A training module for data-driven Public Policy in Indonesia

Shaan Perlambang (71735892)

shaanbarca@keio.jp

Keio University

Faculty of Policy Management

Supervisor(s):

Mika Kunieda

mkunieda@sfc.keio.ac.jp

Introduction

The government of Indonesia has launched an initiative called Satu Data Indonesia (SDI) to consolidate data, improve data quality, and enable access to central and local government data. The objectives of the program in its own words are to increase transparency and accountability of the government and help the development of the nation. By leveraging machine learning, we can help realize the goals of the program to help policymakers make data-driven decisions. Badan Pusat Statistik (BPS) Indonesia is a government bureau that collects, surveys, and analyzes data for the government. They are the main entities that provide statistical analysis to help decision-makers to make data-driven policies.

Current situation

The primary focus of the BPS is on sampling or data acquisition and not data analysis[2]. While some data analysis is done, most government ministries, agencies and other forms of organizations need to conduct the analysis independently. As the analysis is being conducted individually, there is yet to be a standardized way to clean, visualize, understand and evaluate the results of their analysis.

What does this notebook hopes to address

Previous studies on (ML) for public policy in Indonesia have largely focused on prediction/forecasting [4]. Certain implemented models were also “blackbox” methods, where the results were not interpretable. For this application, policymakers must not only have predictions of future forecasts but also reasoning on how the predictions were derived. While prediction is certainly an important component, it is also important to identify the levers that influence such outcomes. For example, if a model is developed to accurately predict the hospitalization rate in a particular area, we must also know what influences such factors. Thus, in this domain, we must not only employ models, but also ensure that they are interpretable.
Research has been published on the use of interpretable models in Indonesia. They employed Causal Forest, an ML-based model for Indonesian public healthcare that identifies causal variables [7]. While this was able to successfully derive causal factors, the emphasis on the papers was largely on singular modeling rather than a diverse approach. Situations related to public policy are often multifaceted problems that require a diverse approach to derive an understanding from different perspectives. Recognizing the use of multiple approaches is something that also needed to be further explored.

Finally, there is a lack of emphasis on understanding visualized data in the current literature. Descriptive statistics are typically included however with current frameworks used in Data Science, “Explanatory Data Analysis” goes beyond just looking at the mean, mode, median and deviation etc. in tabular fashion. Different types of visualizations are needed to understand the variables involved and to fully grasp the situation. It also helps us to decide which models are the most ideal for the dataset.

Notebook goal

This notebooks aims to serve as a framework for Data science applications for public policy in Indonesia. Specifically, it addresses the main gaps in the current literature by employing data explanatory techniques (1), using an ensemble of models (2) methods to make them interpretable (3), and translating its results to actionable policy recommendations (4). Additionally, it also discusses the end-user's involvement and the limitations of ML to avoid negligent use (5). Finally, we will briefly discuss on deriving causality from our results (6).

To fully illustrate this, we analyze the **Penghasilan Asli Daerah / Local Government Revenue or referred to as PAD per Capita** in our dataset of each province in Indonesia. The PAD is income of a city minus subsidy/funds they receive from the central government. The goal of this notebook would then be to see which factors influence a given city's income

An extensive list of variables are included, such as geographical location, level of education, and employment to name a few. These variables were represented as continuous, categorical, ordinal, and spatial values.
Dealing with such a diverse dataset allows demonstrations of how machine learning can be used in public policy. Because the Satu Data Indonesia initiative is still ongoing, the dataset used may not be fully accurate. Thus, the model interpretations and policy recommendations made in this study serve only as an example and to illustrate the framework.

In this notebook we are going to explore datasets from the ministry of finance and population and civil registration agency.

NOTE: Throughout the notebook we will be referring to independent variables and features interchangeably. In Data science or Machine Learning lingo, this means the same thing.

Prerequisites

There are several prerequisites that need to be fulfilled before following this notebook.

1. A basic understanding of python

Understanding variables, functions and using libraries should be enough

2. Experience using commonly used data science libraries
3. Matplotlib, pandas and NumPy and scikit-learn
4. A basic understanding of Machine Learning (Statistics, Probability, Linear Algebra)
5. Experience using Jupyter Notebooks

Here are some resources that you can refer to :

Python :

https://www.learnpython.org/

Pandas : https://pandas.pydata.org/docs/user_guide/10min.html

Numpy : https://numpy.org/doc/stable/user/quickstart.html

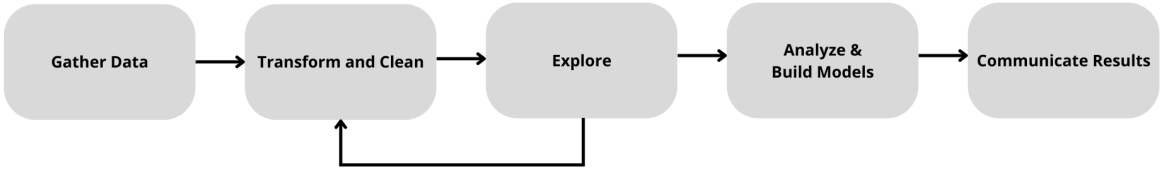
Scikit-learn and machine learning:

https://scikit-learn.org/stable/tutorial/basic/tutorial.html

How to use this guide:

It is highly recommended to use your own dataset and use this guide as a reference on how to clean, explore, build models and interpret your findings. Sections with advanced in title can be skipped however it is recommended to go through the visualizations to see what insights can be extracted from these steps. This guide will not give an in-depth explanation for each section, that will be outsourced to other resources. Instead, this demonstrates an end to end implementation of DS for Public Policy.

Framework



1. Gather data. Data can come from many and different sources, in this notebook we will deal with data from different sources and how to merge them
2. Transform and clean. Often times, the data that we work with is dirty. This means there are typos. If we do not check the quality of our data, this will lead to spurious regression and false conclusions.
3. Explore. Exploring our data allows us to see which variables might be interesting, serves as a roadmap on what models to build and also allows to see which data might be false. If we see data that does not make sense, we have to clean them. We will consistently clean and explore data throughout the notebook as we see fit.
4. Build models and Analyze. Ultimately, in this notebook our goal is to use Machine Learning models. There are plenty of examples of using linear regression and we will use some linear regression in this notebook however we would also like to explore how to use state of the art models! Note: A lot of utility can be derived from linear regression. When possible always use simpler models.
5. Communicate our results. Ultimately, the goal of our notebook is to interpret our finding and articulate them to policy recommendations. Often we instances where practitioners stop after they have built models and focusing too much on how to improve performance. While an accurate model is no doubt important, for policy makers they want to know what actions to take. This is why we should spend our time on interpreting our results.

Libraries

These are the python libraries that will be used to allow us to conduct the analysis

Installing libraries can be tricky. Navigate to the **Appendix** below and read the **Reproducible environments** section.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mlp
import seaborn as sns
from matplotlib import rcParams
import geopandas as gpd
#----------------------- Data analysis libs

from sklearn.feature_selection import mutual_info_regression
from sklearn.model_selection import GridSearchCV
import xgboost as xgb
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder, StandardScaler, OrdinalEncoder
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import plot_confusion_matrix


#-------------- We want to set the figure size here
sns.set(rc={'figure.figsize':(20,10)})
plt.style.use(
    "https://github.com/aeturrell/coding-for-economists/raw/main/plot_style.txt"
)
rcParams['figure.figsize'] = 20,10

#----------------- Pandas settings
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

Load dataset

There are several problems that will occur frequently when dealing with datasets.

- Data inputed incorrectly
- Inconsistent naming, this is especially difficult when dealing with multiple datasets
- Modifying data to ensure they are in the correct format

Throughout the notebook, we will be dealing with all these problems and clean them along the way

```
df = pd.read_excel('normalized_gp2_data.xlsx')
df.head()
```

	Unnamed: 0	Provinsi	Kabupaten/Kota	Jumlah Kecamatan	Jumlah Desa	Jumlah Kelurahan	Jumlah Penduduk	Jumlah KK	Luas Wilayah (km2)	Kepadatan Penduduk	Perpindahan Penduduk	Jumlah Meninggal	Perubahan Data	Wajib KTP	Kepercayaan terhadap Tuhan YME	Laki-Laki	Per
0	0	ACEH	PIDIE	23	730	0	435797	132522	3133	139	220656	828	415357	310044	0	215141	
1	1	ACEH	SIMEULUE	10	138	0	94368	25611	1821	52	45990	282	85629	66453	1	48378	
2	2	ACEH	KOTA BANDA ACEH	9	90	0	253198	74245	55	4534	126525	939	231423	171676	0	126673	
3	3	ACEH	KOTA SABANG	3	18	0	42696	12850	122	350	21161	175	40653	29413	0	21535	
4	4	ACEH	KOTA LANGSA	5	66	0	185836	53727	217	856	92425	441	165304	134356	4	93411	

```
df.shape
```

```
(507, 91)
```

91 columns ! Thats quite a lot of features. We should explore how the relationship of our different features with each other and see which columns are most important.

Functions

These are the 2 functions that we will use. As you can imagine calculating summary statistics or outliers is something that will be done multiple times

```
def descriptive_table(data):
  """Outputs descriptive statistics of our DataFrame


   Args:
         data (DataFrame): Pandas DataFrame to calculate summary statistics

   Returns:
         sum_table(DataFrame): Pandas DataFrame summary statistics transposed

    """
    sum_table = data.describe().round(1)
    sum_table = sum_table.T
    return sum_table

def outlier_data(data,col):
    """outlier value of particular data frame column


    Args:
        data (DataFrame): Dataframe to calculate outlier from
        col (str): column to calculate outlier from

    Returns:
        outlier (float): outlier value of a particular column

    """
    percentiles = descriptive_table(data[[*col]])
    q3 = percentiles['75%']
    q1 = percentiles['25%']
    iqr = q3-q1
    outlier = q3 + (1.5*iqr)
    return float(outlier)
```

To reuse these functions multiple times, we can create a utility library with these functions (or more) within it instead of defining the function for every notebook/script. Store your notebook/script and utilities in the same path

```
📁 project
|——main.ipynb or main.py
|——utilities.py
```

Import them to your notebook/script

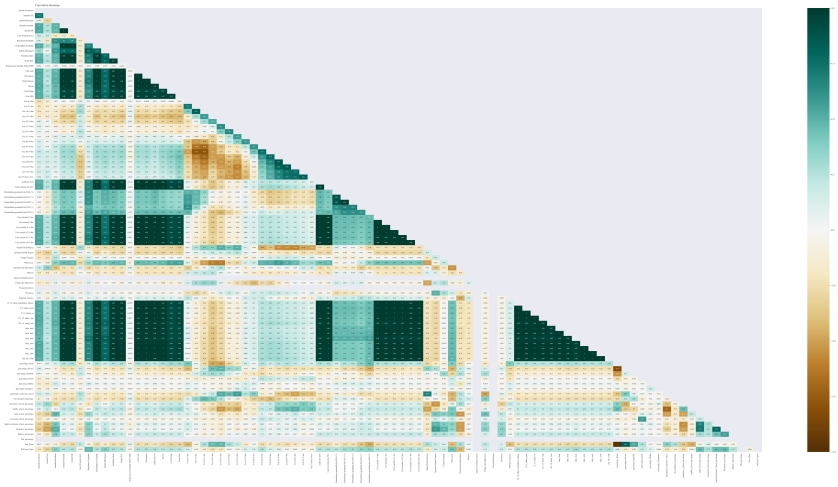
```
from utilities import descriptive_table, outlier_data
```

Initial Exploratory data analysis

Heat map

By using a heat map, we can see how each variable correlates with each other. We can also see which variables are correlated to our target variable (PAD per Capita).

```
plt.figure(figsize=(100, 50))
mask = np.triu(np.ones_like(df.iloc[:,3:].corr(), dtype=bool))


heatmap =sns.heatmap(df.iloc[:,3:].corr() ,mask=mask, vmin=-1,vmax=1,annot=True,cmap='BrBG')
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':18}, pad=12);
plt.savefig('heatmap.png', dpi=300, bbox_inches='tight')
```



From this heat map we can see variables such as Agama and Kepercayaan or Tenaga kesehatan has no correlation to the target variable. This could also be that these independent variables has not been inputed appropriately.

