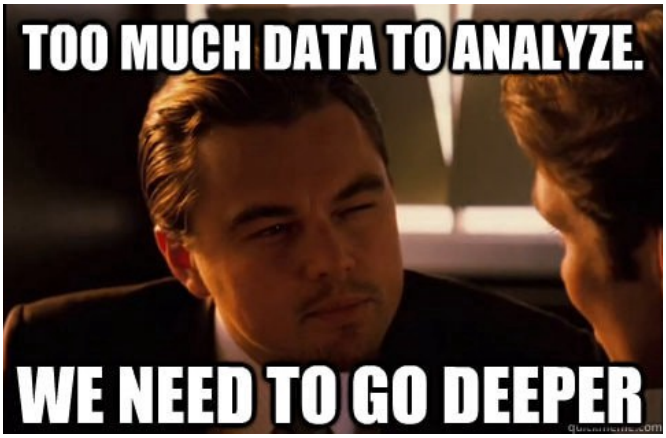
Introduction to Exploratory Data Analysis

Kaushik Mani Follow
Jan 29 · 9 min read



From the outside, it always looks like Data Science consists of advanced statistical and machine learning techniques. However, one of the most important components to any data science experiment that doesn't get as much importance as it should is Exploratory Data Analysis (EDA).

In short, EDA is "A first look at the data". It is a critical step in analyzing the data from an experiment. It is used to understand and summarize the content of the dataset to ensure that the features which we feed to our machine learning algorithms are refined and we get valid, correctly interpreted results.

In general, looking at a column of numbers or a whole spreadsheet and determining the important characteristics of the data can be very tedious and boring. EDA techniques and certain libraries provided by Python/R come to our rescue here! EDA is generally classified into two methods, non-graphical or graphical. And each method can be applied to one variable/column (univariate) or a combination of variables/columns(bivariate).

Also, it is good practice to understand the problem statement and the data before you get your hands dirty, which in view, helps to gain a lot of insights. I will try to explain the concept using the Adult dataset/ Census Income dataset available on the UCI Machine Learning Repository.

The problem statement here is to predict whether the income exceeds 50k a year or not based on the census data.

Let's start by loading the required modules to start our experiment. I will be using numpy, pandas, seaborn and matplotlib. These are the most popular ones, but feel free to use any other modules you are comfortable with.

```
In [1]: #Loading required modules
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        %matplotlib inline
```

Now, we must download the dataset and store it so that we can start using it.

```
In [2]: #Downloading the dataset
        url = "https://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"

        columns = ["age", "work-class", "fnlwgt", "education", "education-num","marital-status",
                   "occupation", "relationship","race", "sex", "capital-gain", "capital-loss",

                   "hours-per-week", "native-country", "income"]

        data = pd.read_csv(url, names=columns, sep=',', na_values='?', skipinitialspace = True)
```

Before we go further, it is always good to have a look at the various columns and what kind of attributes are present in the data so that we could get a rough idea on how to begin with our EDA.

```
In [3]: #Printing the first few values, to understand about the attributes.

        #Head data
        data.head()
```

Out[3]:

	age	work-class	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

Now based on the above result, we can classify the various data columns into numerical/categorical attributes:

1. Numerical Attributes: Age, FnlWgt, Education-num, Capital-Gain, Capital-Loss, Hours-Per-Week.
2. Categorical Attributes: WorkClass, Education, Marital-Status, Occupation, Relationship, Race, Sex, Native-Country, Income.

Data Statistics

When it comes to numerical attributes, we should start by observing various statistics like count, mean, standard deviation, etc. Let's have a look at few of those for all our numerical categories.

In [6]:
```
# Provide basic statistics for the attributes - e.g., counts, percentiles,
# mean, median, standard deviation.
# The statistics should be relevant for the type of attribute.
data.describe()
```

Out[6]:

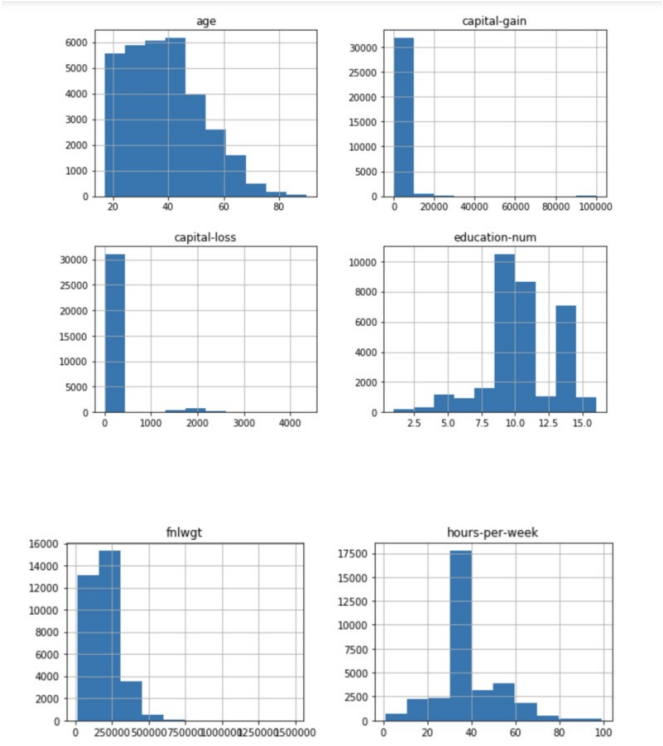
	age	fnlwgt	education-num	capital-gain	capital-loss	hours-per-week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

The statistics give us an idea about attributes containing missing values, the average value and maximum for a particular attribute, which are going to be very useful when we decide to do preprocess our data before feeding it into our model.

Data Visualization

Visualizing attributes is perhaps the most important/interesting part of EDA. Anything in this world could be better understood if we have an image/visualization of it. But, we should always keep in mind whether the visualization is appropriate for the given data type. We shouldn't just try out all the visualization techniques we know, and say we are done with EDA. Rather, we should do specific visualizations and make sure that we understand what the visualization is telling us. Let's start by plotting a histogram for all the numerical attributes.

In [13]:
```
# Plotting histogram for numerical attributes
numerical_attributes = data.select_dtypes(include=['int'])

numerical_attributes.hist(figsize=(10,12))
```



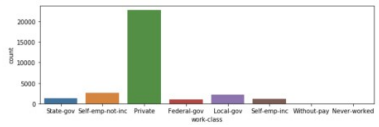
The visualization of the numerical attributes give us a few interesting insights on the distribution of values.

1. Most of the individuals have an age < 50 while the maximum age is around 90.
2. In general, people do not have investments other than their regular income. However, there are very few people who invest, and there are also a small number of outliers who earn more than 90000 via capital gains. However, among the people who had a, capital loss the average loss looks to be around 2000.
3. On average, most of the people have studied till education number 9 or 10 in the areas where the census was taken.
4. Most of the people work around 40 hrs per week. However there are a few who don't work and a few who work for almost 100 hours a week.

Now, for categorical attributes, it would be better if we could view their frequency distributions. We will use count plot to achieve this.

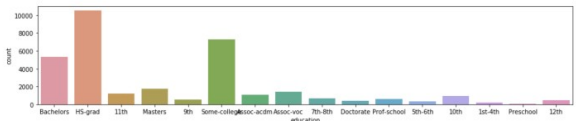
```
In [16]: # Plotting count plot for categorical values
         categorical_attributes = data.select_dtypes(include=['object'])

         # Work-class Count plot
         plt.figure(figsize=(10,3))
         sns.countplot(data = categorical_attributes, x = "work-class")
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x1a23b52d30>
```



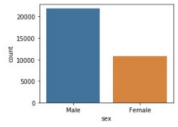

```
In [19]: plt.figure(figsize=(16,3))
         sns.countplot(data = categorical_attributes, x = "education")
```

```
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x1a240d91d0>
```

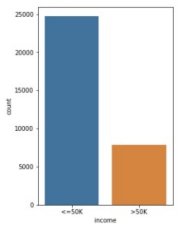



```
In [24]: # Sex Count plot
         plt.figure(figsize=(4,3))
         sns.countplot(data = categorical_attributes, x = "sex")
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x1a24908eb8>
```



```
In [18]: # Income Count plot
         plt.figure(figsize=(4,6))
         sns.countplot(data = categorical_attributes, x = "income")
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x1a16e47b00>
```



From the above plots, we can interpret the following things:

1. Most of the people work in private sectors, and the rest are evenly distributed among state-gov, federal-gov, local-gov, self-emp-inc and self-emp-not-inc.
2. Most of the people are high-school grads or have studied in some college. This is same as the education-num attribute, just that each of these values have been assigned a number there. We can use one of the two columns for our model, and ignore the other.
3. Majority of the population according to the census data are male.
4. Majority of the people have an income of less than 50k according to the data given, indicating that the data is somewhat skewed.

