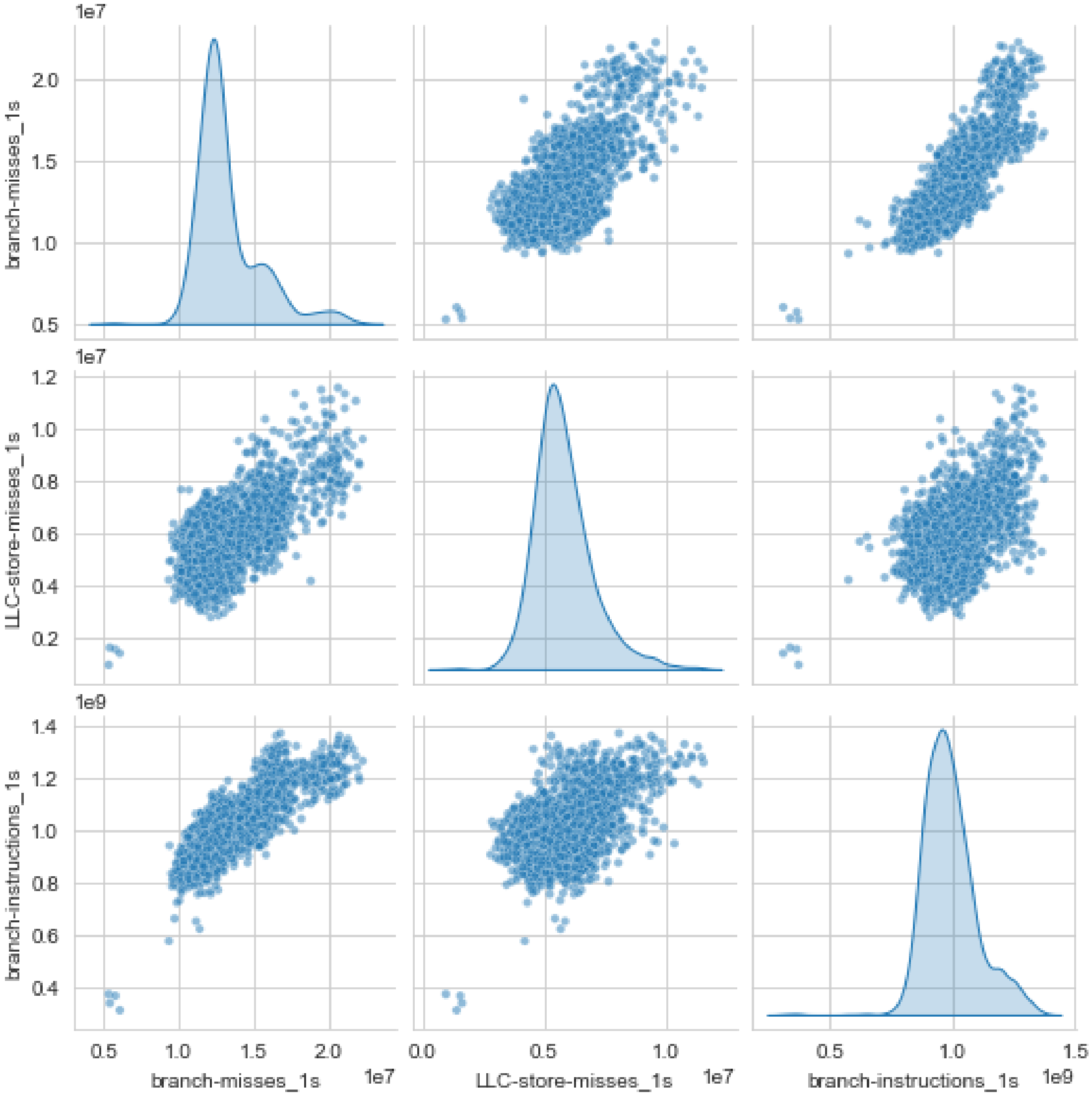
```
[11]: # Let's use Seaborn to do the heatmap
      plt.figure(figsize=(10, 6))
      sns.heatmap(corr, vmin=-1, vmax=1, center=0, annot=False, cmap= 'RdBu')
```

```
[11]: <matplotlib.axes._subplots.AxesSubplot at 0x24f679666d8>
```



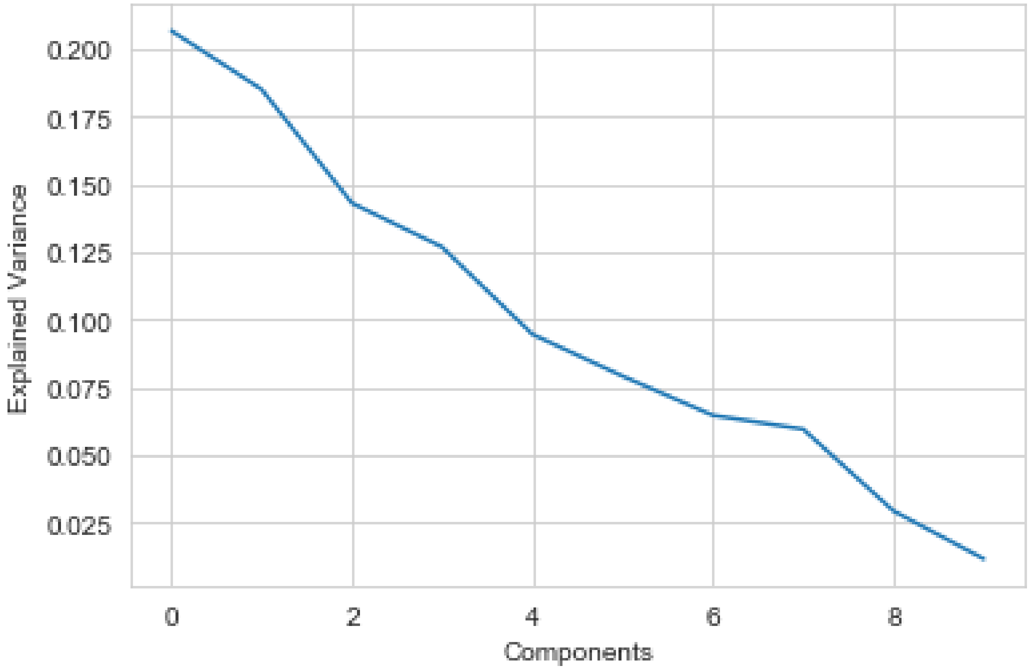
```
[12]: sns.pairplot(data=data,
                   # Use kde for the diagonal subplots
                   vars=['branch-misses_1s', 'LLC-store-misses_1s',␣
       ↪'branch-instructions_1s'],
                   diag_kind='kde',
                   plot_kws=dict(
                       size=.5,
                       alpha=.5,
            ))
```

[12]: <seaborn.axisgrid.PairGrid at 0x24f3d49cef0>



[13]:
```
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

df = pd.DataFrame(data=np.random.normal(0, 1, (20, 10)))

df_normalized=(df - df.mean()) / df.std()
pca = PCA(n_components=df.shape[1])
```

```
pca.fit(df_normalized)

loadings = pd.DataFrame(pca.components_.T,
columns=['PC%s' % _ for _ in range(len(df_normalized.columns))],
index=df.columns)

plt.plot(pca.explained_variance_ratio_)
plt.ylabel('Explained Variance')
plt.xlabel('Components')
plt.show()
```


Explained Variance
Components