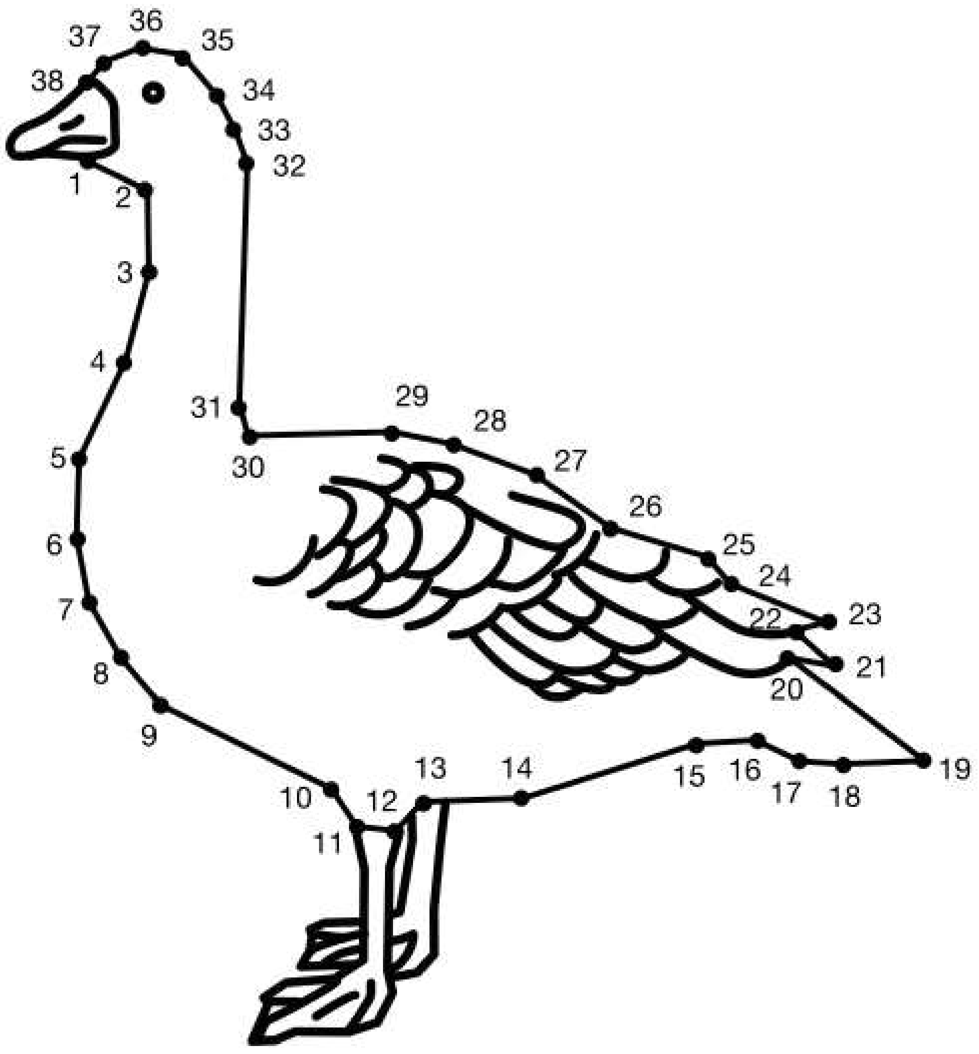
Published in Towards Data Science

Leonie Monigatti Follow
Aug 16, 2022 · 11 min read · · Listen

Save

99 Lessons on Data Analysis from Placing Top 5 in 5 Kaggle Analytics Challenges

(Grand)Masterclass: How to Approach a Kaggle Analytics Challenge



(Image by the author)

I have to agree with the critics: Kaggle Analytics challenges are only distantly related to writing real-world data analysis reports. But I like them because they can teach you a lot about the fundamentals of telling a story with data.

> Kaggle Analytics challenges are only distantly related to writing real-world data analysis reports. But [...] they can teach you a lot [...].

This article was originally going to be a proper article with paragraphs and images. But the first draft already had over 7,000 words, so a listicle is what you get instead.

In this article, **I share the “secret sauce” that got me in the top 5** [1] of five Kaggle Analytics challenges until now.