I haven't included the count plot of every categorical attribute here, but I will add a github link for a Jupyter Notebook at the end, which has plots of every attribute along with explanations/interpretations.

Until now, we have tried to understand the data using both visualizations and basic statistics. Another important aspect is looking at the quality of data.

Data Quality

There are a few things we always need to check when it comes to data quality.

1. Missing Values: It is possible that when the census was taken, few people refused to give information about their occupation/work-class. So, we would not have any kind of data for that column for those people. A good practice is to replace the missing values by median of the column or we could just drop these rows if they are not huge in number.
2. Outliers: It could also be that there was a person whose capital gain was $100,000 while the median capital gain was 0 dollars. These people are called outliers. It is a good practice to remove outliers from your training data, as these tend to shift your model towards the wrong results.
3. Duplicate data: We could also have information about the same person repeated in the data. We should remove all the duplicate data, as it could lead to overfitting if found in huge numbers.

```
In [19]: #Verify data quality: explain any missing values, duplicate data, or outliers.
         #Check missing values
         data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
age             32561 non-null int64
work-class      30725 non-null object
fnlwgt          32561 non-null int64
education       32561 non-null object
education-num   32561 non-null int64
marital-status  32561 non-null object
occupation      30718 non-null object
relationship    32561 non-null object
race            32561 non-null object
sex             32561 non-null object
capital-gain    32561 non-null int64
capital-loss    32561 non-null int64
hours-per-week  32561 non-null int64
native-country  31978 non-null object
income          32561 non-null object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB

In [20]: print("Missing value count occupation: ", data['occupation'].isnull().sum())
         print("Missing value count work-class: ", data['work-class'].isnull().sum())
         print("Missing value count native-country: ", data['native-country'].isnull().sum())

Missing value count occupation:  1843
Missing value count work-class:  1836
Missing value count native-country:  583
```

We can see that the missing count is not huge in number compared to the overall entries, so we can just drop these rows. We will also drop the duplicates and outliers. The histogram above showed that capital gain has outliers where very few people have an extremely high capital gain compared to most of the population.

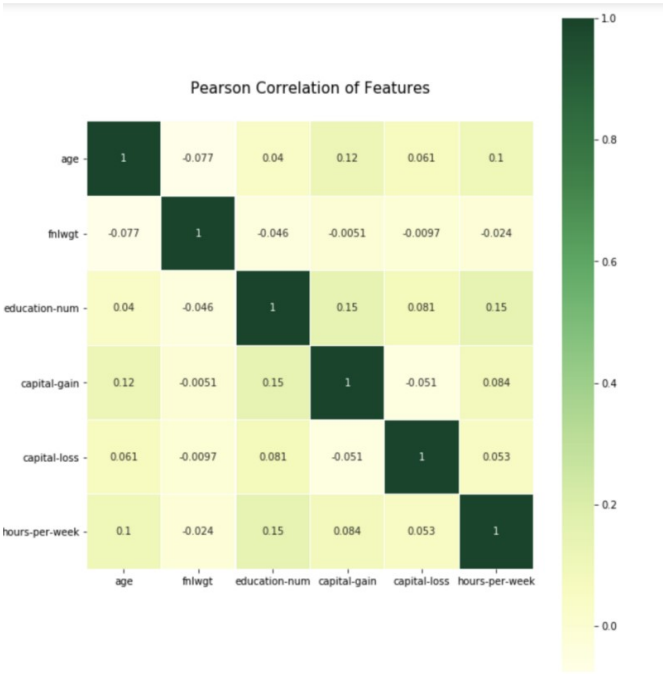
```
In [25]: # Dropping missing values
         data = data.dropna()
         # Dropping duplicate values
         data = data.drop_duplicates()
         # Dropping outlier
         i = data[data['capital-gain'] > 80000].index
         data = data.drop(i)
```

Now, that we have done all kinds of analysis and quality check on every column, let's have a look at the relationship among various columns.

