Python for Data Analytics

Module 3: Data Visualization

DeepLearning.AI

Data Visualization

Module 3 introduction

DeepLearning.AI

Module 3 outline

Peer to peer loans

- **Plotting with matplotlib**
  - Column charts
  - Scatter plots
  - Chained methods
- **Visual appeal with Seaborn**
  - Styles
  - Color palettes
  - Box plots
  - Histograms
- **Multiple charts**
  - Charts on charts
  - Side-by-side
  - Looping techniques

Data Visualization

Plotting with matplotlib

DeepLearning.AI

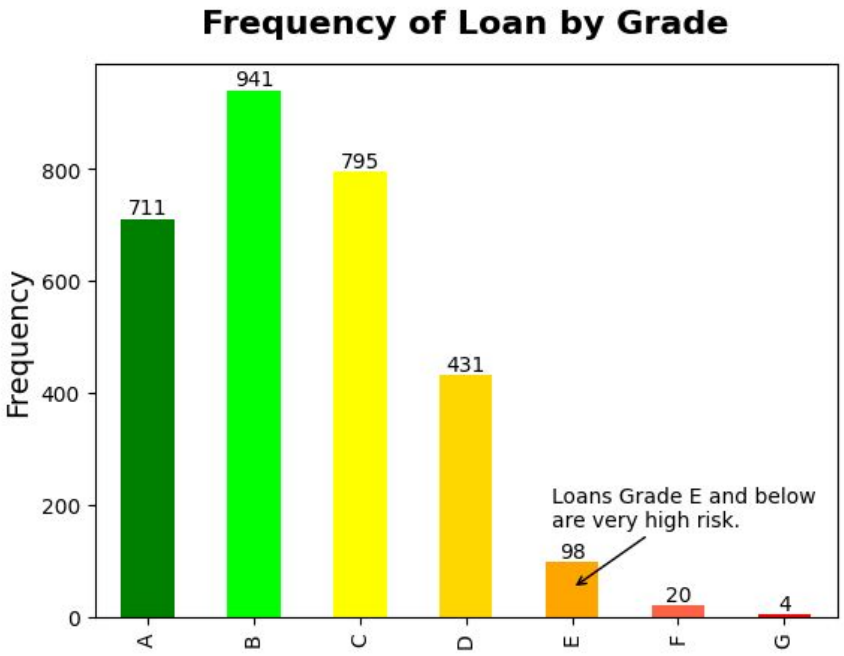
Matplotlib

- A visualization module

```
import matplotlib
```

- Customization features:
  - Titles
  - Annotations
  - Colors
  - Axis limits
  - Label formatting

- Hundreds of thousands of lines of code already written for you



Figure

- Can customize figure directly
  - Control the size of the canvas
  - Set options like background color