Don't misinterpret this list as a collection of rules. I haven't always followed these tips in my previous winning Kaggle Notebooks. Instead, this list is a **collection of lessons** I have learned along the way.

. . .

For this article, I will provide code snippets in Python with functions from the Pandas, Matplotlib, and Seaborn libraries. Feel free to use whatever programming language and complementing visualization libraries you like.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

. . .

Preparations

1. Understand the difference between **exploratory** and **explanatory** data analysis.

> *"When we do exploratory analysis, it's like hunting for pearls in oysters. We might have to open 100 oysters […] to find perhaps two pearls. When we're at the point of communicating our analysis to our audience, we really want to be in the explanatory space, meaning you have a specific thing you want to explain, a specific story you want to tell — probably about those two pearls." — [3]*

2. Read the problem statement — understand the problem statement.

3. Get an overview of the **whole dataset:**

- What is the file structure?
- How many files does the whole dataset have?
- What relationship do the files have?
- What are common key columns?

4. Get an overview of **each file** in the dataset with `df.info()`:

- How many columns are in a file?

- How many rows are in a file?
- What do the column names mean?
- What type of data do you have (e.g., numerical, categorical, time series, text, etc.)?

5. Read the dataset's description.

6. Check unique values with `df.nunique()` for plausibility and cardinality.

7. Get an overview of missing values.

```
# Display the percentage of missing values
df.isna().sum(axis = 0) / len(df) * 100

# Visualize the missing values
sns.heatmap(df.isna(),
            cbar = False,
            cmap = "binary")
```

8. You don't have to look at all the data if you have a large dataset— but **don't be lazy** with your selection.

9. Be prepared that not every plot you create is going to make it into the final report. Make a lot of plots anyways.

10. You only need **three types of plots for univariate analysis:** Histograms or boxplots for numerical features and bar charts (count plots) for categorical features.

```
# Explore numerical features
sns.histplot(data = df,
             x = num_feature)
sns.boxplot(data = df,
            y = num_feature)

# Explore categorical features
sns.countplot(data = df,
              x = cat_feature)
```

11. Document what you are doing in the code — you'll thank yourself later.

12. Data cleaning and feature engineering should happen *naturally* during exploratory data analysis (EDA).

13. Numerical features can be disguised as categorical features and vice versa.

14. Be on the lookout for NaN values disguised as implausible values (e.g. -1, 0, or 999). Sometimes, they will show themselves as suspicious peaks in otherwise unsuspicious looking distributions.

```
# Replace invalid values with NaN
invalid_value = 9999

df["feature"] = np.where((df["feature"] == invalid_value),
                          np.nan,
                          df["feature"])
```

15. Don't ignore outliers. You just might find something interesting.

16. Look at edge cases (top 5 and bottom 5).

17. Create new features by either **splitting a single feature into multiple features** or **combining multiple features into a new one**.

```
# Splitting features
df["main_cat"] = df[feature].map(lambda x: x.split('_')[0])
df["sub_cat"] = df[feature].map(lambda x: x.split('_')[1])

df[["city", "country"]] = df["address"].str.split(', ', expand=True)

# Combining features
df["combined"] = df[["feature_1", "feature_2"].agg('_'.join, axis=1)
df["ratio"] = df["feature_1"] / df["feature_2"]
```

18. Create count and length features from text features.

```
# Creating word count and character count features
df["char_cont"] = df["text"].map(lambda x: len(x))
df["word_count"] = df["text"].map(lambda x: len(x.split(' ')))
```

19. `datetime64[ns]` features contain a lot of new features:

```
# Convert to datetime data type
df["date"] = pd.to_datetime(df["date"],
                            format = '%Y-%m-%d')
```

```
# Creating datetimeindex features
df["year"] = pd.DatetimeIndex(df["date"]).year
df["month"] = pd.DatetimeIndex(df["date"]).month
df["day"] = pd.DatetimeIndex(df["date"]).dayofyear
df["weekday"] = pd.DatetimeIndex(df["date"]).weekday
# etc.
```

20. For coordinates the first number is always the latitude and the second is the longitude (but latitude corresponds to the y-axis and longitude to the x-axis when you plot the coordinates).

```
# Splitting coordinates into longitude and latitude
df["lat"] = df["coord"].map(lambda x: x.split(", ")[0]))
df["lon"] = df["coord"].map(lambda x: x.split(", ")[1]))
```