Data Relationships

To explore the relationships among various attributes, we can use scatter plots and correlation heatmaps among the different attributes. We will have a look at the correlation heatmap of various attributes excluding the class attribute, i.e Income. Again, we can plot it only for numerical attributes.

```
In [26]: # Compute the correlation matrix
         corr = data.corr()

         # Set up the matplotlib figure
         f, ax = plt.subplots(figsize=(10, 12))

         # Generate a custom diverging colormap
         cmap = sns.diverging_palette(220, 10, as_cmap=True)

         # Draw the heatmap
         _ = sns.heatmap(corr, cmap="YlGn", square=True, ax = ax, annot=True, linewidth = 0.1)

         plt.title('Pearson Correlation of Features', y=1.05, size=15)
```

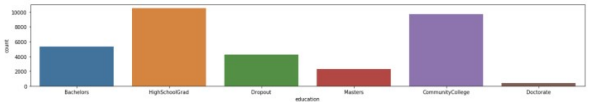


We can see that there isn't a great correlation between the numerical features.

1. The fnlwgt feature looks useless.
2. There is some correlation between age, hours per week and education-num according to the correlation heatmap.
3. Capital gain and Capital loss don't anti-correlate a lot, which says people can invest only if they have money.

For categorical attributes, we can use something called the cross-tabulation, but before we look into it, it would be better and easier for us if we group the categorical attributes into lesser number of categories. For example, I could group the education values to Dropout, HighSchoolGrad, Community College, Bachelors, Masters, Doctorate.

```
In [28]: #Grouping education
         data['education'].replace('Preschool', 'Dropout',inplace=True)
         data['education'].replace('10th', 'Dropout',inplace=True)
         data['education'].replace('11th', 'Dropout',inplace=True)
         data['education'].replace('12th', 'Dropout',inplace=True)
         data['education'].replace('1st-4th', 'Dropout',inplace=True)
         data['education'].replace('5th-6th', 'Dropout',inplace=True)
         data['education'].replace('7th-8th', 'Dropout',inplace=True)
         data['education'].replace('9th', 'Dropout',inplace=True)
         data['education'].replace('HS-Grad', 'HighSchoolGrad',inplace=True)
         data['education'].replace('HS-grad', 'HighSchoolGrad',inplace=True)
         data['education'].replace('Some-college', 'CommunityCollege',inplace=True)
         data['education'].replace('Assoc-acdm', 'CommunityCollege',inplace=True)
         data['education'].replace('Assoc-voc', 'CommunityCollege',inplace=True)
         data['education'].replace('Prof-school', 'Masters',inplace=True)

         fig = plt.figure(figsize=(20,3))
         sns.countplot(x="education", data=data)
Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x1a261a8630>
```

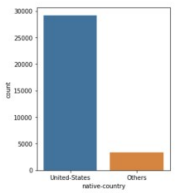


This makes our work much easier, and also visualizations much easier to interpret. We could do this for all categorical attributes which have a

lot of categories, Let's try one more column. We will divide the native-country to USA and Others.

```
In [30]: # Grouping native-country
data['native-country'] = data['native-country'].apply(lambda el: "United-States" if el == "United-States" else "Others")

# Plotting count plot for native-country
fig = plt.figure(figsize=(4,5))
sns.countplot(x="native-country", data=data)
```

```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1a26622d30>
```



Also, cross-tabulation techniques with any other column and it looks easier to interpret here.

```
In [31]: # Plotting cross tabulation values for native-country and sex
pd.crosstab(data['native-country'],data['sex'], margins=True)
```

Out[31]:

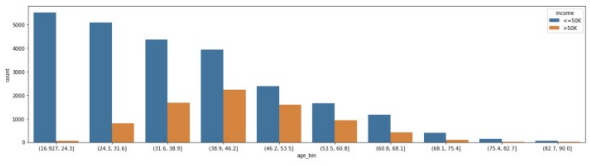
sex	Female	Male	All
native-country			
Others	1089	2302	3391
United-States	9682	19488	29170
All	10771	21790	32561

From the above results, we can say that the ratio of men and women from US as well as other natives are more or less similar, however, the number of US nationals are considerably more in the dataset.

It is also important to explore the relationship between attributes and the class-attribute. This would tell us the importance of that particular attribute in our final model. Let's look at a few relationships.

