EE 046202 - Technion - Unsupervised Learning & Data Analysis

Computer Assignment 1 - Statistics & Dimensionality Reduction

Due Date: 27.12.2020

Agenda

- Statistics
  - Descriptive Statistics
  - Point Estimation
  - Confidence Intervals
  - Hypothesis Testing
- Dimensionality Reduction
  - Robustness to Noise & Outliers
  - Comparing Different Methods

Use as many cells as you need

אפשר גם לכתוב בעברית, אבל עדיף באנגלית

- Code Tasks are denoted with:
- Questions (which you need to answer in a Markdown cell) are denoted with:

Students Information

- Fill in

Name	Campus Email	ID
Student 1	student_1@campus.technion.ac.il	123456789
Student 2	student_2@campus.technion.ac.il	987654321

Submission Guidelines

- Maximal garde: **100**.
- Submission only in **pairs**.
  - Please make sure you have registered your group in Moodle (there is a group creation component on the Moodle where you need to create your group and assign members).
- **No handwritten submissions.** You can choose whether to answer in a Markdown cell in this notebook or attach a PDF with your answers.
- SAVE THE NOTEBOOKS WITH THE OUTPUT, CODE CELLS THAT WERE NOT RUN WILL NOT GET ANY POINTS!
- What you have to submit:
  - If you have answered the questions in the notebook, you should submit this file only, with the name: `ee046202_wet1_id1_id2.ipynb` .
  - If you answered the questionss in a different file you should submit a `.zip` file with the name `ee046202_wet1_id1_id2.zip` with content:
    - `ee046202_wet1_id1_id2.ipynb` - the code tasks
    - `ee046202_wet1_id1_id2.pdf` - answers to questions.
  - No other file-types (`.py` , `.docx` ...) will be accepted.
- Submission on the course website (Moodle).
- **Latex in Colab** - in some cases, Latex equations may no be rendered. To avoid this, make sure to not use *bullets* in your answers ("* some text here with Latex equations" -> "some text here with Latex equations").

Working Online and Locally

- You can choose your working environment:
    1. `Jupyter Notebook`, **locally** with [Anaconda (https://www.anaconda.com/distribution/)](https://www.anaconda.com/distribution/) or **online** on [Google Colab (https://colab.research.google.com/)](https://colab.research.google.com/)
        - Colab also supports running code on GPU, so if you don't have one, Colab is the way to go. To enable GPU on Colab, in the menu: `Runtime` $\rightarrow$ `Change Runtime Type` $\rightarrow$ `GPU`.
    2. Python IDE such as [PyCharm (https://www.jetbrains.com/pycharm/)](https://www.jetbrains.com/pycharm/) or [Visual Studio Code (https://code.visualstudio.com/)](https://code.visualstudio.com/).
        - Both allow editing and running Jupyter Notebooks.
- Please refer to `Setting Up the Working Environment.pdf` on the Moodle or our GitHub ([https://github.com/taldatech/ee046202-unsupervised-learning-data-analysis (https://github.com/taldatech/ee046202-unsupervised-learning-data-analysis)](https://github.com/taldatech/ee046202-unsupervised-learning-data-analysis)) to help you get everything installed.
- If you need any technical assistance, please go to our Piazza forum (`wet1` folder) and describe your problem (preferably with images).

Tip

If you find it more convenient, you can copy the section to a new cell, and answer the question or rite the code just right below it. For example:

Question 0

1. What is the best course in the Technion?
2. Why does no one pick Bulbasaur as first pokemon?
3. Why is there no superhero named Catman?

Answers - Q0

Q0 - Section 1

- Q: What is the best course in the Technion?

```
In [ ]: print("ANAM!")
```

Q0 - Section 2

- Q: Why does no one pick Bulbasaur as first pokemon?

It is really a riddle....

Q0 - Section 3

- Q: Why is there no superhero named Catman?

I got nothing.