21. Extend your dataset with additional data. It demonstrates your creativity. You have three options to get additional data (in order of descending effort):

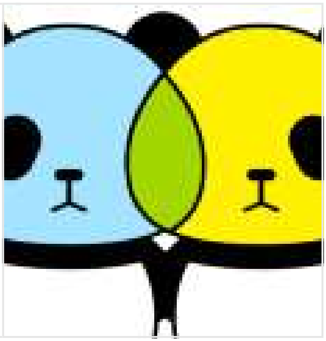
- Create your own dataset,
- find a dataset and import it to Kaggle,
- or use a dataset that is already available on Kaggle (I prefer this one).

22. Don't lose valuable data points when merging two DataFrames. Also, make sure that the **spelling** of the key column matches in both DataFrames. Double-check your work by checking the resulting DataFrame's length with `len(df)`.

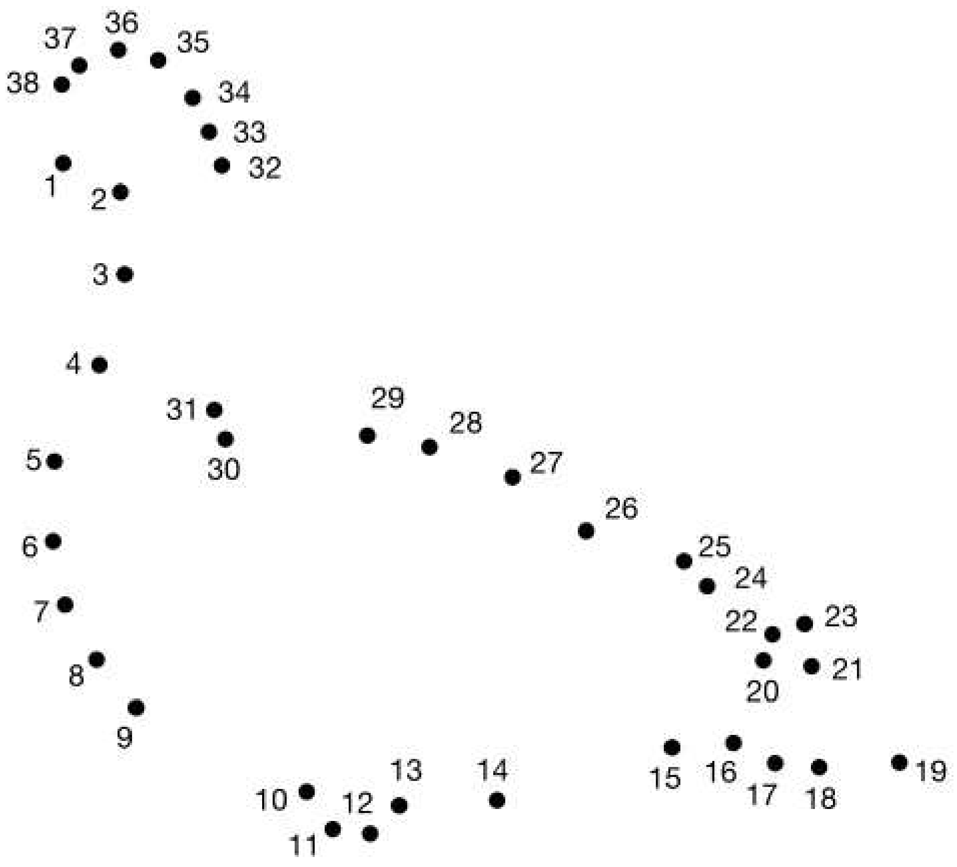
How to Merge Pandas DataFrames

How to Avoid Losing Valuable Data Points (incl. Cheat Sheet)

towardsdatascience.com



23. Review the evaluation criteria.



Before you start exploring data (Image by the author)

Exploratory Data Analysis

24. Accept that you (probably) can't look at all the relationships in the data. Looking at every possible combination will scale up *exponentially* with (n + (n over 2) + (n over 3)) if n is the number of features.