Multicollinearity

Depending on the model we will be using, we should remove variables that are highly correlated. The main reason multicollinearity is a problem is it makes interpreting our models much more difficult. It makes it difficult to isolate how a variable impacts our model and also makes our model unstable. We will explore this in detail in the pre-processing section

Remove features that have no correlation

In this instance, all of the columns that have no correlation (grey) in relation to the target variable will be dropped. This of course depends on our knowledge of the dataset too. There are certain variables that may not be useful on their own but important once we combine it with other variables. In this case, it might not be wise to completely drop them. In the appendix, there will be other alternatives on how to deal reducing dimensionality of our data

```
df = df.drop(columns=['Agama dan Kepercayaan','Tenaga Kesehatan','Kepercayaan terhadap Tuhan YME','Unnamed: 0','f4_18_tahun_pendidikan
                      'f5_6_tahun_paud','f7_12_tahun_sd','f12_15_tahun_smp','f16_18_tahun_sma','lahir_thn4','lahir_thn5','lahir_thn6',
                      'lahir_seb5','lahir_seb6','Usia sekolah 3-4 thn','Usia sekolah 5 thn','Usia sekolah 6-11 thn','Usia sekolah 12-2
```

Descriptive statistics

Descriptive statistics useful to see if our data is reliable. In this particular table we have the count, mean, std, min max and percentiles.From here we can see if the ranges of our data make sense. We can easily cross check our data with other data sources to verify its validity. For example, we can easily check that Jakarta is the most populated city with a population of around 11 million. If any city/regency exceeds this value, then it is most likely erroneous data. There are other ways to check

for outliers but this is the most basic way. For instance, we know that the population in Indonesia is around 275 million, if the average or max population of our Jumlah Penduduk(Number of population) exceeds this amount or is close to this amount, we know that there is something wrong with our data.

```
summary_stats_all = descriptive_table(df)
summary_stats_all.head(5)
```

	count	mean	std	min	25%	50%	75%	max
Jumlah Kecamatan	507.0	14.3	8.2	2.0	8.0	12.0	19.0	51.0
Jumlah Desa	507.0	147.6	122.1	0.0	61.5	127.0	214.5	852.0
Jumlah Kelurahan	507.0	16.7	25.6	0.0	2.0	9.0	20.5	267.0
Jumlah Penduduk	507.0	534415.5	771077.1	24855.0	155021.5	285420.0	618542.5	11261595.0
Jumlah KK	507.0	171284.6	253514.2	5465.0	47446.0	88427.0	186818.0	3655673.0

Preprocessing data

Feature Engineering ：Feature scaling

Since we are interested in the Income per Capita of a City/Regency, we must also get the percentages of features instead of raw value. For example, if we had 2 cities, City A and City B and they both have 10,000 people who have attained a bachelors degree if City A has a population of 100,000 while City B has a population of 1,000,000 we know that in City A, 10% of the population are well educated while in City B only 1% are well educated. By not scaling them to percentages, we are not giving a clear picture of the whole situation.

As we are calculating income per capita, the same logic follows when scaling our data:
with percentage of those with a bachelors education for example it would be :

Percentage of those with a bachelors education = number of people with a bachelors degree/ population of city

```
# calculate population density ourselves
df['population_density'] = df['Jumlah Penduduk'] / df['Luas Wilayah (km2)'] # results were different then the one calculated by the go
df['Percentage_married'] = df['Kawin'] / df['Jumlah Penduduk']
df['Percentage_divorced_alive'] = df['Cerai Hidup'] / df['Jumlah Penduduk']
df['Percentage_divorced_death'] = df['Cerai Mati'] / df['Jumlah Penduduk']

# remove redundant features

df = df.drop(columns=['Cerai Hidup','Cerai Mati','Kawin','Belum Kawin','Kepadatan Penduduk'],axis=1)
```

Removing features

While additional features/independent variables can help us making better predictions, often multi- collinear features or data that is not relevant to our model may harm it. The reason for this is our model will start overfitting. Overfitting occurs when our

model starts memorizing our dataset instead of identifying patterns. Our model performs well in the 'training set' the sample its uses to find the optimal parameters but struggles to generalize in the 'test set'.

There are multiple ways to approach this. Depending on our knowledge of the data we can simply remove features we know are not important. If not there are other methods to remove or even combine features that will be explored on later.

```
df = df.drop(columns=['Cerai Hidup','Cerai Mati','Kawin','Belum Kawin','Kepadatan Penduduk'],axis=1)
```

Exploratory data analysis

Exploratory data analysis (EDA) summarizes the datasets through summary statistics and visual representations. This allows us to recognize patterns, identify outliers, and understand the types of data that we are dealing with. It helps us build the intuition of our dataset by selecting how we should clean our data, what potential problems we may face when modeling our data, what types of analysis we can conduct, what models we should select, and help those with less technical knowledge better understand our data.

Our dataset can be visualized in multiple ways. These are divided into three types of visualization analysis.

1. Univariate analysis was performed to visualize the distribution of the data. This is done when analyzing a single variable.
2. Bivariate analysis is done to visualize the relationship of two variables.
3. Multivariate analysis is done when there are two or more.

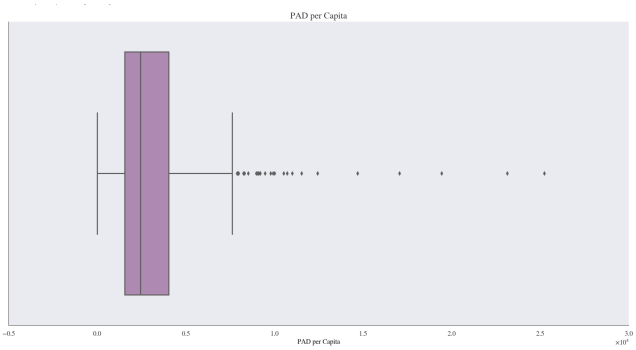
Common visualizations are histograms, box plots and scatterplots.

Box plots

Looking at data from tables to see and check which are anomalous or potentially erroneous would take a long time. A solution to that is by utilizing box plots to detect outliers. We can identify outliers by using the function we defined above. We need to ensure the outliers are actually valid.

Outliers should not be removed immediately as they can give us useful insights on why they occur. Later on in the notebook, we will try to understand why they occur

```
plt.figure(figsize=(20, 10))
ax = sns.boxplot(x=df['PAD per Capita'])
ax.set_title("PAD per Capita", loc="center")
```



We can see quite a few outliers when we visualize our data using box plots, lets see what those outliers are

```
pad_outlier = outlier_data(df,'PAD per Capita')

df[df['PAD per Capita']> pad_outlier][['Kabupaten/Kota','PAD per Capita']
].sort_values(
by='PAD per Capita',ascending=False)
```

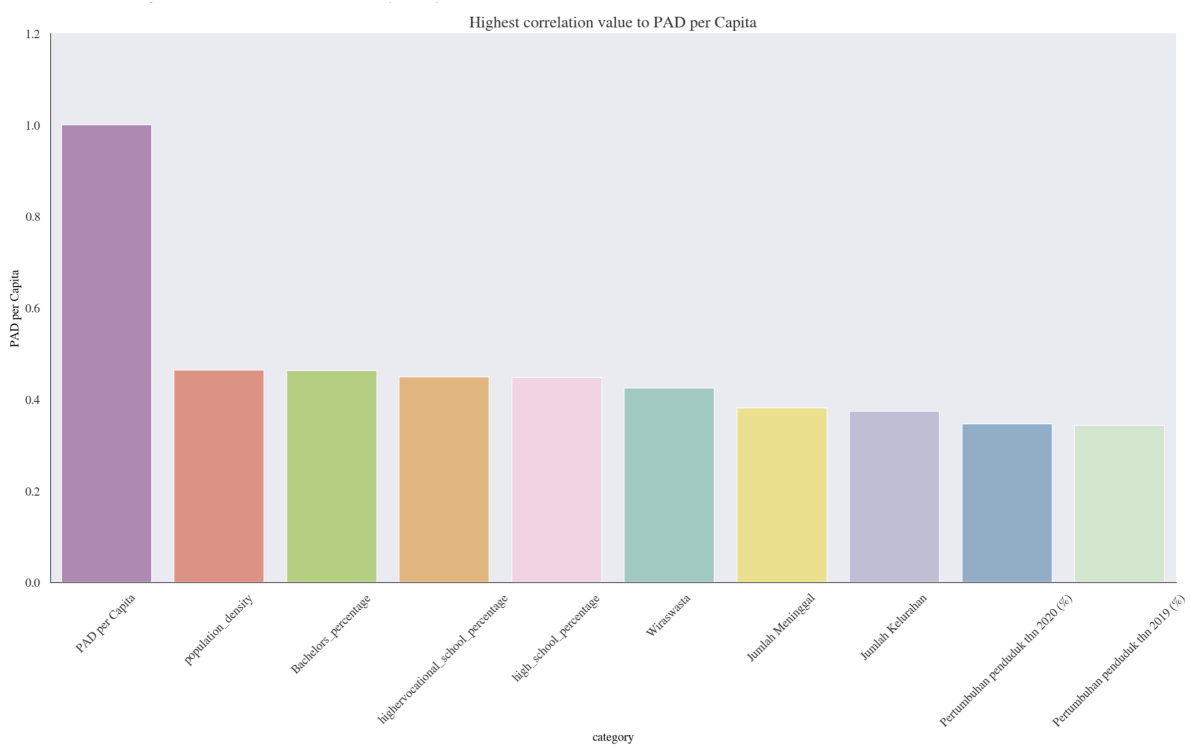
	Provinsi	Kabupaten/Kota	PAD per Capita
506	DKI JAKARTA	DKI JAKARTA	25239.854568
363	BALI	BADUNG	23138.824627
438	PAPUA	MIMIKA	19448.098101
189	JAWA TENGAH	KOTA MAGELANG	17067.465089
484	PAPUA BARAT	MANOKWARI SELATAN	14696.091065
228	JAWA TIMUR	KOTA MOJOKERTO	12438.083687
318	SULAWESI TENGAH	MOROWALI	11532.466909
231	JAWA TIMUR	KOTA SURABAYA	11014.621576
263	KALIMANTAN TENGAH	KOTAWARINGIN BARAT	10729.515865
237	JAWA TIMUR	GRESIK	10525.994465
471	GORONTALO	KOTA GORONTALO	10022.287437
477	KEPULAUAN RIAU	KARIMUN	9930.473486
208	DAERAH ISTIMEWA YOGYAKARTA	KOTA YOGYAKARTA	9788.514810
458	BANTEN	KOTA CILEGON	9471.252188
3	ACEH	KOTA SABANG	9202.360877
1	ACEH	SIMEULUE	9186.715836
192	JAWA TENGAH	KOTA SEMARANG	9113.944697
283	KALIMANTAN TIMUR	KOTA BONTANG	9056.005351
226	JAWA TIMUR	KOTA MADIUN	8975.438318
461	BANTEN	KOTA TANGERANG SELATAN	8522.197886
369	BALI	KLUNGKUNG	8311.435388
371	BALI	KOTA DENPASAR	8254.660440
127	LAMPUNG	KOTA METRO	7991.586490
152	JAWA BARAT	KOTA CIREBON	7936.205687
481	KEPULAUAN RIAU	BINTAN	7893.857331

We can then check one by one if there are any cities with data thats are falsely inputed. While this process can still be tedious, this narrows the search for cities that might have data that was falsely inputed.

Correlation

Now we want to see which of our independent variables are linearly correlated with our independent variable. When we use our heat map before, we saw how they were correlated with each other, similar to a correlation matrix. Now, we can zoom in and see how they are correlated with PAD per Capita

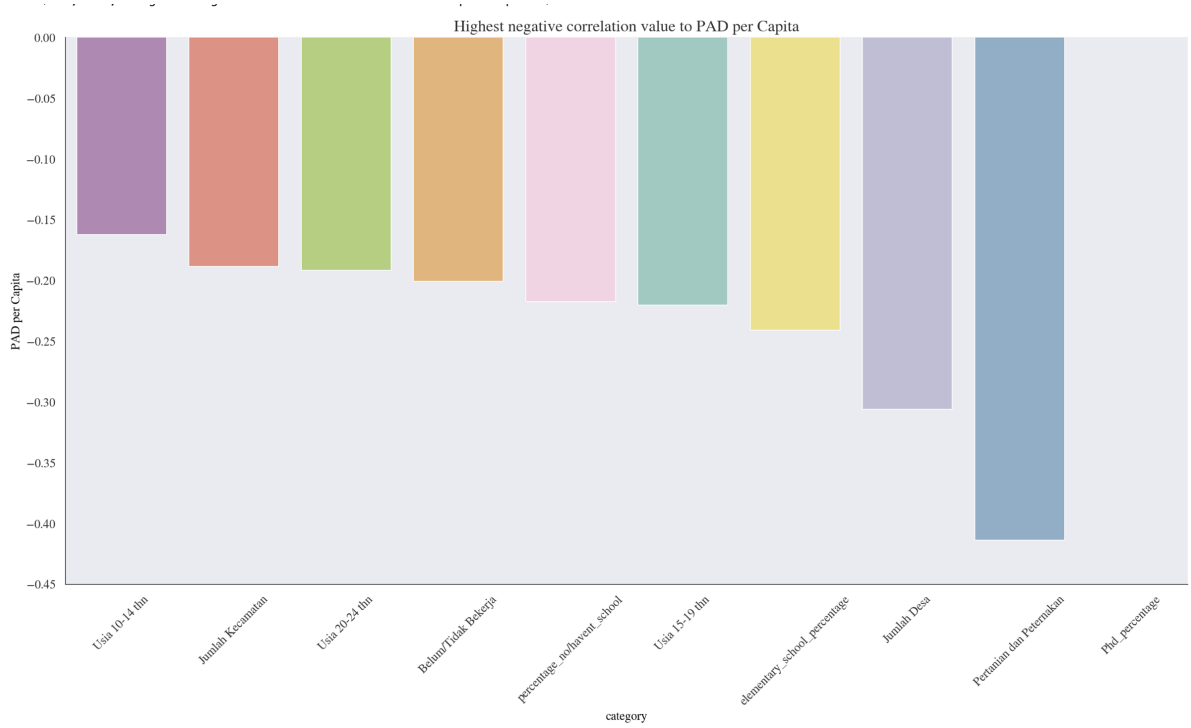
```
highest_pos_corr_pad = df.corr()[['PAD per Capita']].sort_values(by='PAD per Capita',ascending=False).head(10)
highest_pos_corr_pad['category'] = highest_pos_corr_pad.index
highest_pos_corr_pad.head()
ax = sns.barplot(x="category", y="PAD per Capita", data=highest_pos_corr_pad)
ax.set_xticklabels(labels = highest_pos_corr_pad.category,rotation=45)
ax.set_title("Highest correlation value to PAD per Capita", loc="center")
```