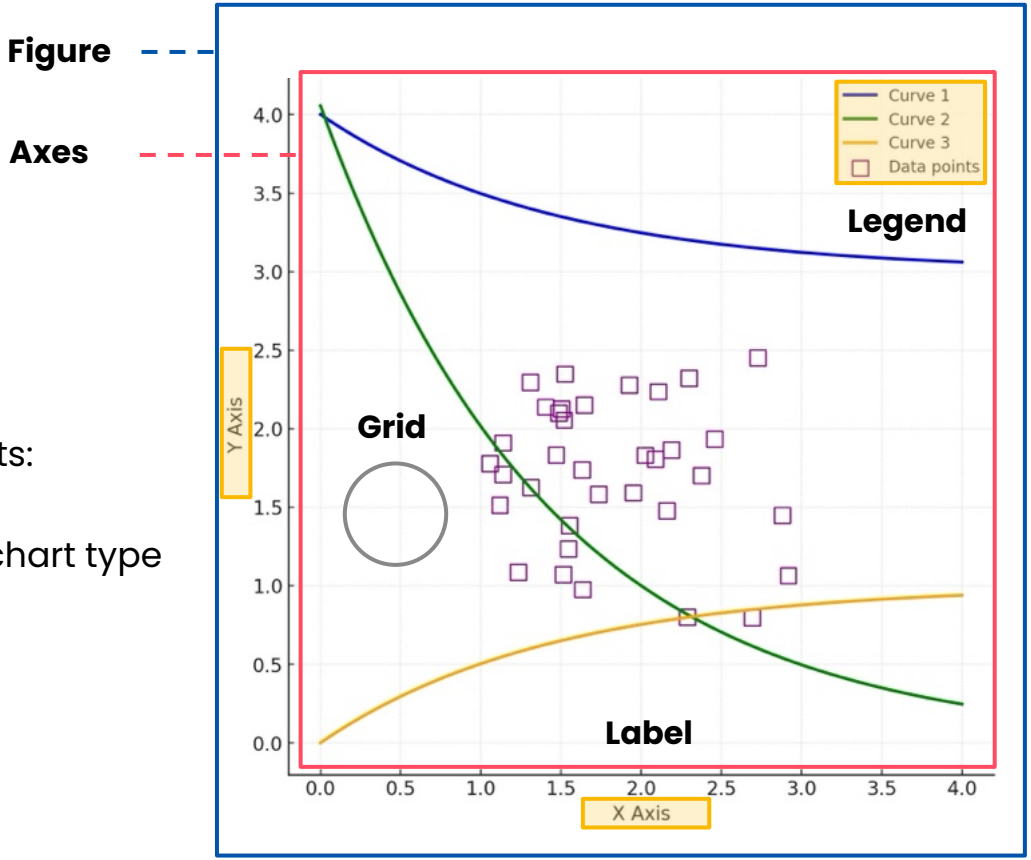
Axes (plots)

- Create using different functions:
  - `.plot()`
  - `.scatter()`
  - `.hist()`

  Have named arguments:
  - Data for plotting
  - Data ink for that chart type

Additional chart elements

- `.title()`
- `.xlabel()`
- `.ylabel()`
- `.annotate()`
- `.legend()`



Scenario

You
Data Analyst

🏆 **Goal**: Develop a state-of-the-art risk management strategy for providing loans to different communities across the United States

📊 **Dataset**: Loans from Lending Tree, a peer-to-peer lending platform

🎯 **Task**: Conducting exploratory data analysis to better understand the characteristics of loans with different levels of risk

- ⬜ Develop a report of findings to share with the bank
- ⬜ Develop insightful visualizations to help client understand different risk profiles

Recap: Matplotlib

1. Select & order data using Pandas

```
df["grade"].value_counts().sort_index()
```

2. Stack commands to enhance visualizations

```
import matplotlib.pyplot as plt
sorted_grades.plot(kind="bar")
plt.xlabel("")
plt.ylabel("Frequency")
plt.title("Frequency of Loan by Grade")
```

3. Use `plt.show()`:
   - To clean up the output of each code cell
   - Multiple times to display multiple plots

Data Visualization

Colors, grids, & saving plots

DeepLearning.AI

Recap: Colors, grids, & saving plots

- To specify color for chart:

"8af133"

```
sorted_grades.plot(kind="bar", color="purple")
```

- To use list to give each bar its own color:

```
colors = ["Green", "Lime", ... , "Red"]
sorted_grades.plot(kind="bar", color=colors)
```

- To add grid lines:

```
plt.grid(axis="y", color="black", alpha=0.7, linestyle="--")
```

Specify x or y axes, or both **Color** **Line style**

- To save an image

```
plt.savefig("loan_column_chart.png")
```

File name

Any common image format will work:

- .jpeg
- .svg
- .pdf

Data Visualization

Text & annotations

DeepLearning.AI

- **To adjust size and style of the font:**

```
plt.title("Frequency of Loan by Grade", fontsize=16, fontweight="bold", pad=15)
plt.xlabel("Frequency", fontsize=14)
```