25. Start by plotting a few (random) relationsships — just to get comfortable with the data.

26. Don't waste time on creating *fancy* data visualizations during the EDA (I promise we'll get there later).

27. Domain knowledge is king. Do some research to get familiar with the topic.

28. Invest some time to think about what aspects you want to explore. Brainstorm a **line of questioning that's worth answering.** If you don't know where to start, start with the research questions suggested by the challenge host.

29. Get an overview of possible relationships to look at before you begin with the multivariate analysis (but keep in mind that both of the following methods will only consider the numerical features).

```
# Display pair plot
sns.pairplot(df)

# Display correlation matrix
sns.heatmap(df.corr(),
            annot = True,
            fmt = ".1f",
            cmap = "coolwarm",
            vmin = -1,
            vmax = 1)
```

30. Take notes of your findings in bullet point form after each plot.

31. You only need **four types of plots for bivariate analysis:**

- Scatterplots for the relationship between two numerical features

```
sns.scatterplot(data = df,
                x = "num_feature_1",
                y = "num_feature_2")
```

- Boxplots for the relationship between a categorical and a numerical feature

```
sns.boxplot(data = df,
            x = "cat_feature",
            y = "num_feature")
```

- Heatmaps or bar charts for the relationship between two categorical features

```
temp = pd.crosstab(index = df["cat_feature_1"],
                   columns = df["cat_feature_2"])

# Bar chart
temp.plot.bar()

# Heatmap
sns.heatmap(temp, annot = True)
```

32. The `groupby()` method is your friend for multivariate analysis.

```
# How many feature_1 per feature_2?
df.groupby("feature_2")["feature_1"].nunique()

# What is the average feature_1 for each feature_2?
df.groupby("feature_2")["feature_1"].mean()
```

33. Got time series data? Conduct a **trend analysis using line plots**.

```
sns.lineplot(data = df,
             x = "time_feature",
             y = "num_feature")
```

34. You can conduct **multivariate analysis without learning any new plots:**

- Scatterplot with hue or size for relationship between three numerical features

```
sns.scatterplot(data = df,
                x = "num_feature_1",
                y = "num_feature_2",
                hue = "num_feature_3")

sns.scatterplot(data = df,
                x = "num_feature_1",
                y = "num_feature_2",
                size = "num_feature_3")
```

- Scatterplot with hue or style for relationship between two numerical features and a categorical feature

```
sns.scatterplot(data = df,
                x = "num_feature_1",
                y = "num_feature_2",
                style = "cat_feature")

sns.scatterplot(data = df,
                x = "num_feature_1",
                y = "num_feature_2",
                hue = "cat_feature")
```

- Grouped bar charts or boxplots for relationship between two categorical features and a numerical feature

```
sns.barplot(data = df,
            x = "cat_feature_1",
            y = "num_feature",
            hue = "cat_feature_2")

sns.boxplot(data = df,
            x = "cat_feature_1",
            y = "num_feature",
            hue = "cat_feature_2")
```

- Stacked grouped bar charts for relationship between three categorical features

35. Always doubt your findings. Take some time to sanity-check and double-check your plots for data fallacies like **Simpson's paradox** [2].

> *"A phenomenon in which a trend appears in different groups of data but disappears or reverses when the groups are combined." — [2]*