We see that population density and variables pertaining level of education are positively correlated to a cities PAD per Capita

```
lowest_pos_corr_pad = df.corr()[['PAD per Capita']].sort_values(by='PAD per Capita',ascending=False).tail(10)
lowest_pos_corr_pad['category'] = lowest_pos_corr_pad.index
#lowest_pos_corr_pad.head()
ax = sns.barplot(x="category", y="PAD per Capita", data=lowest_pos_corr_pad)
ax.set_xticklabels(labels = lowest_pos_corr_pad.category,rotation=45)
ax.set_title("Highest negative correlation value to PAD per Capita", loc="center")
```

We can also visualize which variables are negatively correlated to our independent variable

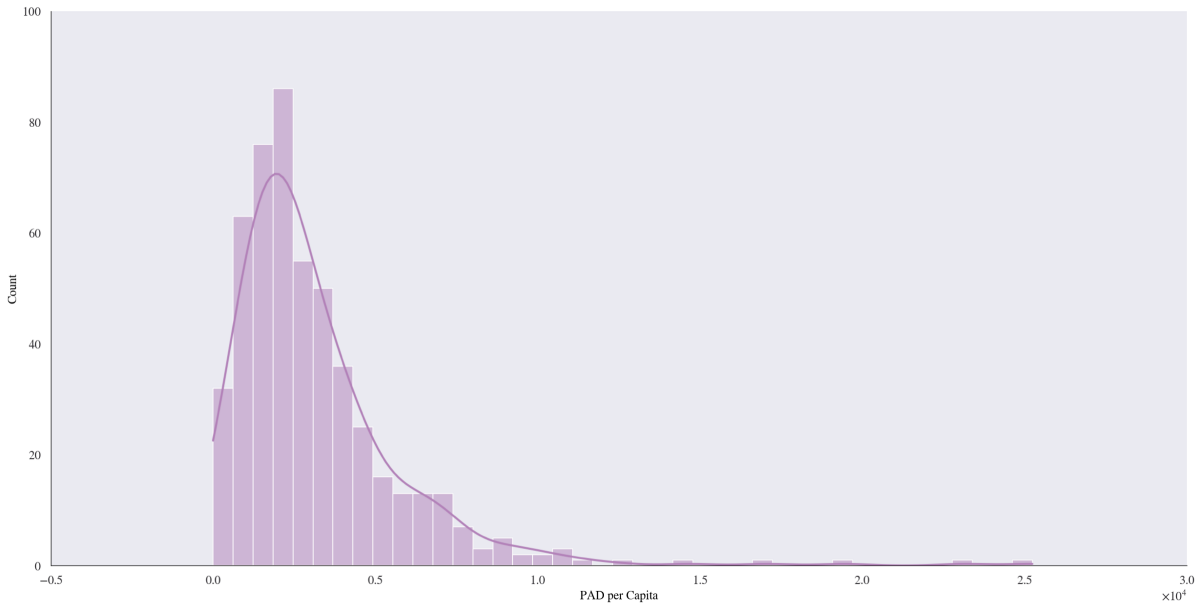


Now that we have identified which variables have the highest correlation value, let visualize their distributions

Distributions

We will first start by plotting the distribution of the PAD per Capita

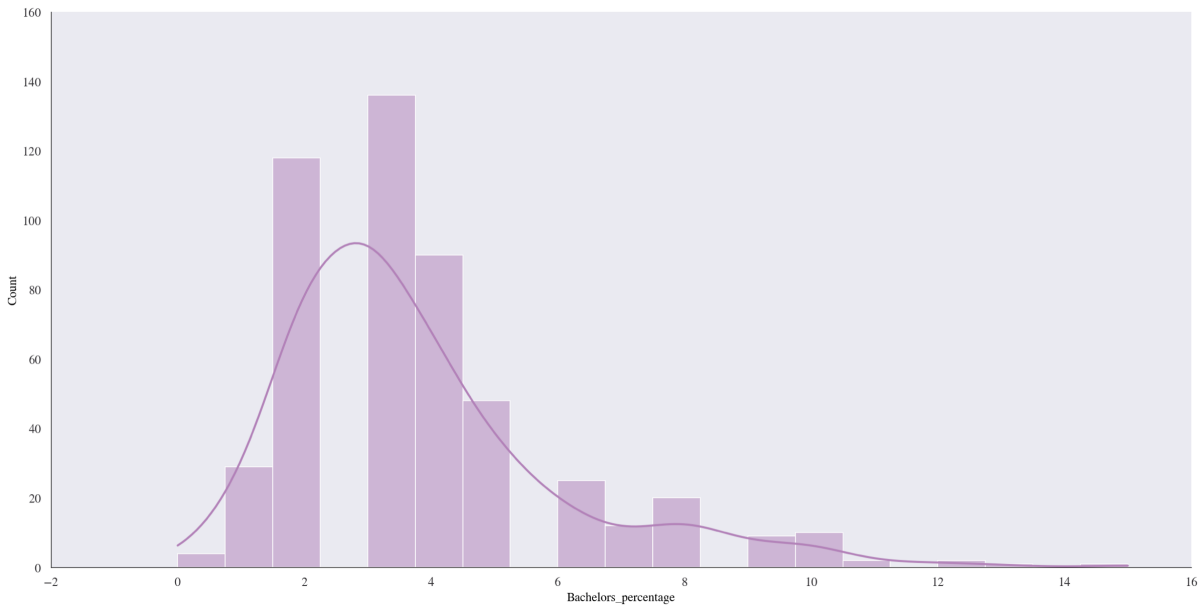
```
ax = sns.histplot(data=df,x="PAD per Capita",kde=True)
```



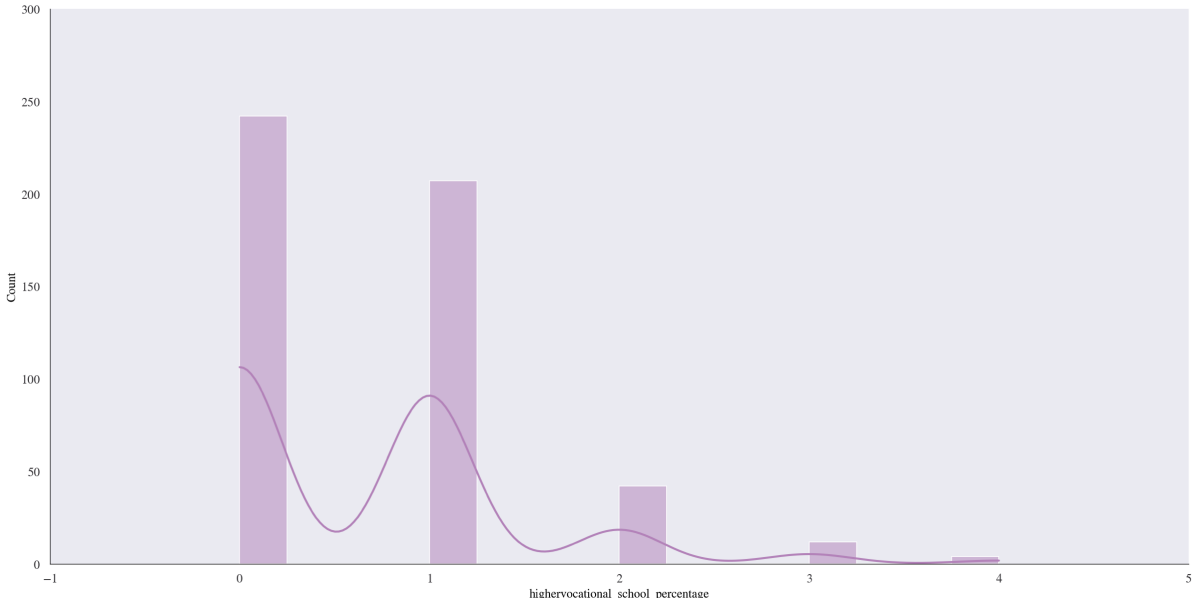
By plotting the distribution, we see that we have a positively skewed distribution. This means the median of our data is less than the mean. What this is telling us is that a few cities/regencies in Indonesia have an Income per Capita that is much higher than the rest that influences our mean to be greater than the median. Or another way of saying it is The income distribution in Indonesia is imbalanced.

Let us then plot the distributions of other variables that show a strong correlation to the PAD per Capita

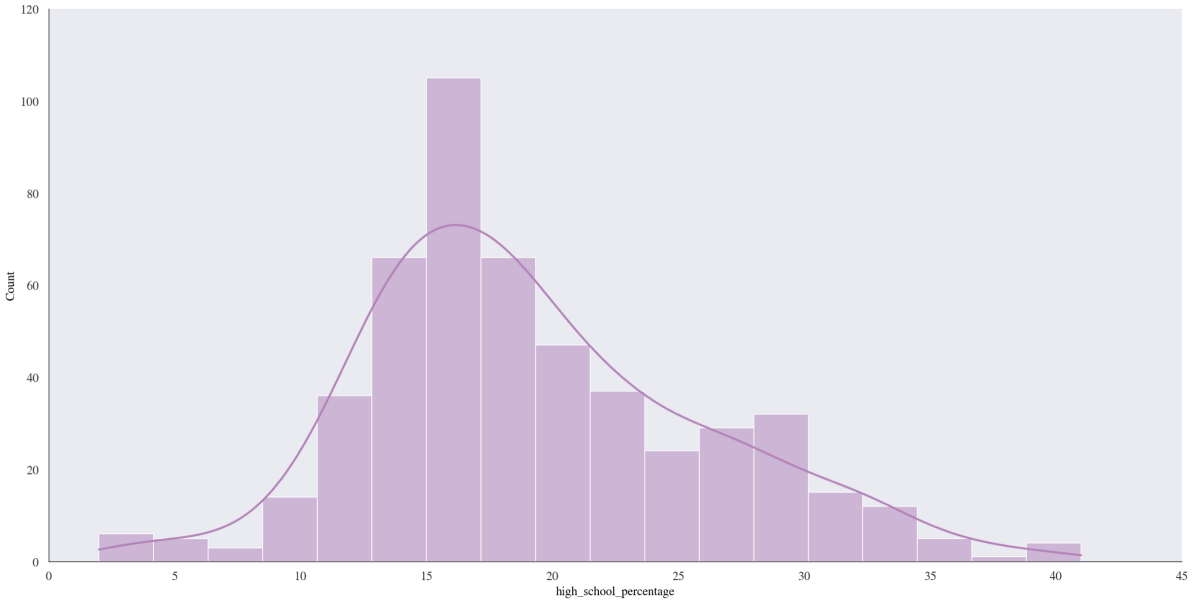
```
ax = sns.histplot(data=df,x="Bachelors_percentage",kde=True)
```



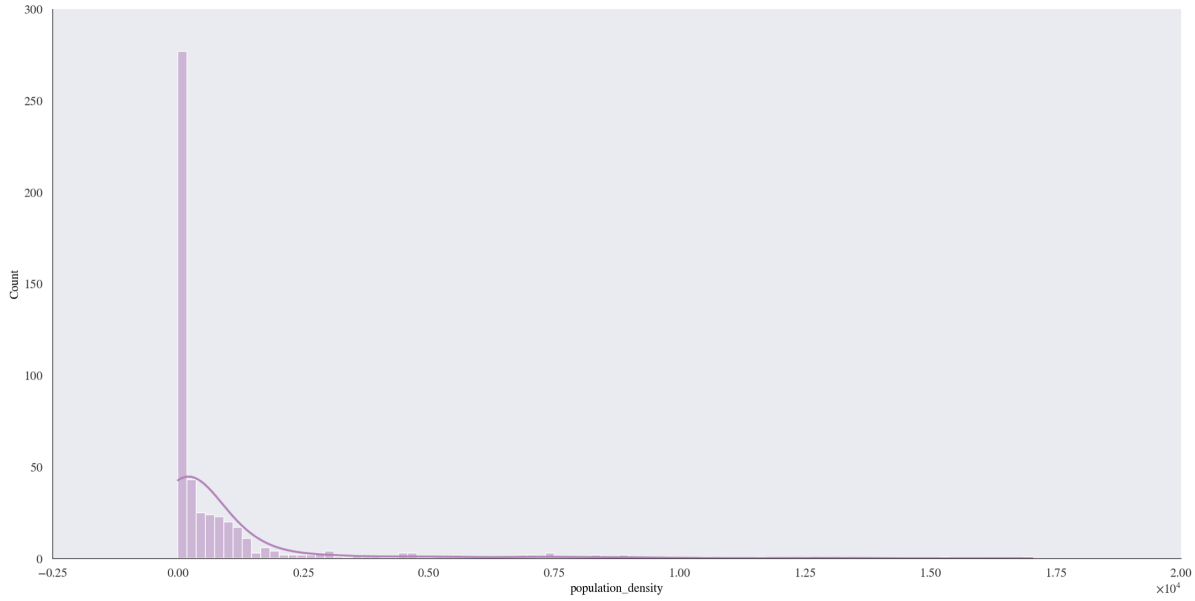
```
ax = sns.histplot(data=df,x="highervocational_school_percentage",kde=True)
```




```
ax = sns.histplot(data=df,x="high_school_percentage",kde=True)
```



```
ax = sns.histplot(data=df,x="population_density",kde=True)
```



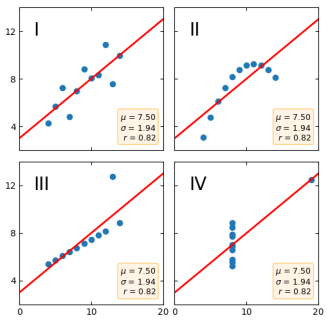
After we plot all the variables that are strongly correlated to the PAD per Capita we see they all have a skewed distribution. Intuitively this makes sense with as we know that majority of the income is concentrated in certain cities, based on our initial visualization, and so do the factors that positively contribute to a city's Income.