- **Move annotation text:**

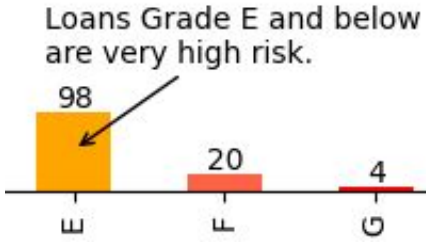
```
plt.annotate(text="Loans Grade E and below \nare very high risk.",
             xy=(4, 50),
             xytext=(-10, 30),
             textcoords="offset points",...)
```

Position for the text

How to move relative to point

Didn't set absolute value for text location

- **Used LLM to:**
  - Create arrow
  - Label each bar with its frequency



Data Visualization

Ticks & spines

DeepLearning.AI

Recap: Ticks & spines

- To rotate the x axis labels

```
plt.xticks(rotation=0)
```

- To save the result of a plot method into a variable:

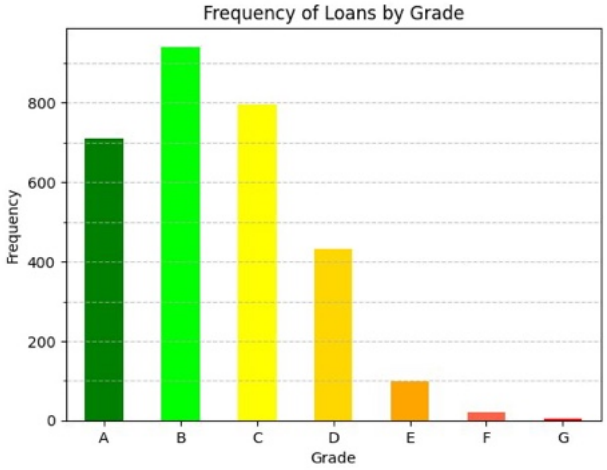
```
ax = sorted_grades.plot(kind="bar", color=colors)
```

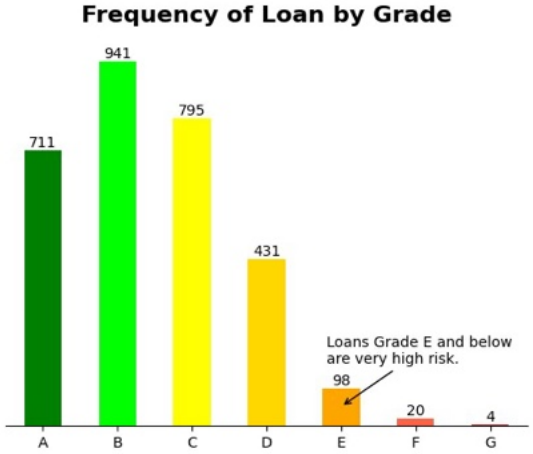
- To add more ticks to plot:

```
ax.yaxis.set_minor_locator(AutoMinorLocator(2))
```

- To remove the spines from plot:

```
ax.spines["left"].set_visible(False)
```



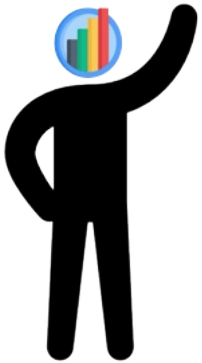


Data Visualization

Grouped column charts

DeepLearning.AI

Scenario



You
Data Analyst

🏆 **Goal**: Identify key characteristics of loans from states with highest average loan amount

- District of Columbia
- Alaska
- Hawaii

🎯 **Task**: Plot the loan amount by grade across these three states using a grouped bar chart

1. **Set up your data:**
   - Filter data to only include top three states
   - Group by state and grade
   - Select loan amount column and calculate mean for each group
   - Create a grouped column chart showing the mean value of loans of each grade

Alaska A

Alaska B

Hawaii A

⋮

Recap: Grouped column charts

1. Selected the rows of interest

```
# names of top three states for loan amount
states = ["DC", "AK", "HI"]
filtered_df = df[df["state"].isin(states)]
```