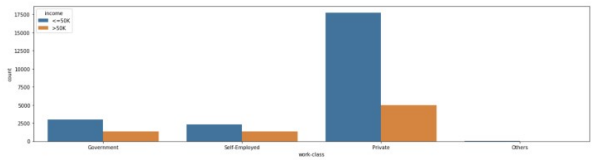
```
In [34]: #Identify and explain any interesting relationships between the class attribute and
#the other attributes.

# Dividing age into ranges
data['age_bin'] = pd.cut(data['age'], 10)

# Plotting count plot for age and income
fig = plt.figure(figsize=(20,5))
sns.countplot(x="age_bin", hue="income", data=data)
```

```
Out[34]: <matplotlib.axes._subplots.AxesSubplot at 0x1a268814e0>
```

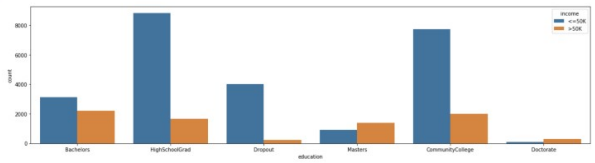


The above plot shows that experience matters to earn more income. More experienced people earn more than less experience people. This is true in the real world too. Now, we are making more more sense. Let's try it out with a couple of more attributes.

```
In [35]: #Grouping working class
data['work-class'].replace('State-gov', 'Government',inplace=True)
data['work-class'].replace('Self-emp-not-inc', 'Self-Employed',inplace=True)
data['work-class'].replace('Private', 'Private',inplace=True)
data['work-class'].replace('Federal-gov', 'Government',inplace=True)
data['work-class'].replace('Local-gov', 'Government',inplace=True)
data['work-class'].replace('Self-emp-inc', 'Self-Employed',inplace=True)
data['work-class'].replace('Without-pay', 'Others',inplace=True)
data['work-class'].replace('Never-worked', 'Others',inplace=True)

# Plotting count plot for working-class and income
fig = plt.figure(figsize=(20,5))
sns.countplot(x="work-class", hue="income", data=data)

Out[35]: <matplotlib.axes._subplots.AxesSubplot at 0x1a2681f0b8>
```




```
In [36]: # Plotting count plot for education and income
fig = plt.figure(figsize=(20,5))
sns.countplot(x="education", hue="income", data=data)

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1a266865f8>
```



From the above two plots, we can see that self-employed people in general make more money and people with higher education have a greater income.

Now, as a final step, we will combine any columns that are similar, drop unnecessary columns, and then we are done. We could start using our data to do the cool part of Machine Learning, Training the model and predicting stuff.

As a final step, I will add basic thoughts about how all columns could be combined. This is my opinion and understanding, and each person can have their own interpretations as long as they can make sense.

1. education-num—This attribute is same as education column, so we can use just one of those two.
2. fnlwgt—This attribute is useless, as it doesn't have any correlation with any attribute. We will remove this attribute.
3. capital-gain and capital-loss—We can sum up capital gain and loss to have a column "Net-Capital-Gain" rather than having two columns representing different values.
4. education—We could divide this into different classes like described above(Dropout, HighSchoolGrad, Community college, Bachelors, Masters, Doctorate). This gives us more solid categories, which could have more impact on prediction of the income.
5. marital-status—Similar to education, we could just have 4 classes like NotMarried, Married, Separated and Widowed to categorize this attribute.
6. race—Remain as it is.

7. occupation—Similar to education, we could just have 5 classes like Blue-Collar jobs, White-Collar jobs, service jobs, professional-speciality jobs, others.
8. working-class—Similar to education, we could just have 4 classes like Government, Private, Self-employed, others.
9. Sex—Remain as it is.
10. Hours-per-week—Remain as it is.
11. Income—Remain as it is.
12. Native-Country—Similar to education, we could just have 2 classes namely USA and others.

I hope you now have a basic understanding of how to play with data, or in other words, what Exploratory Data Analysis is all about. You can have a look at the complete python notebook here.

Related Stories from DDI:

Deep Learning Explained in 7 Steps

with cats

medium.com

Which is More Promising: Data Science or Software Engineering?

About a month back, while I was sitting at a café and working on developing a website for a client, I found this woman...

medium.com