Importance of plotting/visualization

After displaying the correlation, we want to plot it to ensure that our variables are indeed correlated.

Anscombe's quartet

This phenomena occurs when a group of datasets have descriptive statistics that are nearly identical to each other and yet are very different when we visualize them



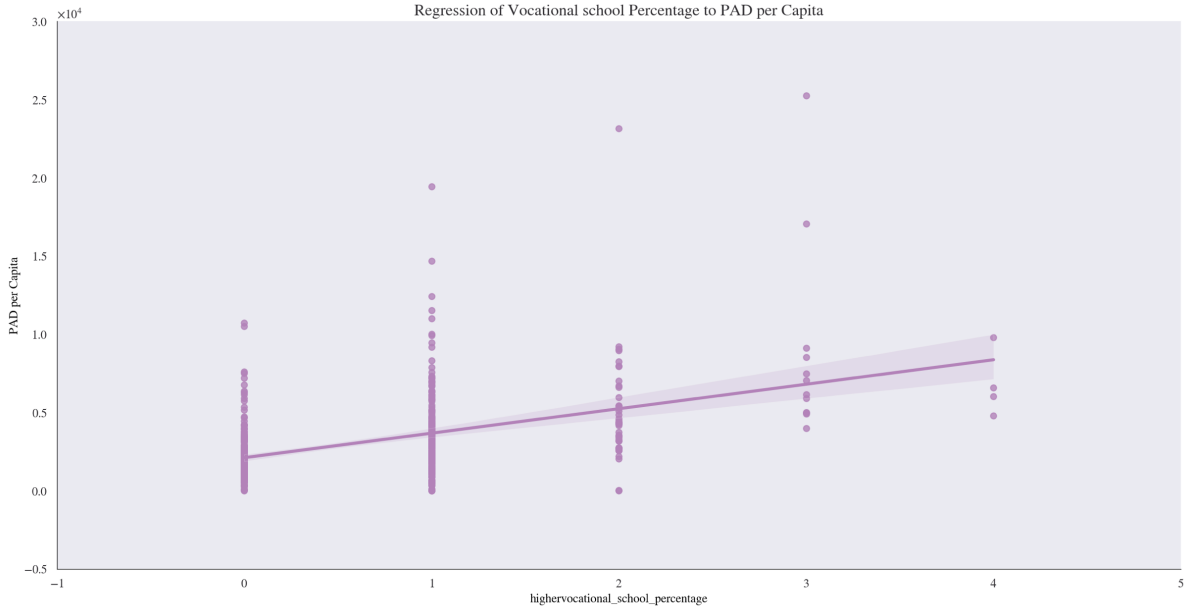
data source : matplotlib[9]

From the figure above can see despite the 4 separate datasets having identical descriptive statistics, yet, they are actually different.

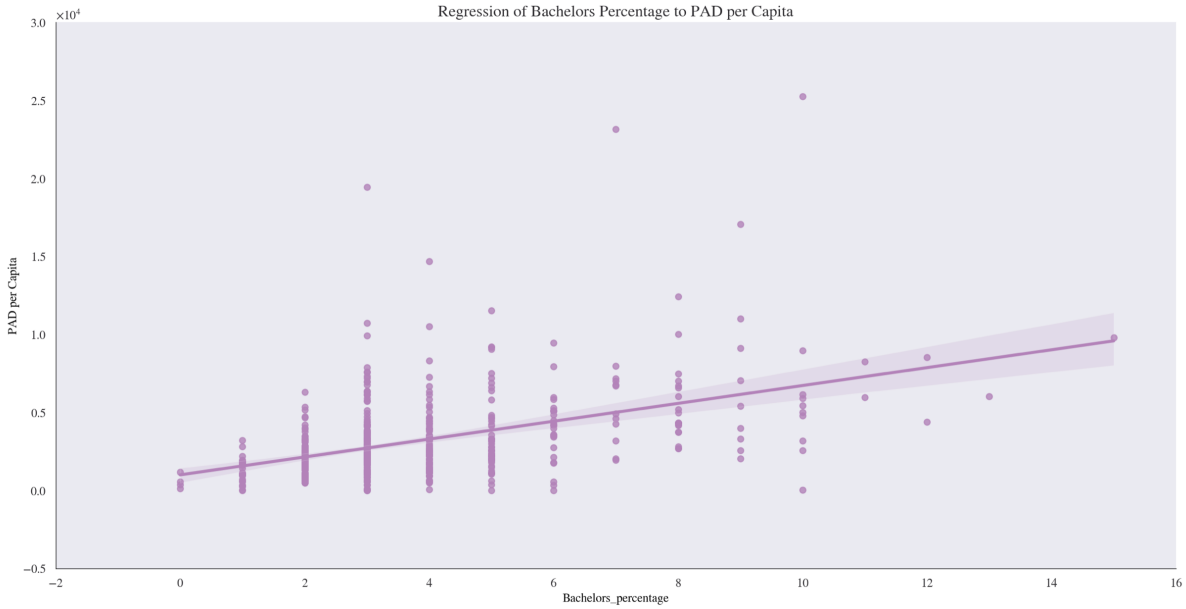
Now we can plot a scatterplot and add a regression line to see if there actually is a relationship

```
ax = sns.regplot(x="high_school_percentage", y="PAD per Capita", data=df)
ax.set_title("Regression of Highschool Percentage to PAD per Capita", loc="center")
```


```
ax = sns.regplot(x="highervocational_school_percentage", y="PAD per Capita", data=df)
ax.set_title("Regression of Vocational school Percentage to PAD per Capita", loc="center")
```

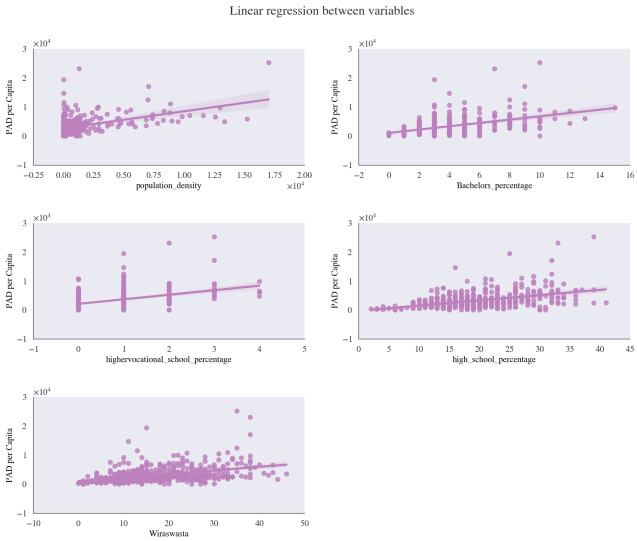



```
ax = sns.regplot(x="Bachelors_percentage", y="PAD per Capita", data=df)
ax.set_title("Regression of Bachelors Percentage to PAD per Capita", loc="center")
```



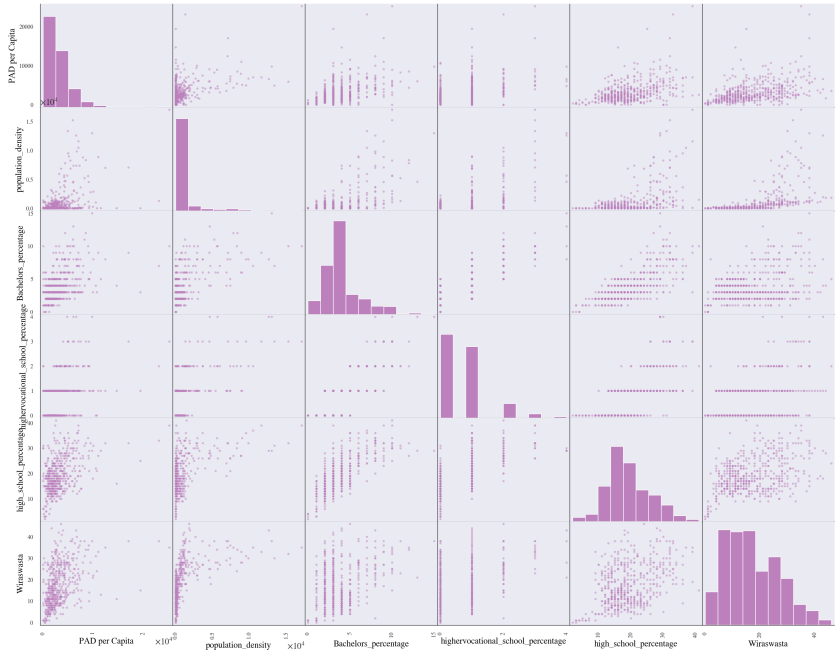
We can use subplots to make it more our regression more compact. This also prevent us from repeating our code.

```
# using subplots
plt.figure(figsize=(15, 12))
plt.subplots_adjust(hspace=0.5)
plt.suptitle("Linear regression between variables", fontsize=18, y=0.95)

# loop through our list of variables
for n, col in enumerate(highest_correlation_list):
    # add a new subplot iteratively
    ax = plt.subplot(3, 2, n + 1)

    sns.regplot(x=col, y="PAD per Capita", data=df,ax=ax)

    # chart formatting
    ax.set_xlabel(col)
plt.savefig('subtplot_reg.png')
```



An alternative to plotting the regression is by using pandas's `scattermatrix()`

```
highest_pos_corr_pad = df.corr()[['PAD per Capita']].sort_values(by='PAD per Capita',ascending=False).head(6)
highest_correlation_list = list(highest_pos_corr_pad.index)
highest_correlation_list # list of cols with the highest correlation
scatter_matrix(df[highest_correlation_list],figsize=(20,16))
plt.show()
```



By doing so, we can see the relationship of each variable not only to our dependent variable but also to other variables. The advantage of this to a heat map is we can actually see how the relationships look like when they are visualized.

Querying our data

Based on the our figures, we see that level of education (bachelors, vocational, high school etc) strongly correlate with a cities income. Let see which cities that have high income and yet have a level of education that is below average.

We will first see the percentile values of our different features

```
descriptive_table(df[['PAD per Capita','Bachelors_percentage',
'highervocational_school_percentage','high_school_percentage',
'population_density']])
```

	count	mean	std	min	25%	50%	75%	max
PAD per Capita	507.0	3171.7	2752.9	0.0	1550.1	2439.5	4048.0	25239.9
Bachelors_percentage	507.0	3.8	2.2	0.0	2.0	3.0	5.0	15.0
highervocational_school_percentage	507.0	0.7	0.8	0.0	0.0	1.0	1.0	4.0
high_school_percentage	507.0	19.2	6.8	2.0	14.0	18.0	23.0	41.0
population_density	507.0	957.0	2184.5	1.3	55.1	139.8	789.9	17037.2

Now we just extract the values below the 50th percentile and specify the multiple conditions to filter the results

```
low_educ_high_income = df.loc[(df['PAD per Capita']>=100) &
(df['Bachelors_percentage']< 3) & (df['high_school_percentage'] < 18) &
(df['highervocational_school_percentage'] < 1) &
(df['population_density'] < 139.8)][['Provinsi','Kabupaten/Kota',
'Bachelors_percentage','high_school_percentage',
'highervocational_school_percentage']]
low_educ_high_income
```

	Provinsi	Kabupaten/Kota	Bachelors_percentage	high_school_percentage	highervocational_school_percentage
20	ACEH	ACEH TIMUR	2	14	0
41	SUMATERA UTARA	PADANG LAWAS	2	14	0
45	SUMATERA UTARA	NIAS UTARA	2	9	0
79	RIAU	KEPULAUAN MERANTI	2	14	0
81	RIAU	INDRAGIRI HILIR	2	12	0
84	RIAU	KAMPAR	2	14	0
87	RIAU	ROKAN HILIR	1	17	0
88	RIAU	ROKAN HULU	2	14	0
98	JAMBI	TANJUNG JABUNG BARAT	2	15	0
99	JAMBI	TANJUNG JABUNG TIMUR	2	12	0
100	JAMBI	TEBO	2	12	0
102	SUMATERA SELATAN	BANYUASIN	2	14	0
105	SUMATERA SELATAN	OGAN KOMERING ULU SELATAN	1	14	0
107	SUMATERA SELATAN	MUSI RAWAS UTARA	1	11	0
108	SUMATERA SELATAN	LAHAT	2	17	0
109	SUMATERA SELATAN	MUSI BANYUASIN	2	13	0
110	SUMATERA SELATAN	MUSI RAWAS	1	12	0

These are just some of the cities that have a level of education that is below average (some of the cities here have level of education within the 25th percentile) and yet they have income above the 75th percentile. This does not add up with our

correlation results and the scatter plots. There must be a variable that we are not accounting for.

If we search the sources of income from the cities above, it is mostly from natural resources. We do not have data for natural resources in our dataset and this is why certain cities have high income despite their having low levels of education. There might also be other factors that we did not take into account to which we will explore in the modeling section.

Geospatial visualization (Advanced)

This section is optional and can be skipped. There is however a data cleaning section which shows how to join data from multiple databases with inconsistent naming that is worth reading.

To get a clearer picture of our data, as we are dealing with the income of different cities in Indonesia, we want to see how the distribution of income looks from a spatial perspective. Prior to this, we visualized the income per capita by plotting the distribution. While it was useful to see that the income was positively skewed, It did not tell us where in Indonesia the cities that had high income were located. Perhaps their spatial location could give us a clue on why they had such high incomes. Using GeoPandas, we can load, manipulate and visualize spatial data.