Keyboard Shortcuts

- Run current cell: **Ctrl + Enter**
- Run current cell and move to the next: **Shift + Enter**
- Show lines in a code cell: **Esc + L**
- View function documentation: **Shift + Tab** inside the parenthesis or `help(name_of_module)`
- New cell below: **Esc + B**
- Delete cell: **Esc + D, D** (two D's)

Part 1 - Statistics -Prologue - US 2016 Elections

In this exercise we are going to analyze the 2016 presidential elections in the United States of America. The data is located in `./election-context-2018.csv` .

The complete details of the fields can be found here: https://github.com/MEDSL/2018-elections-unoffical/blob/master/election-context-2018.md (https://github.com/MEDSL/2018-elections-unoffical/blob/master/election-context-2018.md)

Fields:

- **state** - state name
- **county** - county name
- **trump16** - presidential candidate (Trump) vote totals
- **clinton16** - presidential candidate (Clinton) vote totals
- **otherpres16** - presidential candidate (Other) vote totals
- **total_population** - total population
- **cvap** - citizen voting-age population (or: how many citizens were allowed to vote)
- **white_pct** - non-Hispanic whites as a percentage of *total population*
- **black_pct** - non-Hispanic blacks as a percentage of *total population*
- **hispanic_pct** - Hispanics or Latinos as a percentage of *total population*
- **nonwhite_pct** - non-whites as a percentage of *total population*
- **foreignborn_pct** - foreign-born population as a percentage of *total population*
- **female_pct** - females as a percentage of *total population*
- **median_hh_inc** - *median* household income in the past 12 months
- **clf_unemploy_pct** - unemployed population in labor force as a percentage of total population in civilian labor force
- **lesshs_pct** - population with an education of less than a regular high school diploma as a percentage of total
- **lesscollege_pct** - population with an education of less than a bachelor's degree as a percentage of total population

Let's have a look at the data

In [ ]:
```python
# imports for the exrcise - part 1
# you can add more if you wish (but it is not really needed)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import scipy
import seaborn as sns
```

In [ ]:
```python
df = pd.read_csv('./data/election-context-2018.csv')
df = df[['state', 'county','trump16', 'clinton16', 'otherpres16','total_population',
         'cvap', 'white_pct', 'black_pct', 'hispanic_pct', 'nonwhite_pct', 'foreignborn_pct',
         'female_pct', 'median_hh_inc', 'clf_unemploy_pct', 'lesshs_pct', 'lesscollege_pct']]
df
```

Task 1 - Preprocessing

We don't actually care about the counties, and thus we will perform the following preprocessing steps:

- For each state, remove counties that has NaN in at least one of their fields (use `df.dropna()`)
- For each state, *sum* all the countable fields (votes, total population, cvap...)
- For each state, first change percentage fields (*_pct*) to numbers, and then *sum* for each state and finally re-calculate the percentage out of the new total population. Note that the percentage is out of the *total_population*.
- For each state, calculate the *weighted median* of the the household median and then calculate the mean (average of medians)
  - We want the household median for each state, and in order to that we need to compute the weighted median of each county ($median * \text{total\_population}$) and then the best we can do (as we don't that data available) is calculate the mean.
- Add a new column called `trump16_pct` - the precentage of votes Trump got in each state.
- You should end up with a `DataFrame` similar to the original (cell above), with the extra column, but with less rows. The name of the new DataFrame should be `prep_df` instead of `df` .

Hints: `pandas` has great tools for doing all of this. Check out: `df[df['state']=='Alabama']` , `df.uniuqe()` , `df.sum()` , `df.append()` ...

In [ ]:
```python
# your code here - you can use as many cells as you need
```

Question 1 - Descriptive Statistics

Run the following command (code cell below): `fig = sns.pairplot(data=prep_df[['trump16_pct', 'median_hh_inc', 'lesshs_pct', 'black_pct', 'white_pct']])` and answer:

- Explain, in general, what does this plot show.
- What are the points on the graphs?
- Identify interesting trends.
- If the winner of the elections was decided based purely in the number of voters fot each candidate, who would have won the elections in 2016? (Hint: use `prep_df.sum()`)