36. Read, read, read. Extend your EDA with research.

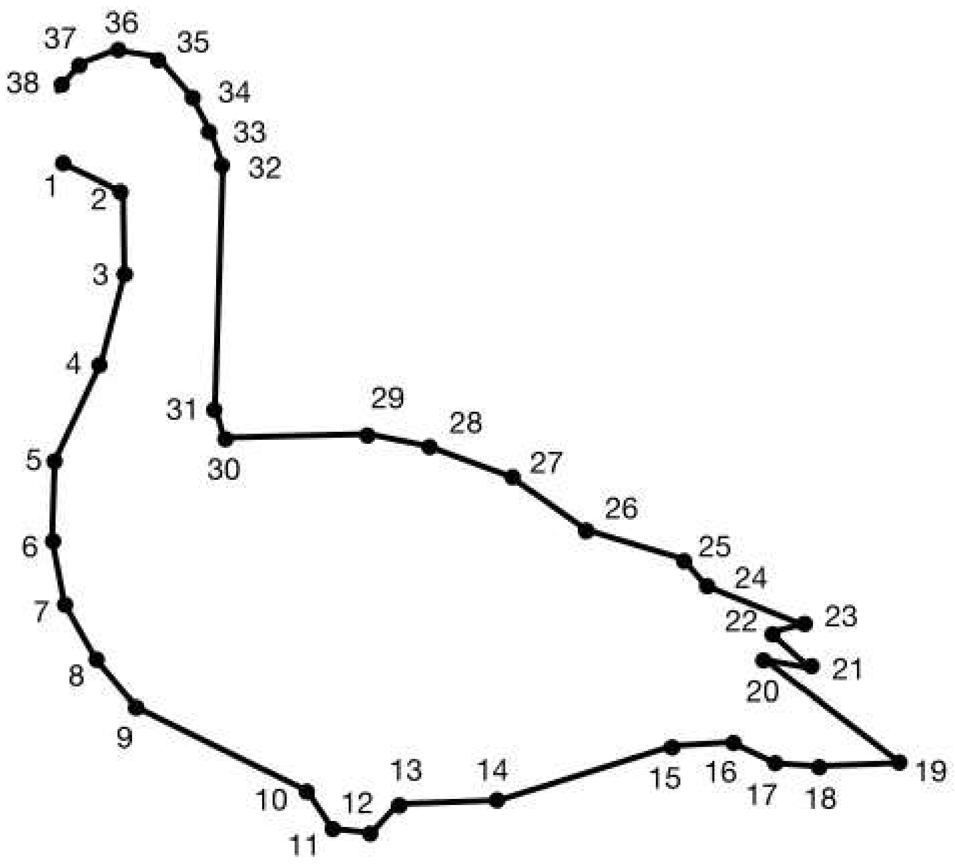
37. **Modeling can be useful for data analysis**. E.g., you can build a linear regression model to predict the value for the next year, you can apply clustering to create a new feature, or you can use feature importances to gain insights.

38. Take a minute to make sure you are interpreting the plots correctly.

39. Make sure you have a sufficient amount of plots (or insights to build a story around them).

40. Refactor your code. It helps you detect errors and makes your code more accessible and reproducible for your audience.

41. It's OK if the variety of data visualizations is underwhelming at this stage.



Connecting the dots — The insights after EDA (Image by the author)

Explanatory Data Analysis

42. Storytelling is more important than data visualizations — Trust me, I won my first prize with mostly bar charts.

43. Pick a clear topic and build a story around it. Make sure the question you are answering is useful to the competition host.

44. The entry for a Kaggle Analytics challenge is **not a collection of plots**. It needs an introduction, a body, and a conclusion.

45. Create an outline based on your findings.

46. Don't hesitate to discard most of your plots so far.

47. Tell your audience about the dataset. What data are you working with? How many data points are there? Did you add external data?

48. Explain what you did.

49. Show what you didn't see. Did you have an interesting hypothesis but the data didn't support it? Show that and discuss it.

50. Don't include a point just because you think the data visualization is cool. If the finding is not relevant to the overall story you are telling, cut it.

51. Write a first draft.

52. Now is the time for the fancy plots (see, I kept my promise).

53. You'll make better data visualizations if you know **what you want to show**.

54. Double-check if the **metric** you are using is suitable for what you want to show. E.g., to measure a platform's popularity would you use the total number of accounts or the average number of daily active users?

55. Avoid vanity metrics.

56. Get some inspiration from the pros — but understand that boring data visualizations (aka bar charts) are usually the most effective.

57. Decide **which data visualization to use based on what** (distribution, relationship, composition, comparison) you want to show.

58. Your best bets are these **six types of plots** and their variations: bar charts, heatmaps, histograms, scatterplots, boxplots, and line plots.

59. Remember that single numbers and tables can be data visualizations, too.

Essential Techniques to Style Pandas DataFrames

How to Effectively Communicate Data with Tables (including Cheat Sheet)

towardsdatascience.com

60. Please don't use pie charts. Also, please don't use donut charts. **If your plot is named after a dessert, don't use it** (and when you do, know that you shouldn't).