Note: The specific details of geospatial visualization is beyond the scope of this guide. In fact, we could dedicate a whole separate guide for geospatial visualization alone. Rather, we will use geospatial visualization here to show how this could be useful for public policy.

In order to spatially visualize our data, we must get the .shp file of our area of interest. A shp file contains vector features of our area of interest. In our case these are all the shapes of all the villages in Indonesia. This data at the village level of a country is publicly available in the geospatial information agency.

We are going to be combining the shp file, with our current dataset which has information containing income, level education, etc.

A separate guide can be dedicated for geospatial visualization and analysis. For a more comprehensive guide refer to : https://geopandas.org/en/stable/docs.html

Load dataset

```
full_data = gpd.read_file('Batas Desa terbaru Maret 2020/Batas Desa terbaru Maret 2020.shp')
city_df = pd.read_excel('normalized_gp2_data.xlsx') # reload df
```

Subset the data extracting only columns of interest. In the data that we are using, we have data at the village level, since we are only interest in at the city level, we find the column containing the city name and also the geometry of each village. Since the naming depend on the author of the dataset, you should read the documentation ahead of time to identify these columns

```
geo_df = geo_df[['WADMKK','geometry']]   # subsetting columns
geo_df = geo_df.dropna(axis=0) # remove missing data
```

How to aggregate spatial data

Since the data that we have is at the village level, we must aggregate it to make it city level. The The `dissolve()` function does just that.

```
geo_df['geometry'] = geo_df.buffer(0.01) # prevents overlap
geo_df = geo_df.dissolve(by='WADMKK') # group our data points by city
geo_df.plot()
geo_df["kota"] = geo_df.index # we want then to extract the city name
```

Data cleaning : Joining datasets with inconsistent names

When dealing with datasets with multiple sources, one of the biggest challenges is the inconsistency in naming. Right now, we want to combine our dataset containing the income per capita of each city and the dataset containing the location. The city names however are not consistent with each other. How can we then combine our datasets together ?

Fortunately a library called fuzzywuzzy allows us to join data based on names the are **similar** to each other. This can save us days or even weeks !

Refer to this guide to use fuzzywuzzy : https://www.geeksforgeeks.org/fuzzywuzzy-python-library/

```
# merge data
city_df['key']=city_df['Kabupaten/Kota'].apply
(lambda x : [process.extract(x, geo_df['kota'], limit=1)][0][0][0])
merged_df = city_df.merge(geo_df,left_on='key',right_on='kota')
merged_df = gpd.GeoDataFrame(merged_df, crs="EPSG:4326", geometry='geometry') # convert our data to geopandas data frame
```

The code in the first line is telling our program to find strings that are similar to each other between these 2 columns from our 2 different datasets.

Exploring our spatial data

In the previous section, we saw that level of education (Bachelors, Vocational, High school) were the most important factors to determine a cities income level, we want to see how this holds up by visualizing which areas have the highest level of education and then overlaying our data with the percentage of the population with a bachelors degree. Again, we select only our columns of interest and we can spatially visualize our data.

```
merged_df = merged_df[['Kabupaten/Kota','PAD per Capita','geometry','Bachelors_percentage','highervocational_school_percentage','high_
merged_df['Kabupaten/Kota'] = merged_df['Kabupaten/Kota'].astype(str)
```

```
def truncate_colormap(cmap, minval=0.0, maxval=1.0, n=100):
    new_cmap = colors.LinearSegmentedColormap.from_list(
        'trunc({n},{a:.2f},{b:.2f})'.format(n=cmap.name, a=minval, b=maxval),
        cmap(np.linspace(minval, maxval, n)))
    return new_cmap

arr = np.linspace(0, 50, 100).reshape((10, 10))
fig, ax = plt.subplots(ncols=2)

cmap = plt.get_cmap('RdPu')
new_cmap = truncate_colormap(cmap, 0.3, 1)

ax[0].imshow(arr, interpolation='nearest', cmap=cmap)
ax[1].imshow(arr, interpolation='nearest', cmap=new_cmap)
plt.show()
```

Then we choose which variables we want to overlay our map with. Here we selected Bachelors percentage, but this could be any variable we are interested in.

```
# gdp per capita
fig, ax = plt.subplots(1, figsize=(12,8))
merged_df.plot(column='PAD per Capita', cmap=new_cmap, linewidth=1, ax=ax, edgecolor='0.9', legend = True)
ax.axis('off')
ax.set_title('Indonesian cities/regencies by GDP Per Capita', fontdict={'fontsize': '15', 'fontweight' : '3'})
fig.savefig("Indonesia_gdp_spatial.png", dpi=300)
```


```
# bachelors percentage
fig, ax = plt.subplots(1, figsize=(12,8))
merged_df.plot(column='Bachelors_percentage', cmap=new_cmap, linewidth=1, ax=ax, edgecolor='0.9', legend = True)
ax.axis('off')
ax.set_title('Indonesian cities/regencies by Bachelors', fontdict={'fontsize': '15', 'fontweight' : '3'})
fig.savefig("Indonesia_gdp_spatial_educ_bach.png", dpi=300)
```

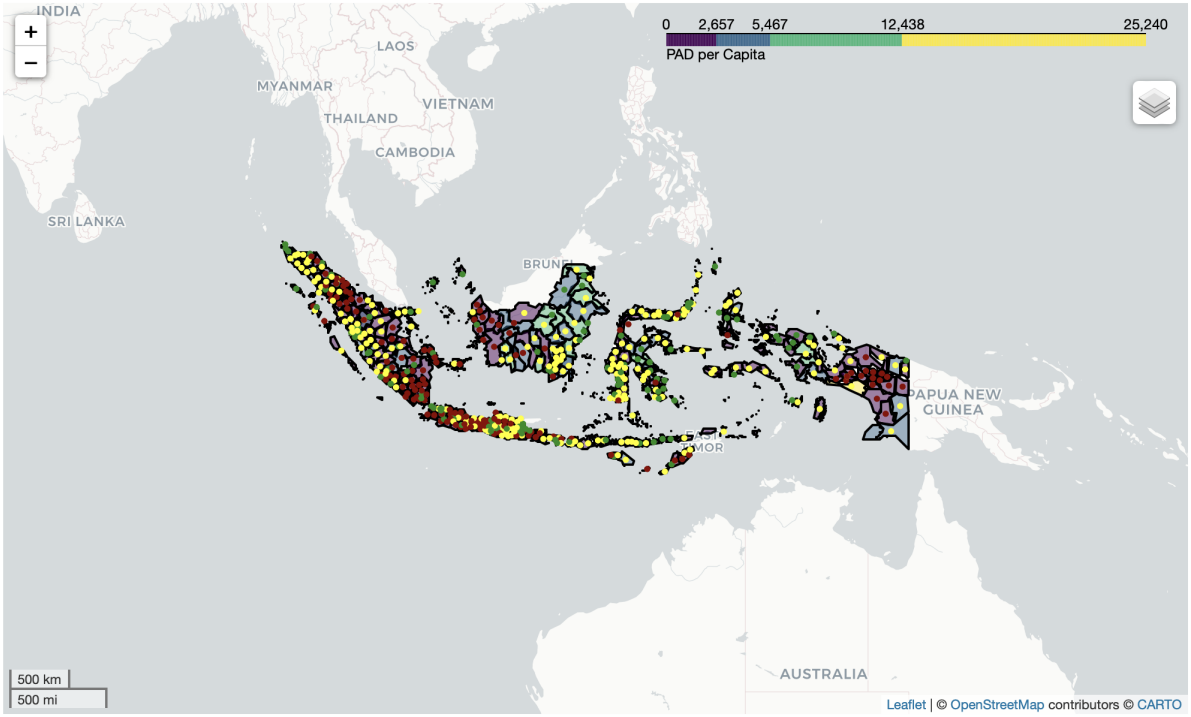

```
# highschool percentage
fig, ax = plt.subplots(1, figsize=(12,8))
merged_df.plot(column='high_school_percentage', cmap=new_cmap, linewidth=1, ax=ax, edgecolor='0.9', legend = True)
ax.axis('off')
ax.set_title('Indonesian cities/regencies by highschool', fontdict={'fontsize': '15', 'fontweight' : '3'})
fig.savefig("Indonesia_gdp_spatial_educ_high.png", dpi=300)
```

Interactive visualization

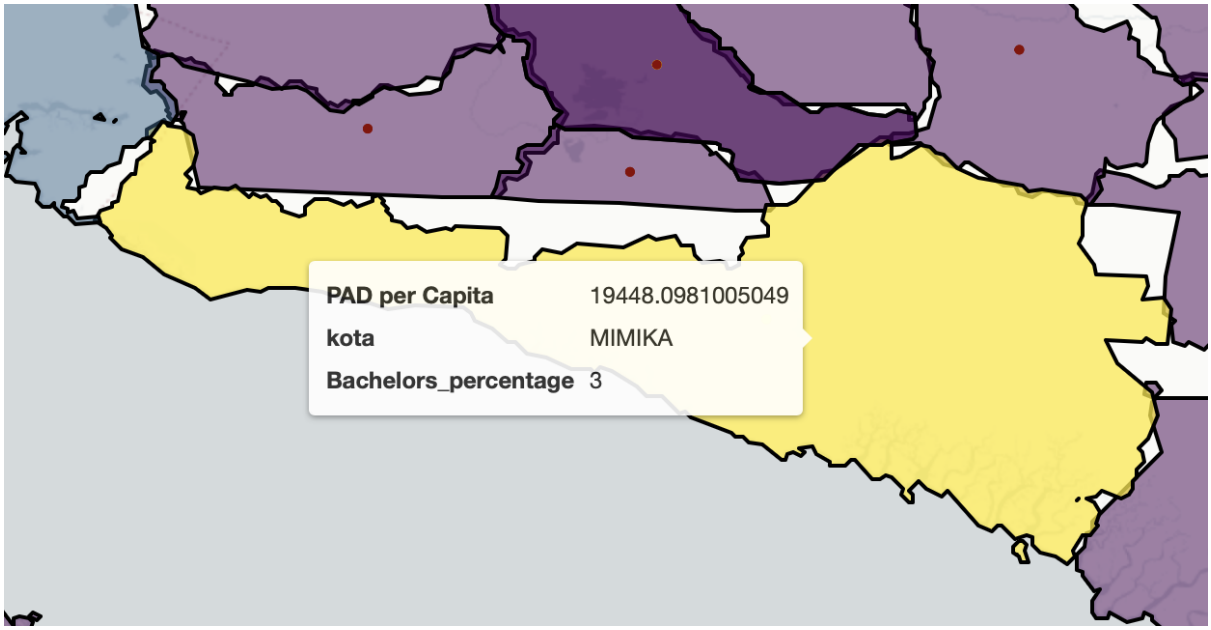
Within the geopandas documentation, there are examples on how to create interactive maps. We can create a map where centroids would represent the percentage of the population who have attained a bachelors degree and the color of the city represent the level of income. By doing so we can see if the level of education is a consistent determinant of a city's income.

```
merged_df['kota'] = merged_df.index
m = merged_df.explore(
    column = 'PAD per Capita',
    tooltip=['PAD per Capita','kota','Bachelors_percentage'], # show "BoroName" value in tooltip (on hover)
    popup=True, # show all values in popup (on click)
    tiles="CartoDB positron", # use "CartoDB positron" tiles
    scheme="naturalbreaks", # use "Set1" matplotlib colormap
    k=4, # number of categories
    name='kota',
    style_kwds=dict(color="black") # use black outline
)
folium.LayerControl().add_to(m)

# get centroids to color our
merged_df = test_gdf.to_crs(4326)
merged_df['lon'] = test_gdf.centroid.x
merged_df['lat'] = test_gdf.centroid.y
```

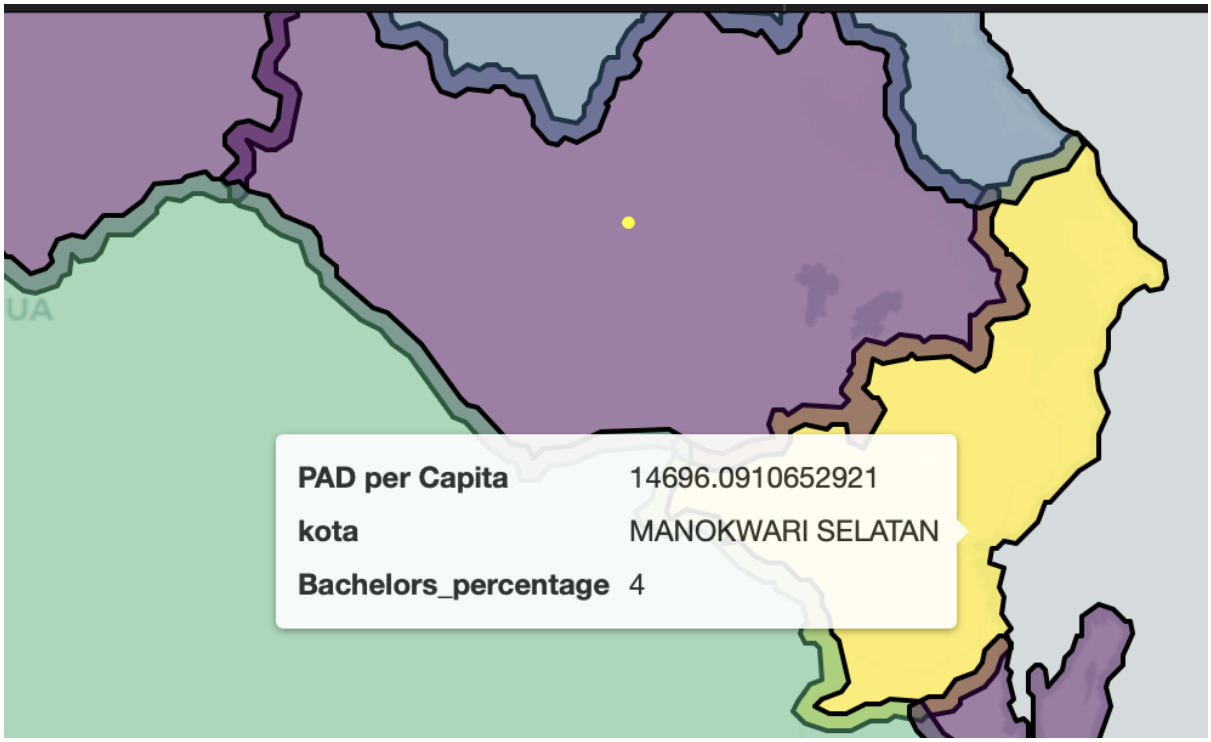
```
# overlay with educ
def color_producer(val): # mapping colors to level of education
  if val  >= 5 :
    return 'forestgreen'
  elif val >= 3:
    return 'yellow'
  else:
      return 'darkred'

# Add a bubble map to the base map
for i in range(0,len(merged_df)):
    Circle(
        location=[merged_df.iloc[i]['lat'], merged_df.iloc[i]['lon']],
        radius=20,
        legend = True,
        color=color_producer(merged_df.iloc[i]['Bachelors_percentage'])).add_to(m)

# Display the map
m
```



Lets zoom in on cities that have high income



Mimika has a relatively low bachelors percentage and yet it has extremely high income



The same is true for Manokowari Selatan. When we search what are the major sources of income for the both cities, we can see it come from natural resources. This is why, despite some of the cities not having a high level of education (inconsistent with the findings of our correlation), they had a high income. This is an important variable that we do not have in our dataset.