Note: please refrain from expressing personal opinion, we are merely analyzing the data.

```
In [ ]: fig = sns.pairplot(data=prep_df[['trump16_pct', 'median_hh_inc', 'lesshs_pct', 'black_pct', 'white_pct']])
```

Task 2 - Point Estimates

As learned in class, point estimates are estimates of population parameters based on sample data. For instance, if we wanted to know the average age of registered voters in the U.S., we could take a survey of registered voters and then use the average age of the respondents as a point estimate of the average age of the population as a whole. The average of a sample is known as the sample mean. The sample mean is usually not exactly the same as the population mean. This difference can be caused by many factors including poor survey design, biased sampling methods and the randomness inherent to drawing a sample from a population

In this task, we will go back to the original data, `df` in the code, and we will treat each *county* as one vote and we will try to estimate the **mean** *median income* of voters.

- Constant the seed (`np.random.seed(0)`)
- Clean the original data by removing counties with NaN (as before, use `df.dropna()`)
- Calculate the **true** mean of the *median income* (print it)
- Take a random sample of $n = 1000$ samples, calculate the sample mean, and calculate the difference from the true mean (print them)
- Repeat the process of sampling $n = 1000$ (with replacement) $N = 10, 20, 40, 50, 100, 200, 500, 1000, 2000$ times, and for each N calculate the mean, calculate the difference from the true mean, plot the difference from the true mean vs. N.

```
In [ ]: # your code here - you can use as many cells as you need
```

Question 2 - Point Estimates

- What is the process of sampling that you just performed? Explain briefly how it works and what is it good for.
- What will happen as $n \to \infty$? Refer to the Central Limit Theorem (CLT) in you answer.

Task 3 - Confidence Intervals

A point estimate can give you a rough idea of a population parameter like the mean, but estimates are prone to error and taking multiple samples to get improved estimates may not be feasible (would you call 500 people, 2000 times?). A confidence interval is a range of values above and below a point estimate that captures the true population parameter at some predetermined confidence level. For example, if you want to have a 95% chance of capturing the true population parameter with a point estimate and a corresponding confidence interval, you'd set your confidence level to 95%. Higher confidence levels result in a wider confidence intervals.

Calculating a confidence interval - taking a point estimate and then adding and subtracting a margin of error to create a range. Margin of error is based on your desired confidence level, the spread of the data and the size of your sample. The way you calculate the margin of error depends on whether you know the standard deviation of the population or not.

- Calculate a 90%, 95% and a 99% confidence interval for the **mean** *median income* of voters. Take a sample of $n = 1000$.
  - Equivalently, $\alpha = [0.1, 0.05, 0.01]$
  - Don't forget to **standartize** the data. and notice that $\sigma \neq 1$ as in the tutorial.
  - Use the true STD of the population
  - To calculate the inverse of the CDF, use `scipy.stats.norm.ppf()`
- Compare your answer to using `scipy.stats.norm.interval(alpha=, loc=, scale=)` . Read the doc, use Shift + Tab (only in Jupyter) inside the parenthesis to understand how to use it.
  - You need to print the result for each α, with both methods

In [ ]:
```
# your code here - you can use as many cells as you need
```

Question 3 - Confidence Intervals

- Explain the results and their meaning.
  - What is the trend (the higher the confidence...)?
- Why point estimation is not enough?

Task 4 - Confidence Intervals 2

- Take $N = 25$ samples of $n = 1000$, and plot the confidence interval for each sample (x axis - sample #, y axis - confidence interval for a confidence level of 95% ($\alpha = 0.05$)
  - Use `ax.errorbar(x=list(range(N)), y=sample_mean, yerr=[(upper-lower)/2 for upper,lower in intervals], fmt='o')`
  - Add the **true** mean using `ax.hlines((xmin=0, xmax=N, y=true_mean, linewidth=2.0, color="red")`

In [ ]:
```
# your code here - you can use as many cells as you need
```

Question 4 - Confidence Intervals 2

- As you have probably noticed, there intervals that **don't include** the true mean. How is that possible?