61. Replace word clouds with bar charts.

62. Please, please, please don’t use 3D effects.

63. Use choropleth maps intentionally (and not just because you have geographical data).

64. Define a color palette. Have at least one highlight color and one contrast color.

```
# Set color theme
highlight_color = "#1ebeff"
contrast_color = '#fae042'

from matplotlib.colors import LinearSegmentedColormap
custom_palette = LinearSegmentedColormap.from_list(" ",  [highlight_color,
'#ffffff', contrast_color])
plt.cm.register_cmap("custom_palette", custom_palette)
```

65. Make sure to use the right color palette for your purpose:

- Sequential for ordered values (e.g. 1, 2, 3, …,)
- Diverging for opposing values with a neutral mid-value (e.g. -1, 0, 1)
- Qualitative for categorical values

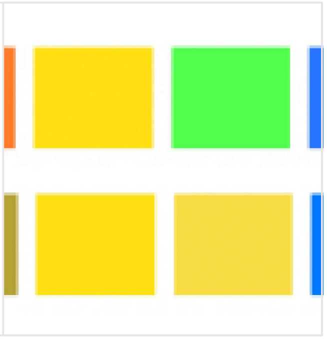
66. Grey is your friend when you need to remove the focus from context information.

67. Make sure your colors are colorblind- and photocopy-safe.

Why Your Data Visualizations Should Be Colorblind-Friendly

Especially if You Are Trying to Convince Men

towardsdatascience.com



68. Visualize like an adult: Add a title and labels to your plot.

69. Add a legend if suitable.

70. Set an appropriate font size.

```
plt.rcParams.update({"font.size": 14})
```

71. Keep it simple. Remove any distractions and redundancies from the plot.

72. Don't start the quantitative axis of your bar charts anywhere other than 0.

73. Don't compare two plots with different axis ranges.

74. Don't mislead your audience with your data visualizations, e.g. by ignoring 72. and 73.

75. Add annotations directly to your plot.

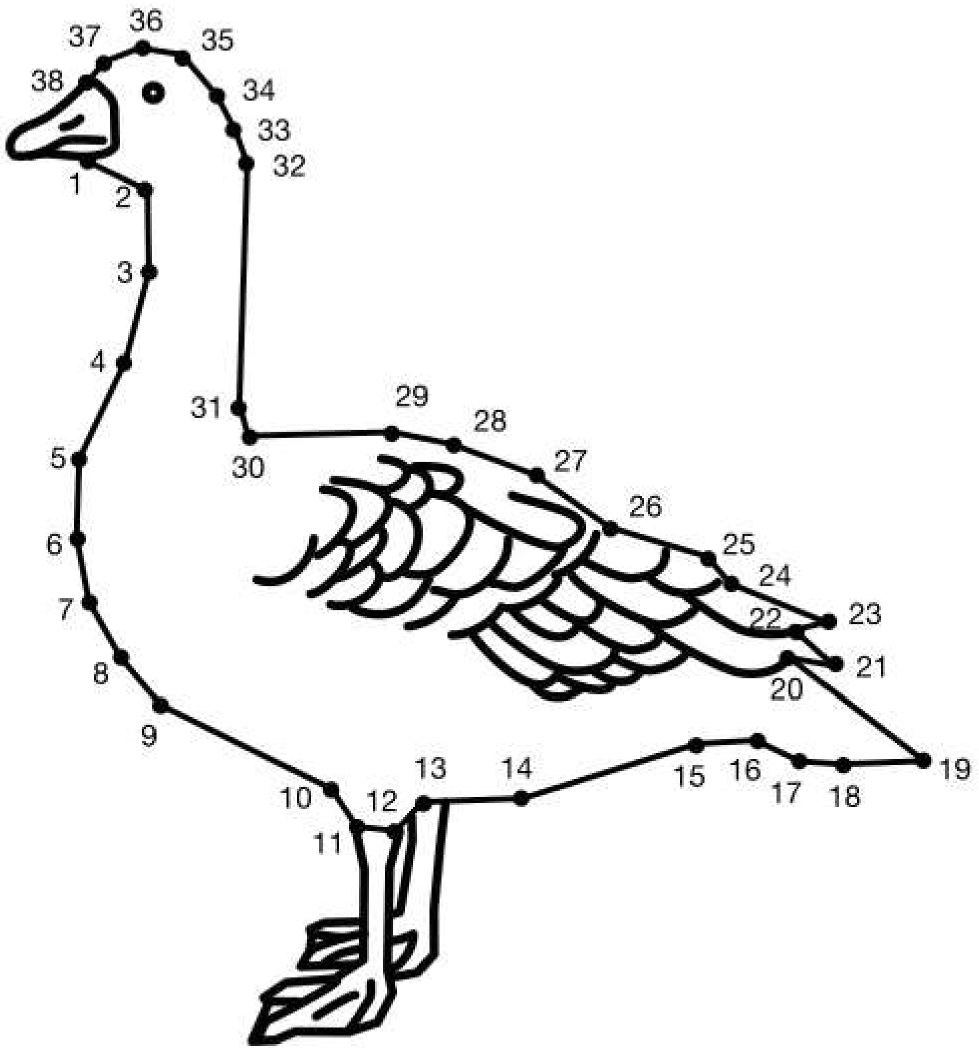
```
ax.annotate("Annotation text",
            xy = (x_pos, y_pos))
```