We can try our regression analysis including our outliers cities, most likely the model will perform badly and so we will remove them later.

Mutual information

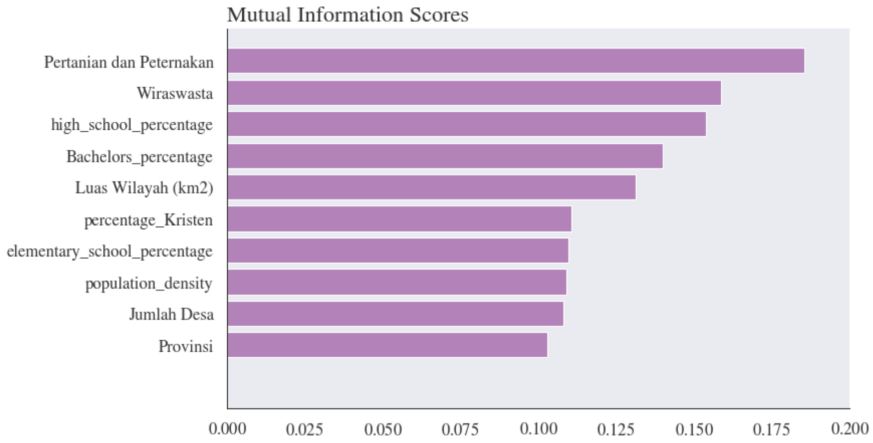
When dealing with data with a large amount of features it is beneficial to identify which are the potentially key features. To do so, we calculate the mutual information scores. What is measures is how much information is lost to predict/classify our target variable when a feature is removed. The advantage of using mutual information over pearson correlation is that it is not limited to identifying just linear relationships. However, with mutual information we only see how useful the variables are on their own, simply selecting variables from mutual information alone might actually be detrimental for our model performance, that said, it can give us a clear idea on what variables we know for sure are important

```
def plot_mi_scores(scores):
    scores = scores.sort_values(ascending=True)
    width = np.arange(len(scores))
    ticks = list(scores.index)
    plt.barh(width, scores)
    plt.yticks(width, ticks)
    plt.title("Mutual Information Scores")
```

```
# mutual information
X = df.copy()
X = X.drop(columns=['Kabupaten/Kota'],axis=1)
y = X.pop('PAD per Capita')

# Label encoding for categoricals
for colname in X.select_dtypes("object"):
    X[colname], _ = X[colname].factorize()

# All discrete features should now have integer dtypes (double-check this before using MI!)
discrete_features = X.dtypes == int
```

After transforming our data we can then call our functions

```
mi_scores = mutual_info_regression(X, y)
mi_scores = pd.Series(mi_scores, name="MI Scores", index=X.columns)
mi_scores = mi_scores.sort_values(ascending=False)

# plot our results
plt.figure(dpi=100, figsize=(8, 5))
plot_mi_scores(mi_scores.head(10))
```



Feature engineering : Creating new features

It might very difficult and not relevant to predict the exact income of our cities. Instead we can group them by their percentiles and classify them from low income to very high income

```
income_df = df[['PAD per Capita','Kabupaten/Kota']]
desc_table = descriptive_table(income_df) # we use this are our reference to group
desc_table
```

	count	mean	std	min	25%	50%	75%	max
PAD per Capita	507.0	3171.7	2752.9	0.0	1550.1	2439.5	4048.0	25239.9

Before grouping them, we want to remove outliers from our data as they will affect our model performance. We can remove them by using the formula for outliers written below.

```
iqr = desc_table['75%'] - desc_table['25%']
outlier =  desc_table['75%'] + (1.5* iqr)
print('Any provinces with an income higher than',outlier,'will be flagged as an outlier')
```

```
Any provinces with an income higher than 7794.85 will be flagged as an outlier
```

We will also removed areas that have a income of 0 as they too are outliers

```
# group them based on which percentiles the city income lies in
income_conditions = [
    (df['PAD per Capita'] <= 0),
    (df['PAD per Capita'] < float(desc_table['25%'])),
    (df['PAD per Capita'] >=  float(desc_table['25%'])) & (df['PAD per Capita'] <= float(desc_table['75%'])),
    (df['PAD per Capita'] >= float(desc_table['75%'])) & (df['PAD per Capita'] < float(outlier)),
    (df['PAD per Capita'] >= float(outlier))]
income_categories = ['very low','low', 'medium', 'high','very high']
df['income_categories'] = np.select(income_conditions, income_categories)

# check output of our transformation
df[['Kabupaten/Kota','PAD per Capita','income_categories']].tail(10)
```

	Kabupaten/Kota	PAD per Capita	income_categories
497	POLEWALI MANDAR	3645.251563	medium
498	MAMASA	378.180931	low
499	PASANGKAYU	1459.136268	low
500	MAMUJU TENGAH	1662.012546	medium
501	BULUNGAN	5820.992719	high
502	MALINAU	4497.088801	high
503	NUNUKAN	6168.092768	high
504	KOTA TARAKAN	4304.448661	high
505	TANA TIDUNG	4596.951864	high
506	DKI JAKARTA	25239.854568	very high

When we spatially visualized our dataset before, we saw that certain cities had high income due to a confounding variable which was natural resources. Since our dataset does not contain data for these factors, we will drop them . Other cities such as Jakarta and Badung also have very high income. This is also a special case as Jakarta is the capital city, where most of the development is concentrated in. Badung is in Bali where there are a lot of expats and high tourism leading it to have very high income revenue. There are also certain cities that depend on government transfers and have an Income of 0. These are due to special circumstances and hence we will also drop them

```
df = df[df["income_categories"] != 'very high']
df = df[df["income_categories"] != 'very low']
```

Feature Engineering ：Creating new features

We saw that from our spatial analysis, there is correlation between income of a regency/city and its surrounding. Let us then create a new feature, get the median income for cities in each province.

```
# get average marketcap per year
df["Median_income_province"] = (
    df.groupby("Provinsi")
    ["PAD per Capita"]
    .transform("median")
)

df[["Kabupaten/Kota", "Provinsi", "Median_income_province","PAD per Capita"]].head(5)
```

:

	Kabupaten/Kota	Provinsi	Median_income_province	PAD per Capita
0	PIDIE	ACEH	3495.361372	3495.361372
2	KOTA BANDA ACEH	ACEH	3495.361372	6026.856452
4	KOTA LANGSA	ACEH	3495.361372	3994.683484
5	KOTA LHOKSEUMAWE	ACEH	3495.361372	2206.225070
6	GAYO LUES	ACEH	3495.361372	3502.447078

Feature Engineering ：Dealing with categorical data

In our Province column, we are dealing with categorical data. Our model cannot interpret words/strings. There are several ways to deal with this. In this situation we will be using dummy variables to represent the different provinces.

Essentially, what this does is we create an individual column for each province and encode it with a 1 if our sample is in a Province and 0 otherwise. By doing so, we will create 34 additional features! This may not necessarily be good for us as this creates high dimensionality. Ultimately, it will depend on our dataset. Below, we will show how to deal with categorical features nonetheless for demonstration purposes

```
df = pd.get_dummies(df, columns=["Provinsi"]) # simply select columns containing categorical data
df.head()
```

e	Provinsi_ACEH	Provinsi_BALI	Provinsi_BANTEN	Provinsi_BENGKULU	Provinsi_DAERAH ISTIMEWA YOGYAKARTA	Provinsi_GORONTALO	Provinsi_JAMBI	Provinsi_JAWA BARAT	Provinsi_JAWA TENGAH	Provinsi_JAWA TIMUR	P
2	1	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	0	0	0	

Data modeling

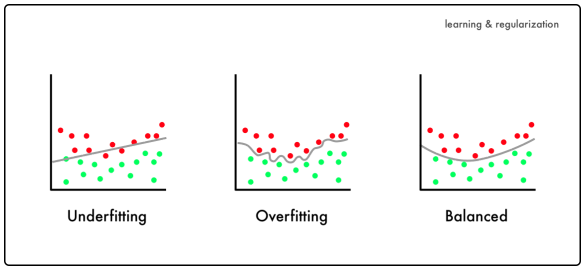
There are many machine learning models available. In our we will use the 2 most commonly used classifier algorithms . One is Random Forest, an algorithm that combines multiple decision trees and XGBoost, which are a boosted version of Random Forest. The official documentation is accessible here:

- Random forest : https://scikit-learn.org/stable/modules/ensemble.html
- XGBoost: https://xgboost.readthedocs.io/en/stable/

Before modeling our data we must first establish how we will evaluate our models

Evaluation methods

Before we continue modeling, We have to establish how to evaluate or models. This process is performed using training and test sets. Cross-validation is implemented in machine learning to prevent overfitting. This is when the dataset is divided into a training set and a test set. Typically, this is performed by allocating 75-80% of the data to the training set and the rest to the test set. The training data are used to train the models by finding the right coefficient, whereas the test set is used to test its performance on a dataset that has never been seen. This was used to verify whether the model generalizes well. Unfortunately this step often gets skipped by domain scientist that lead to unreproducible results!



source : towardsdatascience.com

Why is this important ?

Ultimately, the goal of statistical inference is to know whether our model generalizes well. What works in area may not work in other areas. For public policy, this is important for policy makers as often times they make decision based on historical events. But how do we know the same policies will be effective in other areas?

In the code below, call `train_test_split` which will split our dataset for us. By default it will produce a 75:25 split but we can change these values.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

y = df[['income_categories']]
#X = X.loc[:,~X.columns.duplicated()] # certain cols were duplicates
X = df.copy()
X = X.drop(columns=['Kabupaten/Kota','PAD per Capita','income_categories'],axis=1)
train_X, val_X, train_y, val_y =
train_test_split(X, y, random_state=1,shuffle=True)

my_model = RandomForestClassifier(n_estimators=8000,
                                  random_state=23).fit(
```

```
                                train_X, train_y.values.ravel())
y_pred = my_model.predict(val_X)
print("Accuracy:",metrics.accuracy_score(val_y, y_pred))
```

```
Accuracy: 0.6833333333333333
```

XGBoost

```
xgb_model = xgb.XGBClassifier(objective='multi:softmax', n_estimators=50,
                              learning_rate=0.07,max_depth=5,gamma=5,colsample_bytree=0.7) # parameters were found using grid search.

xgb_model.fit(train_X, train_y.values.ravel())
y_pred = xgb_model.predict(val_X)
print("Accuracy:",metrics.accuracy_score(val_y, y_pred))
```

```
Accuracy: 0.7
```

Grid search / Hyper-parameter tuning

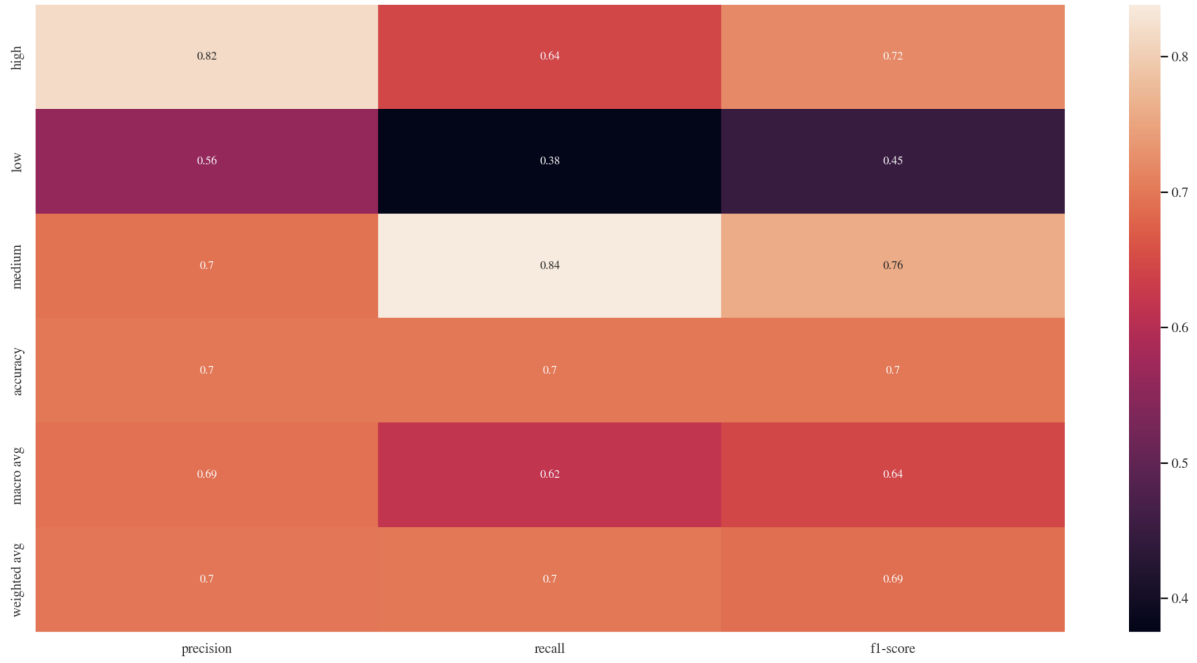
Look at the hyper-parameters inside our `xgb.XGBClassifier()` How do we decide what are the ideal hyper-parameters (learning_rate, max_depth, gamma etc.)? There is no exact method to find the optimal values. Usually, finding the ideal values require trial and error, but that takes a long time. To automate this process we can use a grid search shown below. What this does is try every possible permutation to find which one produces the best result. The more options we give, the longer the process will take but the higher probability we find the optimal number. That decision depends on the time we have.