Task 5 - Hypothesis Testing - Two-Sided One Sample T-Test

Point estimates and confidence intervals are basic inference tools that form the foundation for another inference technique: statistical hypothesis testing. Statistical hypothesis testing is a framework for determining whether observed data deviates from what is expected. The T-test is a statistical test used to determine whether a numeric data sample differs significantly from the population or whether two samples differ from one another.

In this exercise we will perform the t-test for the mean of unemployment fraction (`clf_unemploy_pct`) in the US. We will use the pre-procesesd data (`prep_df`) and a sample of size $n = 10$. We wish to test whether the sample mean differs from the population mean. The null hypothesis, H_0 is that the mean is not different. Recall that we are performng t-tests when we are not given the true mean and true STD. The steps:

- Constant the seed
- Calculate the true mean of unemployment fraction (use fractions and not percents: `clf_unemploy_pct / 100`)
  - In t-test, you assume that the mean (μ) is unknown, then you formulate the $H_0 : \mu = \mu_0$. So if μ is unknown, how do you calculate the t-statistic? Well, assume that somebody told you that in the last elections, $\mu = C$, where C is the also the true mean of this election.
  - In other words, to calculate t-statistic, use the true mean.
- Take a sample of $n = 10$ and calculate its mean.
- Calculate the sample's unbiased STD.
- Calculate the t-statistic value.
  - Use the true mean for $\overline{\mu} - \mu_0$
- Calculate $t_{n-1,\frac{\alpha}{2}}$ for $\alpha = [0.1, 0.05, 0.01]$ significance levels (how many derees of freedom are there for the t-distribution?). Use `scipy.stats.t.ppf()` .
- Calculate the p-value (don't forget that this is a **two-sided** test). For each level, do we reject H_0 or accept it?. Use `scipy.stats.t.cdf()` .
- Compare your results to `scipy.stats.ttest_1samp()` . This function performs the t-test, and outputs the t-statistic and the p-value.

Note: "Calculate" = print the results

- Tips:
  - When calculating the p-value, pay attention to which side are you on (if t-statitic > 0, then you should calculate the p-value for $1 - \phi(x)$)
  - Your results should align with the output of `scipy.stats.ttest_1samp()` (make sure the seed is constant)

In [ ]:
```
# your code here - you can use as many cells as you need
```

Question 5 - Hypothesis Testing - Two-Sided One Sample T-Test

- For a sample of size n, what is the number of degrees of freedom for the student-t distribution?
- What is the relationship between the significance level and the confidence level? Explain.
- What is the meaning of the p-value? What is the meaning of the p-value being lower than some siginificance level?

Part 2 - Dimensionality Reduction - Prologue - Wine Dataset

In this exercise we are going to compare different dimensionality reduction techniques on the Wine dataset. The wine dataset is a classic and very easy multi-class classification dataset. There are 3 types of wine, 178 examples with 13 features each. Even though it is a very dataset for classification, we will do unsupervised analysis to compare different aspects of dimensionality reduction methods. Let's look at the data.

In [ ]:
```
# imports for the exrcise - part 2
# you can add more if you wish (but it is not really needed)
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA, KernelPCA
from sklearn.manifold import TSNE, LocallyLinearEmbedding, Isomap
from sklearn.preprocessing import StandardScaler
from sklearn.datasets import load_wine, load_digits
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, Perceptron
```

In [ ]:
```
from sklearn.datasets import load_wine
X, y = load_wine(return_X_y=True)
pd.DataFrame(np.concatenate((X, y.reshape(-1, 1)), axis=1), columns=list(range(13)) + ['class'])
```

Task 6 - Importance of Feature Scaling

- Perform PCA on the data (`X`) to `n_components=2` and plot it.
- Scale the data using `StandardSaler()` to create `X_scaled` , perform PCA on `X_scaled` to `n_components=2` and plot it.
- Color the datapoints according to their class: `ax.scatter(X[:,0], X[:,1], c=y)`

Note: for all algorithms, use constant seed (`random_state=...`), for example: `PCA(n_component=2, random_state=random_state)`

```
In [ ]: # your code here - you can use as many cells as you need
```