76. If you are working with ordinal categorical data, make sure to also order the bars in the bar charts to represent the ordinal feature correctly.

77. Exploit preattentive processing.

78. Think "mobile first" (because a good portion of your audience is going to read your report on their phone).

79. Look at each data visualization. Does it convey the message without context? If not, revise.



Effectively communicating your insights — The difference between exploratory and explanatory data analysis (Image by the author)

Finishing Touches

80. Write a second draft.

81. A data visualization should always be accompanied by some text. Turn the bullet points into text. Plots alone won't cut it.

82. Revise and edit your second draft.

83. Use an attention-grabbing image at the beginning of your report. (You can find great photographs on Unsplash but make sure to credit your source.)

84. Nobody wants to see your code — hide it.

85. Hide console warnings as well.

```
import warnings # Supress warnings
warnings.filterwarnings("ignore")
```

86. Review your cell outputs. If it is not a data visualization, then it must help tell your story. Otherwise, hide it as well.

87. Lead with the insights — Nobody is going to read every word of your report. Add a summary in bullet point form at the beginning.

88. Use bolding to highlight important points in your text (because nobody is going to read every word of your report).

89. Make sure to write a good conclusion. Did I mention that nobody is going to read every word of your report?

90. Use a spellchecker. I like Grammarly.

91. Make sure you check all the boxes for the challenges evaluation criteria.

92. Phew, emojis. Love 'em or hate 'em — just promise me to not overdo them, alright?

93. Keep in mind that your audience might not be data scientists. Is your analysis accessible?

94. Cite your sources.

95. Invest some time to come up with a good title.

96. Have a friend review your report and/or read it out loud.

97. Let the report rest for a few days.

98. Give your report a final review.

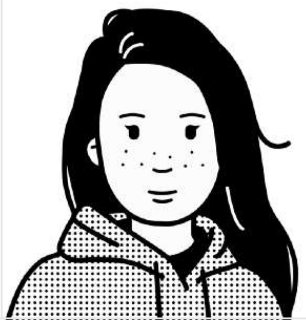
99. Let go of perfectionism and submit.

. . .

Enjoyed This Story?

If you'd like to get my new stories directly to your inbox, make sure to subscribe!

Become a Medium member to read more stories from me and other writers. You can support me by using my referral link when you sign up. I'll receive a commission at no extra cost to you.



Find me on LinkedIn and Kaggle!

References

[1] Below I have listed my portfolio of prize-winning Kaggle Notebooks for your reference:

- 2022: What Happens After A Mention in "Hidden Gems"? for Hidden Gems Notebooks Competition (1st place)
- 2021: "Head in the Clouds" for 2021 Kaggle Machine Learning & Data Science Survey (1 out of 5 winners)
- 2020: Maslow Before Bloom for LearnPlatform COVID-19 Impact on Digital Learning (1 out of 5 winners)
- 2020: Impact Potential Analysis of Water-Use Efficiency for CDP — Unlocking Climate Solutions (3rd place)
- 2019: Japan: Country of the Rising Women for 2019 Kaggle Machine Learning & Data Science Survey (4th place)

[2] geckoboard, "Data fallacies". geckoboard.com. https://www.geckoboard.com/best-practice/statistical-fallacies/ (accessed August 14, 2022)

[3] Nussbaumer Knaflic, C. (2015). *Storytelling with Data*. John Wiley & Sons.

Data Science · Data Analysis · Data Visualization · Programming · Deep Dives