```
gbm_param_grid = {
    'learning_rate' : [0.1],
    'colsample_bytree': [0.8,0.7,0.9],
    'n_estimators': [60,75,77],
    'max_depth': [7,10,12],
    'gamma' : [0.1,0.05],
    'objective': ['multi:softmax']
}

xgb_model = xgb.XGBClassifier()
grid_acc = GridSearchCV(estimator=xgb_model,param_grid=gbm_param_grid, cv=2, verbose=1)
grid_acc.fit(X,y.values.ravel())

print("Best parameters found: ", grid_acc.best_params_)
print( "best score : ",grid_acc.best_score_)
```

Evaluating model performance

Even though we have seen the accuracy of our model, when dealing with a classifier model, we want to see which classes our model is good at predicting vs which classes it struggles at predicting. By using the `classification_report` function, we can easily see this. The classification report outputs a confusion matrix which tell us how our model performed for each class (low, medium, high) .

```
class_rep = classification_report(val_y,y_pred,output_dict=True)
sns.heatmap(pd.DataFrame(class_rep).iloc[:-1, :].T, annot=True)
```



When dealing with classification algorithms, we are dealing with multiple classes. Our model here is trying to predict which cities should be classified as low, medium and high income. To properly evaluate our model's performance, we should evaluate our models performance for individual classes. The metric of interest is the dataset is the f1-score. From our confusion matrix, we can see that the model does a good job in predicting medium and high but struggles with low income per capita cities.

Understanding our metrics

Precision : Measures how many samples that were classified in a specific category were correctly classified

Recall : How many of the samples of the specific category were classified

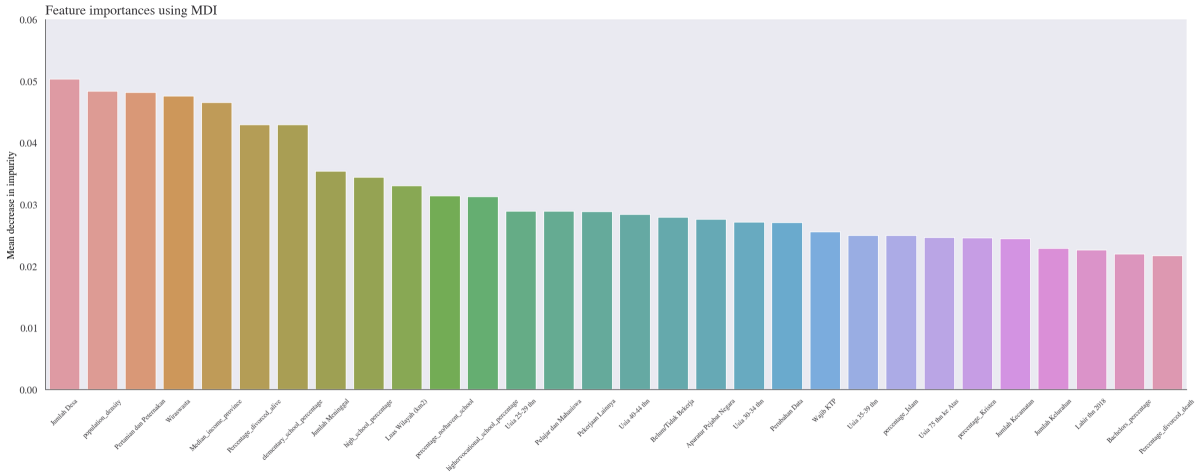
F1-score : harmonic mean of precision and recall

XGBoost feature importance

XGBoost also allows us to see which features it deems most important, this is similar to what we did before by seeing which variables have the highest correlation to the income per capita or when we used mutual information.

```
feature_names = X.columns
importances = xgb_model.feature_importances_

forest_importances = pd.Series(importances, index=feature_names)
forest_importances = forest_importances.sort_values(ascending=False).head(30)

fig, ax = plt.subplots()
#forest_importances.plot.bar(ax=ax)
ax = sns.barplot(x=forest_importances.index, y=forest_importances.values)
plt.xticks(rotation=45,fontsize=8)
```

```
plt.gcf().set_size_inches(20,8)
ax.set_title("Feature importances using MDI")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```



Interpreting our model

XGBoost and Random Forest have high accuracy. Unfortunately, this also comes with a trade-off which is interpretability. We cannot simply see the coefficients of each variable and how it impact the model. Fortunately for us, there are packages that deal with such problems!

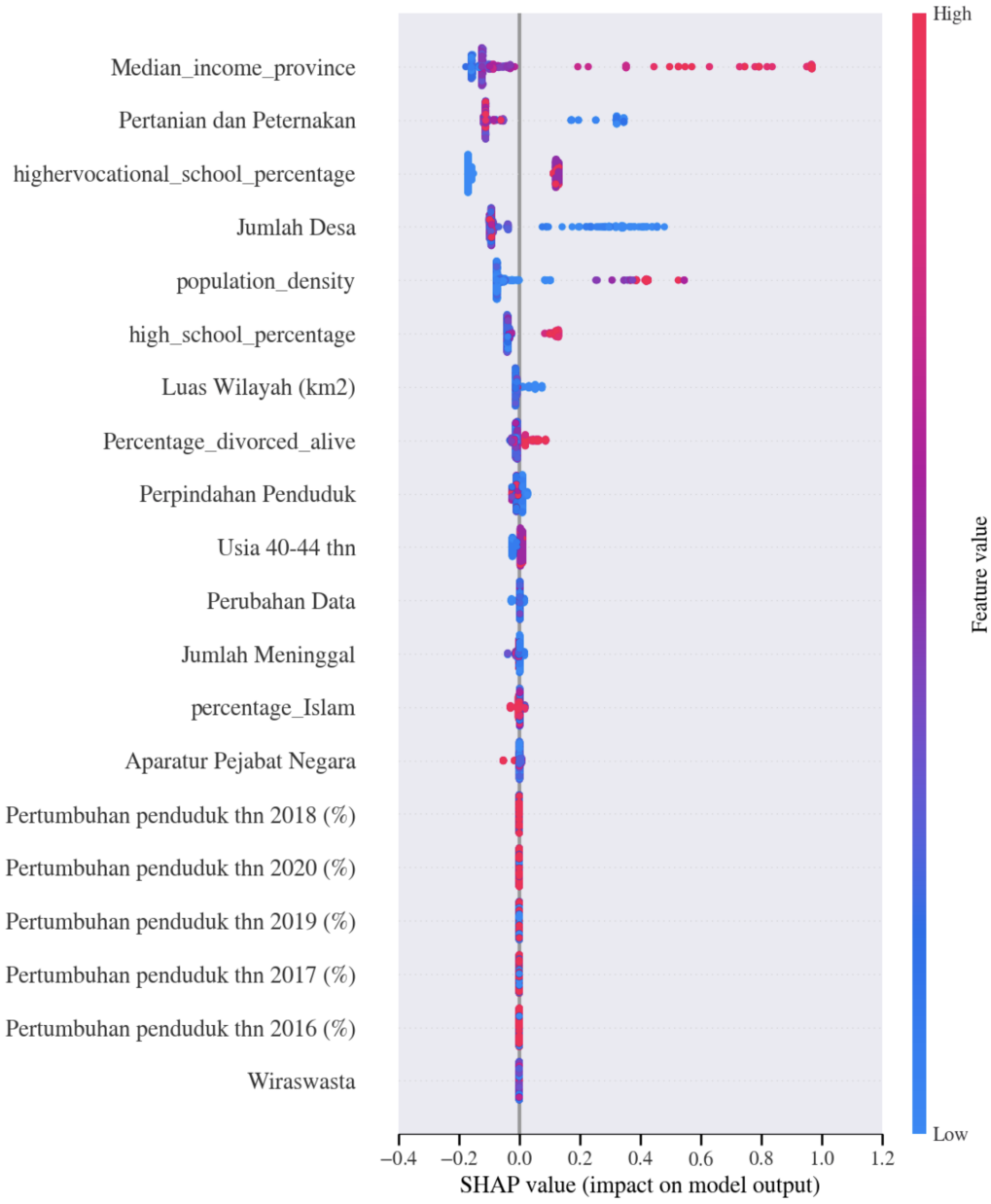
Lets see how our model decides cities with high income

Shap is a library used to explain ML models. It uses a game theory approach to see which features are most important and how each feature impacts our model. The y-axis is the features names and are arranged based on feature importance. Our x-axis shows how a value of that feature impacts the output of our model. The color of the data point indicates the range value of a particulars samples feature.

For more information on how shaps work, read the docs here : https://shap.readthedocs.io/en/latest/index.html

While by plotting feature importance was informative, we do not know how the value of each feature affects the prediction of our model. In this plot we see how our XGBoost model decides if a city gets classified in the high income category .

```
import shap
explainer = shap.TreeExplainer(xgb_model)
shap_values = explainer.shap_values(X)
shap.initjs()

# set labels for each class
class_labels = ['high','low','medium']

# plot
 shap.summary_plot(shap_values[0], X.values, feature_names = X.columns)
```



Interpreting SHAP

Why is it so important to interpret our models in the first place?

Beware of data leakage

What is data leakage? Let us say we have successfully trained a model that produces a high accuracy, great! But what if the model was able to achieve a high accuracy due to it having data that gave it an unfair advantage. For example, in our case, what if it contained data that tell us how much tax revenue a city received? Tax revenue is just the same thing as a cities income (In Indonesia a city may also receive income from city-owned enterprises). This may occur quite often when we are using a lot of data sources. This is why understanding how our model makes its predictions is important.

Understanding SHAP results

We first can see that Median Income of a province is the main contributing factor to a cities income. There are 2 potential reasons for this.

1. High income generating cities indirectly contributes to the surrounding cities income. People commute to higher income cities like Jakarta. Workers live in surrounding cities due to cheaper rent and property prices and spend the money they

earn in the city they actually live in.

2. Cities or Regencies that benefit from natural resources tend to be clustered together. Coal mines or Gold mines may span multiple cities.

How policy makers interpret our models : Correlation does not equal causation !

This topic is often discussed in classical statistics, but it also holds importance even in machine learning. One of the objectives for data scientist in public policy is to distinguish between correlating variables and causality. As discussed above, this is relevant for policymakers seeking to use ML. Often, correlating variables are useful for prediction, but there is no causal relationship between them. Confounding variables can influence the correlating variables that cause spurious relationships. Being aware of such occurrences is important, and this is where domain experts can provide excellent help.

When looking at the results we can see some interesting data, the **higher the divorce rates the more likely a city is going to be classified as high income**. Should we then encouraged people to get divorced ? Probably not. There are most likely confounding factors such as in cities where divorce rates are higher, there are more job opportunities for women leading them to be less dependent on their husbands and allowing them to get divorced compared to areas where women stay in unhappy marriages because they have no other options. This probably then means that policy makers should emphasize on better and more opportunities for women in the work place especially in lower income areas.

It can also be difficult to determine which is the causal factor. We see that population density correlates with income per capita. Does this mean simply encouraging people to live in a city would increase the income per capita or cities with better opportunities attract more people.

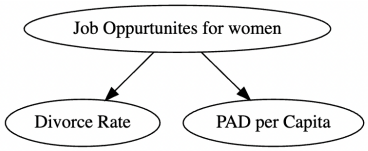
Ultimately while the ML algorithms we employed cannot derive causality but it can help researchers, economist and policy makers narrow down the search of what correlating factors are potentially causal factors. Having domain knowledge and discussing with domain experts can also enlighten us in interpreting the results. If a single sentence could be used as a takeaway for the whole module, it would be **NOT to use ML algorithms blindly !**

Below we will show examples on how to use causal inference to understand our results.