Question 6 - Importance of Feature Scaling

- Why is it so important to perform feature scaling before performing dimensionality reduction, espicially for PCA?
- Describe the results of Task 6

Task 7 - T-SNE

- Perform t-SNE on the scaled dataset to `n_components=2` and plot it.
  - Find the `perplexity` , the yields the best-looking results.

```
In [ ]: # your code here - you can use as many cells as you need
```

Question 7 - T-SNE

- Explain in short how t-SNE works and how it is different from PCA.
- Compare the results of PCA and t-SNE on the Wine dataset

Task 8 - A Short Visit to Supervised Learning

In order to evaluate the quality of our dimensionality reduction, we can, given labels, train a classifier. Though this is the focus of ML course, we will briefly train a linear Perceptron on the lower-dimension dataset (multi-class Perceptron is trained using "One-vs-All" scheme, but it is not important).

- Split the lower-dimension scaled features to train and test test using `train_test_split` . Use 20% of the data for test.
  - Usage: `train_test_split(X_scaled_pca/tsne, y, test_size=0.2, random_state=42)`
- Train a Perceptron on the scaled PCA dataset, evaluate the accuracy on the test set (print the results).
- Train a Perceptron on the scaled t-SNE dataset, evaluate the accuracy on the test set (print the results).

```
In [ ]: # your code here - you can use as many cells as you need
```

Question 8 - A Short Visit to Supervised Learning

- Describe and explain what is Perceptron doing.
- Describe the accuracy results. If there is a difference, explain its source.

Task 9 - Robustness to Noise & Outliers

In this task we are going to test how robust are t-SNE and PCA to noisy features and outliers. We will have 2 new datasets:

1. `X_noisy` - random normal noise ($\mathcal{N}(0,1)$) is added to the current features.
2. `X_skewed` - 20 random samples are added to the sample (total samples: 178 + 20 = 198)

The tasks:

- Perform standardization to create `X_noisy_scaled` , `X_skewed_scaled`
- Peform PCA and t-SNE to both datasets (4 in total) to `n_components=2` (as before) and plot the results (4 plots, can be in pairs and can be a 2X2 plot, don't forget to put titles).
  - You should tune the `perplexity` for t-SNE
- For each of: `X_noisy_scaled_pca` , `X_noisy_scaled_tsne` , `X_skewed_scaled_pca` , `X_skewed_scaled_tsne` train a `LogisticRegression` classifier and measure the accuracy (4 classifiers in total). Don't forget to split to train and test, as before.

In [ ]:
```
# your code here - you can use as many cells as you need
```

Question 9 - Robustness to Noise & Outliers

- Explain the results. In your answer, describe the effect of adding noise and outliers, which algorithm is more affected and why?

Task 10 - Comparing Dimensionality Reduction Methods

In this task we will compare different dimensionality reduction techniques on the Digits dataset (Each datapoint is a 8x8 image of a digit). Reduce dimensions to `n_components=2` . Plot all the results. You should tune each algorithm's hyper-parameters. Use the table below:

Algorithm	Call to...	Hyper-parameters to tune
PCA	`PCA()`	None
KernelPCA	`KernelPCA()`	`kernel`
t-SNE	`TSNE()`	`perplexity`
Locally Linear Embedding (LLE)	`LocallyLinearEmbedding()`	`n_neighbors`
Isomap	Isomap()	`n_neighbors`

- Note, to add a color bar to your plot, use: `scatter = ax.scatter(X_tsne[:,0], X_tsne[:,1], c=y, cmap=plt.cm.rainbow)` `plt.colorbar(scatter)`
- Don't forget to add titles

In [ ]:
```
# your code here - you can use as many cells as you need
```

Question 10 - Comparing Dimensionality Reduction Methods

- Shortly describe what each method from the above table does, and the effect of the hyper-parmeters.
- Which method produced the most satisfying results?

Credits

- Icons from Icon8.com (https://icons8.com/) - https://icons8.com (https://icons8.com)
- Datasets from Kaggle (https://www.kaggle.com/) - https://www.kaggle.com/ (https://www.kaggle.com/)