Next steps: How to derive causality from our results (brief example)

Graphing relationship between divorce rates and income

```
import graphviz as gr
g = gr.Digraph()
g.edge("Job Oppurtunites for women", "Divorce Rate"),
g.edge("Job Oppurtunites for women", "PAD per Capita"),
g
```



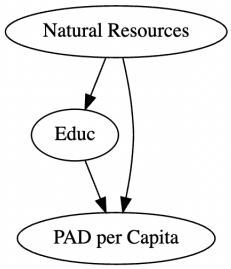
Here we use Graphical Causal Models for our example of divorce rates and higher income per capita. We can see from our model how job opportunities for women is a confounding variable to divorce rates and income.

Graphing Relationship between PAD per Capita, Income and Natural Resources plus Linear Regression

Remember when we hypothesized that natural resources was the reason why we had cities with average levels of education and yet high income. If we want to just how effective education is, we can gather data for natural resources and construct a **multilinear regression model** with education as the treatment, control the amount of natural resources and income per capita as the outcome. By controlling natural resources we can see the true impact of education to a cities income per capita. Could it be the cities with high levels of income have higher levels of education because their local government has more to spend on education?

```
g = gr.Digraph()
g.edge("Natural Resources", "Educ"),
```

```
g.edge("Natural Resources", "PAD per Capita"),
g.edge("Educ", "PAD per Capita")
g
```



The formula to construct our linear regression model would then be:

$$PADperCapita_i = \beta_0 + \beta_1 HighPerc_i + \beta_2 BachPerc_i + Resc_i + u_i$$

HighPerc = high school Percentage

BachPerc = bachelors Percentage

Resc = Natural Resources amount

u = Error Term

Improving our analysis

For public policy, understanding causality is vital. In the final section, we have seen how we can bridge Data science techniques and ML with causal inference. A great in-depth guide on causal inference can be found here : https://matheusfacure.github.io/python-causality-handbook/landing-page.html

Appendix

Reproducible environments

Installing packages and libraries in python can often be problematic. Versions of packages may not be compatible and create dependencies error. To minimize time solving these issues we must create reproducible environments. There are several ways to achieve this but using `.yml` files are an easy way to do so.

For this code notebook download the download the `.yml` file from here https://drive.google.com/file/d/1aLexkh93ajIeRVo-C7zSMPqbR37h7aiU/view?usp=sharing

Once this is downloaded open your conda terminal and create the environment

```
conda env create -f filename.yml
```

Then activate it

```
conda activate 'enviromentname'
```

Exporting your environment

If we want others to use our environment we can export our .yml file

To do so ensure the environement you want to export is activated

```
conda activate enviromentname # remove the parenthesis
```

Then export it

```
conda env export > filename.yml
```

This file can then be shared to anyone else that would like to use the environment

Additional ways to improve our model performance

As mentioned before, removing features may actually improve our model performance. This all falls under dimensionality reduction. High dimension data is when there are a lot of independent variables in comparison to our number of data points. For the most part however, it usually just speeds up training and inference time of our model.

Dimensionality reduction techniques include

1. Recursive elimination
2. PCA

Recursive elimination : https://machinelearningmastery.com/rfe-feature-selection-in-python/

PCA : https://www.simplilearn.com/tutorials/machine-learning-tutorial/principal-component-analysis

Citations

1. Amarasinghe, Kasun, et al. "Explainable. machine learning for public policy: Use cases, gaps, and research directions." *arXiv preprint arXiv:2010.14374* (2020).
2. Badan Pusat Stastik. *Analysis Of Current Issue*. BPS-Statistics Indonesia, 2021, pp. 10 - 37, 53.
3. F W Wibowo and Wihayati 2021 J. Phys.: Conf. Ser. 1844 012006
4. Leonita, G.; Kuffer, M.; Sliuzas, R.; Persello, C. Machine Learning-Based Slum Mapping in Support of Slum Upgrading Programs: The Case of Bandung City, Indonesia. *Remote Sens.* **2018**, *10*, 1522. https://doi.org/10.3390/rs10101522
5. Kaufman, Shachar, Saharon Rosset, Claudia Perlich, and Ori Stitelman. "Leakage in data mining: Formulation, detection, and avoidance." *ACM Transactions on Knowledge Discovery from Data (TKDD)* 6, no. 4 (2012): 1-21.

6. Head ML, Holman L, Lanfear R, Kahn AT, Jennions MD (2015) The Extent and Consequences of P-Hacking in Science. PLoS Biol 13(3): e1002106. https://doi.org/10.1371/journal.pbio.1002106
7. Kreif, Noemi, Karla DiazOrdaz, Rodrigo Moreno-Serra, Andrew Mirelman, Taufik Hidayat, and Marc Suhrcke. "Estimating heterogeneous policy impacts using causal machine learning: a case study of health insurance reform in Indonesia." *Health Services and Outcomes Research Methodology* (2021): 1-36.
8. Molnar, Christoph. “Interpretable machine learning. A Guide for Making Black Box Models Explainable”, 2019. https://christophm.github.io/interpretable-ml-book/.
9. “Anscombe's quartet — Matplotlib 3.5.2 documentation.” *Matplotlib*, https://matplotlib.org/stable/gallery/specialty_plots/anscombe.html. Accessed 21 July 2022.

Data used and documentation

Variable Name	Type	Definition/ English Translation	Data source	Comments
Provinsi	Categorical	Name of province	Department of Population and Civil Registration	
Kabupaten/Kota	Categorical	Name of city or regency	Department of Population and Civil Registration	
Jumlah Kecamatan	Continuous	Number of districts	Department of Population and Civil Registration	
Jumlah Desa	Continuous	Number of villages	Department of Population and Civil Registration	
Jumlah Kelurahan	Continuous	Number of wards	Department of Population and Civil Registration	
Jumlah Penduduk	Continuous	Number of population	Department of Population and Civil Registration	
Jumlah KK	Continuous	Number of families	Department of Population and Civil Registration	
Luas Wilayah (km2)	Continuous	Size of Area	Department of Population and Civil Registration	
Kepadatan Penduduk	Continuous	Population Density	Department of Population and Civil Registration	
Perpindahan Penduduk	Continuous	Migration	Department of Population and Civil Registration	
Jumlah Meninggal	Continuous	Number of deaths	Department of Population and Civil Registration	
Wajib KTP	Continuous	Number with an ID	Department of Population and Civil Registration	
Laki-Laki	Continuous	Percentage of men	Department of Population and Civil Registration	
Perempuan	Continuous	Percentage of women	Department of Population and Civil Registration	
Belum Kawin	Continuous	percentage of unmarried people	Department of Population and Civil Registration	
Kawin	Continuous	Percentage of married people	Department of Population and Civil Registration	
Cerai Hidup	Continuous	Number of divorced	Department of Population and Civil Registration	

Variable Name	Type	Definition/ English Translation	Data source	Comments
Cerai Mati	Continuous	Number of divorced due to death	Department of Population and Civil Registration	
Usia 0-4 thn	Continuous	Percentage of those between ages 0 - 4	Department of Population and Civil Registration	
Usia 5-9 thn	Continuous	Percentage of those between ages 5 - 9	Department of Population and Civil Registration	
Usia 10-14 thn	Continuous	Percentage of those between ages 10 - 10	Department of Population and Civil Registration	
Usia 15-19 thn	Continuous	Percentage of those between ages 15 - 11	Department of Population and Civil Registration	
Usia 20-24 thn	Continuous	Percentage of those between ages 20 - 12	Department of Population and Civil Registration	
Usia 25-29 thn	Continuous	Percentage of those between ages 25 - 13	Department of Population and Civil Registration	
Usia 30-34 thn	Continuous	Percentage of those between ages 30 - 14	Department of Population and Civil Registration	
Usia 35-39 thn	Continuous	Percentage of those between ages 35 - 15	Department of Population and Civil Registration	
Usia 40-44 thn	Continuous	Percentage of those between ages 40 - 16	Department of Population and Civil Registration	
Usia 45-49 thn	Continuous	Percentage of those between ages 45 - 17	Department of Population and Civil Registration	
Usia 50-54 thn	Continuous	Percentage of those between ages 50 - 18	Department of Population and Civil Registration	
Usia 55-59 thn	Continuous	Percentage of those between ages 55 - 19	Department of Population and Civil Registration	
Usia 60-64 thn	Continuous	Percentage of those between ages 60 - 20	Department of Population and Civil Registration	

Variable Name	Type	Definition/ English Translation	Data source	Comments
Usia 65-69 thn	Continuous	Percentage of those between ages 65 - 21	Department of Population and Civil Registration	
Usia 70-74 thn	Continuous	Percentage of those between ages 70 - 22	Department of Population and Civil Registration	
Usia 75 thn ke Atas	Continuous	Percentage of those above 75	Department of Population and Civil Registration	
Lahir thn 2018	Continuous	Born after 2018	Department of Population and Civil Registration	
Lahir sebelum thn 2018	Continuous	Born berfore 2018	Department of Population and Civil Registration	
Pertumbuhan penduduk thn 2016 (%)	Continuous	Change in population in 2016	Department of Population and Civil Registration	
Pertumbuhan penduduk thn 2017 (%)	Continuous	Change in population in 2017	Department of Population and Civil Registration	
Pertumbuhan penduduk thn 2018 (%)	Continuous	Change in population in 2018	Department of Population and Civil Registration	
Pertumbuhan penduduk thn 2019 (%)	Continuous	Change in population in 2019	Department of Population and Civil Registration	
Pertumbuhan penduduk thn 2020 (%)	Continuous	Change in population in 2020	Department of Population and Civil Registration	
Belum/Tidak Bekerja	Continuous	Percentage of No/ No Job yet	Department of Population and Civil Registration	
Aparatur Pejabat Negara	Continuous	Percentage of Government official	Department of Population and Civil Registration	
Tenaga Pengajar	Continuous	Percentage of school workforce	Department of Population and Civil Registration	
Wiraswasta	Continuous	Percentage Entrepeneurs	Department of Population and Civil Registration	
Pertanian dan Peternakan	Continuous	Percentage in Agriculture or farming	Department of Population and Civil Registration	
Nelayan	Continuous	Percentage of fisherman	Department of Population and Civil Registration	
Pelajar dan Mahasiswa	Continuous	Percentage of Students	Department of Population and Civil Registration	
Pensiunan	Continuous	Percentage of Retired	Department of Population and Civil Registration	
Pekerjaan Lainnya	Continuous	Other Jobs	Department of Population and Civil Registration	
f4_18_tahun_pendidikan_khusus	Continuous	Need to search or remove	Department of Population and Civil Registration	

Variable Name	Type	Definition/ English Translation	Data source	Comments
f5_6_tahun_paud	Continuous	Need to search or remove	Department of Population and Civil Registration	
f7_12_tahun_sd	Continuous	Need to search or remove	Department of Population and Civil Registration	
f12_15_tahun_smp	Continuous	Need to search or remove	Department of Population and Civil Registration	
f16_18_tahun_sma	Continuous	Need to search or remove	Department of Population and Civil Registration	
lahir_thn4	Continuous	Need to search or remove	Department of Population and Civil Registration	
lahir_thn5	Continuous	Need to search or remove	Department of Population and Civil Registration	
lahir_thn6	Continuous	Need to search or remove	Department of Population and Civil Registration	
lahir_seb4	Continuous	Need to search or remove	Department of Population and Civil Registration	
lahir_seb5	Continuous	Need to search or remove	Department of Population and Civil Registration	
lahir_seb6	Continuous	Need to search or remove	Department of Population and Civil Registration	
jml_rek_wktp	Continuous	Need to search or remove	Department of Population and Civil Registration	
percentage_Islam	Continuous	Percentage of Muslims	Department of Population and Civil Registration	
percentage_Kristen	Continuous	Percentage of Christians	Department of Population and Civil Registration	
percentage_Katholik	Continuous	Percentage of Katholik	Department of Population and Civil Registration	
percentage_Hindu	Continuous	Percentage of Hindhus	Department of Population and Civil Registration	
percentage_Budha	Continuous	Percentage of Budhist	Department of Population and Civil Registration	
percentage_Konghucu	Continuous	Percentage of Confucianist	Department of Population and Civil Registration	
percentage_no/havent_school	Continuous	Percentage of no school	Department of Population and Civil Registration	
Not_finished_elementary	Continuous	Percentage of population who have not finished elementary school	Department of Population and Civil Registration	

Variable Name	Type	Definition/ English Translation	Data source	Comments
elementary_school_percentage	Continuous	Percentage of population in elementary school	Department of Population and Civil Registration	
middle_school_percentage	Continuous	Percentage in middle school	Department of Population and Civil Registration	
high_school_percentage	Continuous	percentage of people in highschool	Department of Population and Civil Registration	
vocational_school_percentage	Continuous	percentage of people in vocational school	Department of Population and Civil Registration	
highervocational_school_percentage	Continuous	Percentage of people in higher vocational school	Department of Population and Civil Registration	
Bachelors_percentage	Continuous	Percentage of people with a Bachelors degree	Department of Population and Civil Registration	
Masters_percentage	Continuous	Percentage of people with a Masters degree	Department of Population and Civil Registration	
Phd_percentage	Continuous	Percentage of people with a Phd degree	Department of Population and Civil Registration	
Non_Islam	Continuous	Percentage of non Muslims	Department of Population and Civil Registration	
PAD per Capita	Continuous	Actual income per capita of a city or regency. City or regencies can recieve money transfer from the central government to assist them.	Ministry of Finance	
WADMKK	Categorical	Name of cities	Geospatial Information Agency	data for geospatial visualizatior
geometry	MULTIPOLYGON	Multipolygon (Shapes of the cities)	Geospatial Information Agency	data for geospatial visualizatior