2. Grouped by two features

```
grouped_df = filtered_df.groupby(["state", "grade"])
```

3. Calculated mean of loan amount for each state and grade combination

```
grouped_loan_amount = grouped_df["loan_amount"].mean()
```

MultiIndex

state	grade	
AK	A	25750.000000
	B	30833.333333
	C	12500.000000
	D	11100.000000
DC	A	40000.000000
	B	12500.000000
	C	25900.000000
HI	A	1200.000000
	B	17733.333333
	D	10000.000000
	E	31666.666667
	F	28000.000000

Name: loan_amount, dtype: float64

✅ A lot of flexibility to create unique rows

⛔ To plot, you'll need to use `.unstack()`

.unstack()

state	grade	
AK	A	25750.000000
	B	30833.333333
	C	12500.000000
	D	11100.000000
DC	A	40000.000000
	B	12500.000000
	C	25900.000000
HI	A	1200.000000
	B	17733.333333
	D	10000.000000
	E	31666.666667
	F	28000.000000

	A	B	C	D	E	F
AK	25750.0	30833.3	12500.0	11100.0	NaN	NaN
DC	40000.0	12500.0	25900.0	NaN	NaN	NaN
HI	1200.0	17733.3	NaN	10000.0	31666.6	28000.0

```
grouped_loan_amount.unstack().plot(kind = "bar")
```

Columns are grouped by **index** automatically

Data Visualization

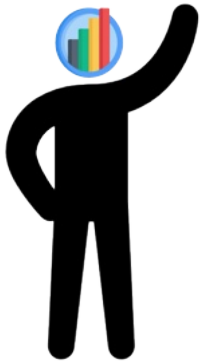
Stacked column charts

DeepLearning.AI

Scenario

🏆 **Goal**: Understand whether the composition of homeownership changes based on the risk profile of the loan

🎯 **Task**: Highlight the proportion of renters for each group



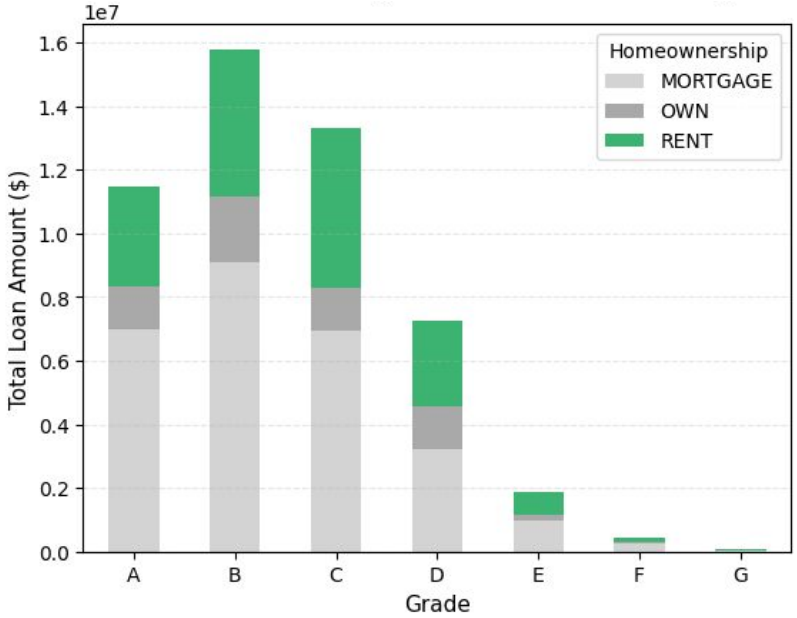
You
Data Analyst

Exploring total loan by:

- Grade
- Homeownership status

Perform similar steps:

- Grouping
- Aggregating
- Unstacking before plotting



Recap: Stacked column chart

- Create a **stacked column chart** rather than grouped one

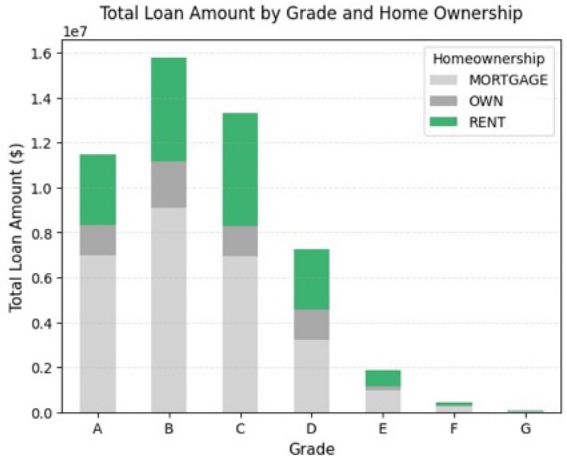
```
grouped_df.plot(kind = "bar", stacked = True,
                color=["lightgray", "darkgray", "mediumseagreen"])
```

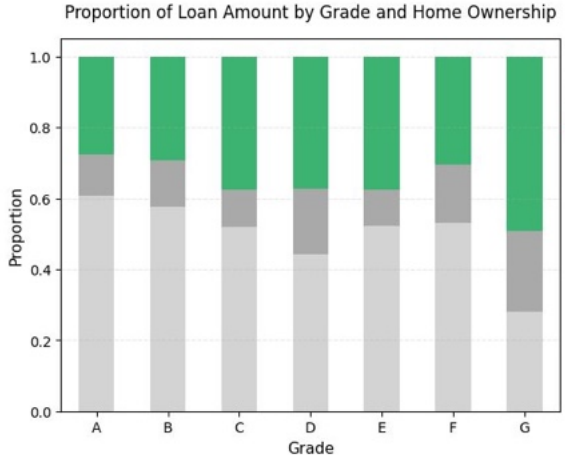
- Compare **within** a category rather than **across** them
- Worked with LLM to develop 100% stacked bar chart:

```
proportion_df = grouped_df.div(grouped_df.sum(axis=1), axis=0)

proportion_df.plot(kind = "bar", stacked = True,
                   color=["lightgray", "darkgray", "mediumseagreen"])
```

- Easier comparison across the different categories



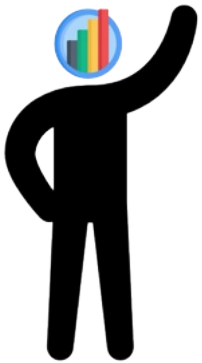


Data Visualization

Scatter plots

DeepLearning.AI

Scenario



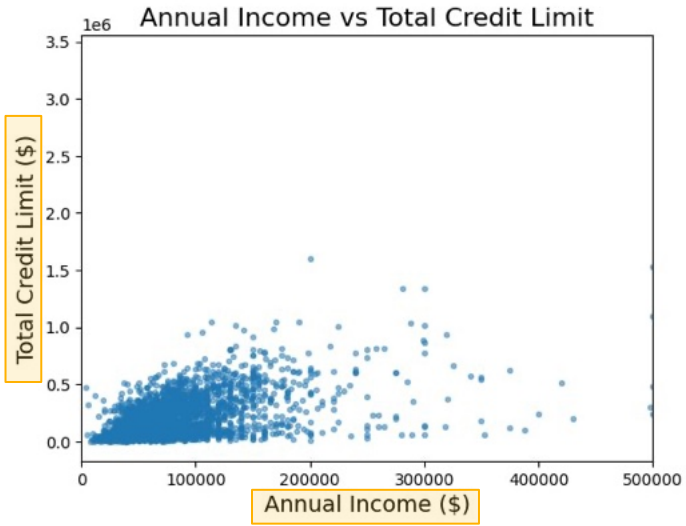
You
Data Analyst

💡 **Findings**: Annual income is a moderate predictor of total credit limit

- Pearson correlation coefficient of 0.55

🎯 **Tasks**:

- Graph how income distribution impacts amount of credit they should offer
- Visualize credit limits of incomes in top 5%



Recap: Scatter plots

Option	Style
"^"	▲
"."	●

- To create a scatter plot:

Columns

```
plt.scatter(df["annual_income"], df["total_credit_limit"], alpha=0.5, marker=".", color="darkgreen")
```

x-axis | Transparency | Style | Color of marker

- To set x axis limit:

```
plt.xlim(0, 500000)
```

```
plt.grid(alpha=0)
```

- To set y axis limit:

```
plt.ylim(...)
```

- To draw vertical line:

```
plt.axvline(x=top_5_percent_income, color="black", linestyle="--")
```

- To draw horizontal line:

```
plt.axhline(...)
```

Data Visualization

Method chaining

DeepLearning.AI

Method chaining

- Process of linking several operations in a row
- Each operation depends on the previous

🔗 **Analogy**: Connecting operations like links in a chain

$$(((5 + 3) \times 2) - 4) \div 2$$

$$((8 \times 2) - 4) \div 2$$

$$(16 - 4) \div 2$$

$$12 \div 2$$

$$6$$

Sean Barnes

```
df.groupby(["grade", "homeownership"])["loan_amount"].sum().unstack().plot(kind="bar", stacked=True)
```

DataFrame → **Group by grade & homeownership** → **Select `loan_amount` column** → **Sum loan amounts for each group** → **Get data into rows and columns** → **Create stacked bar chart**

Groupby Object

amount column from grouped data

```
grade  homeownership
A      MORGAGE          6972200
       OWN              1343400
       RENT             3176775
B      MORGAGE          9081100
       OWN              2056225
       RENT             4655775
C      MORGAGE          6925450
       OWN              1383300
       RENT             5005850
D      MORGAGE          3216450
       OWN              1329075
       RENT             2718175
E      MORGAGE           973275
       OWN               190000
       RENT              700025
F      MORGAGE           237300
       OWN                72900
       RENT              136600
G      MORGAGE            25000
       OWN                20300
       RENT               43750
Name: loan_amount, dtype: int64
```

homeownership	MORGAGE	OWN	RENT
grade			
A	6972200	1343400	3176775
B	9081100	2056225	4655775
C	6925450	1383300	5005850
D	3216450	1329075	2718175
E	973275	190000	700025
F	237300	72900	136600
G	25000	20300	43750

```
grouped_data = df.groupby(["grade", "homeownership"])
grouped_data_loan_amount = grouped_data["loan_amount"]
grouped_sum_of_loans = grouped_data_loan_amount.sum()
unstacked_sum_of_loans = grouped_sum_of_loans.unstack()
unstacked_sum_of_loans.plot(kind="bar", stacked=True)
```

Using variables for each step:

- Useful to save intermediate steps for later
- More flexible, but takes longer

Method chaining:

- Get to plot as quickly as possible
- Less flexible
- If you wanted to stop at an intermediate step, you wouldn't be able to

Data Visualization

Plotting with Seaborn

DeepLearning.AI

Seaborn

- A **visualization tool**

```
import seaborn
```

- Works well with matplotlib
- Main strengths:
  - Improved visual appeal
  - Reduced need to manipulate the data
  - Additional plot types

Plotting with Seaborn

At this point in analysis, you're looking to:

- ⬜ Level up visual appeal of visualizations
- ⬜ Include many plots of different features and relationships in data

Seaborn can help in both:

- ✅ Improving visual appeal
- ✅ Creating many charts quickly
  - Including distributional charts

Recap: Plotting with Seaborn

- Gives additional functions for plotting:
  - `sns.barplot()`
  - `sns.lineplot()`
  - `sns.regplot()`

- Automatically summarizes your data:

Default

```
sns.boxplot(selected_stocks, estimator = np.mean)
```

- Set the estimation to:
  - `np.max`
  - `np.size`
  - `np.std`

Data Visualization

Themes & palettes

DeepLearning.AI

Recap

- Use `sns.set_theme()` to change default styling

```
sns.set_theme(style = "white")
```

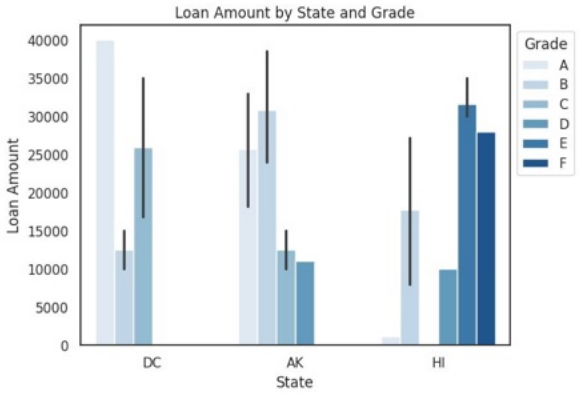
Default matplotlib look

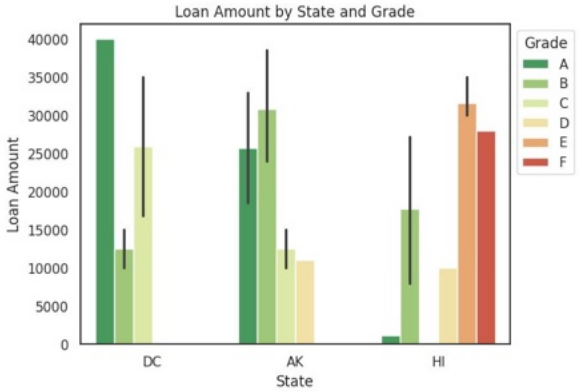
- Gives more control of visual style of plots
  - Used `palette` named argument

```
sns.barplot(...palette = "Blues")
```

```
sns.barplot(...palette ="RdYlGn_r")
```

  - Avoid manually selecting colors



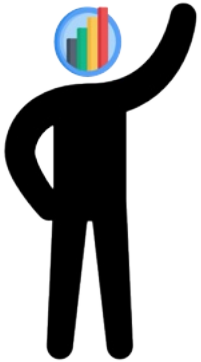


Data Visualization

Box plots

DeepLearning.AI

Scenario



You
Data Analyst

🏆 **Goal**: Characterizing different features present in the dataset

Features of **each person**:

- Occupation
- Loan history

Features of **individual loans**:

- Loan amount
- Interest rate

Recap

Creates box plot: `y = "interest_rate"`

or

```
sns.boxplot(df, x = "interest_rate")
```

Set both to segment by another variable:

```
sns.boxplot(df, x = "grade", y = "interest_rate")
```

Remove some axes from the plot:

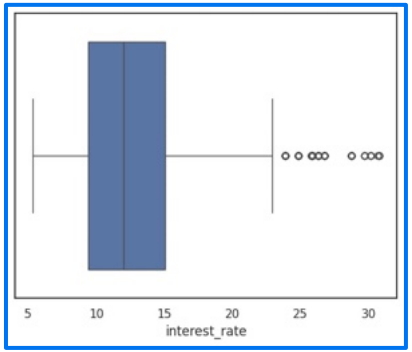
```
sns.despine(left = True, bottom = True)
```

Increase the figure size:

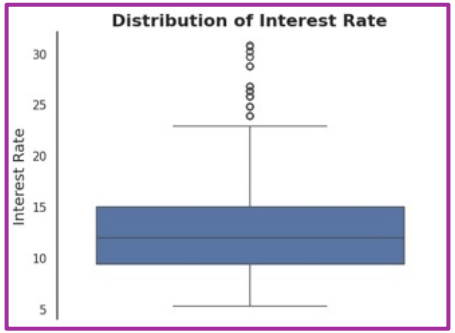
```
plt.figure(figsize = (8, 6))
```

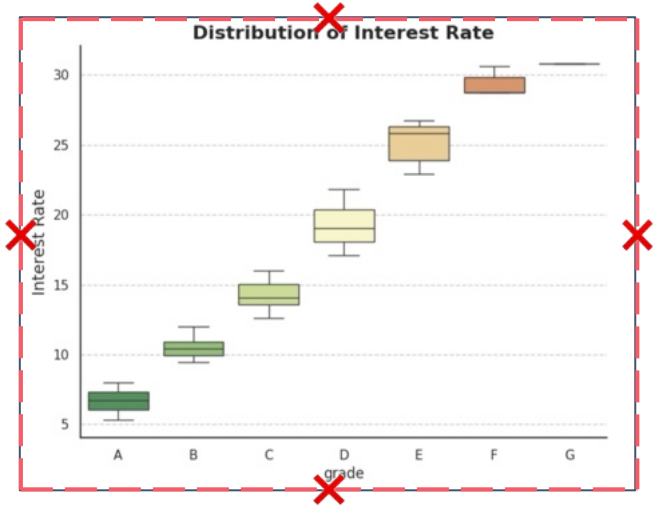
Width & height
in inches

Horizontal ↔️



Vertical ↕️



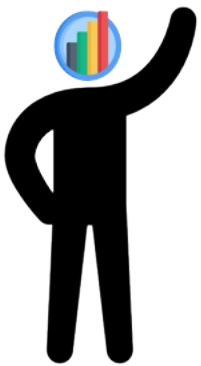


Data Visualization

Histograms

DeepLearning.AI

Scenario



You
Data Analyst

🏆 **Goal**: Characterizing different features present in the dataset

Features of **each person**:

- Occupation
- Loan history

Features of **individual loans**:

- Loan amount
- Interest rate

🎯 **Task**: Visualize using histogram to explain its unique properties

- This visualization will help:
  - Identify and price common loan products

Recap

To graph histogram

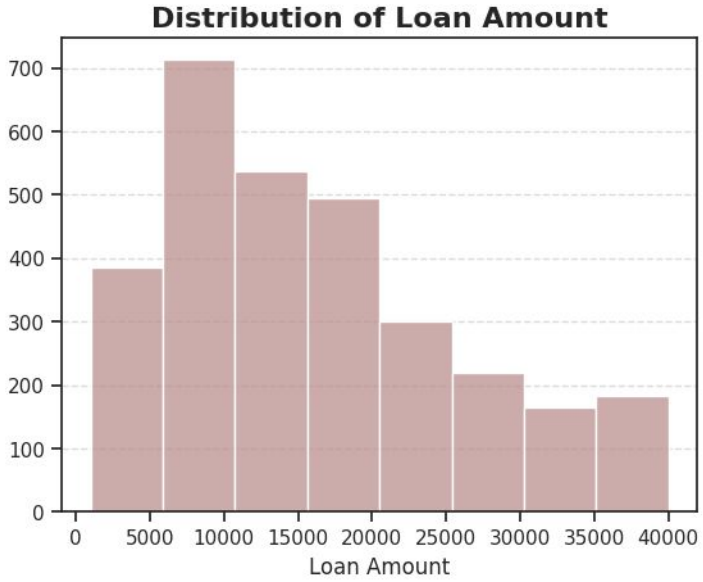
Vertical bars: `y = "interest_rate"` or Horizontal bars: `x = "grade"`

Number of bins: `bins = 10` or Width of bins: `binwidth = 50`

Density curve estimate: `kde = True`

```
sns.histplot( df, x = "grade" , binwidth = 50 , kde = True )
```

```
sns.set_style("ticks")
```



Data Visualization

Other charts

DeepLearning.AI

Upcoming Plots

For each plot, you'll see:

- Overview of the code
- The output

⛔ Try not to focus too much on code

✅ Do keep in mind:

- Commonalities with other visualizations you've created
- Output you expect from each visualization

Data Visualization

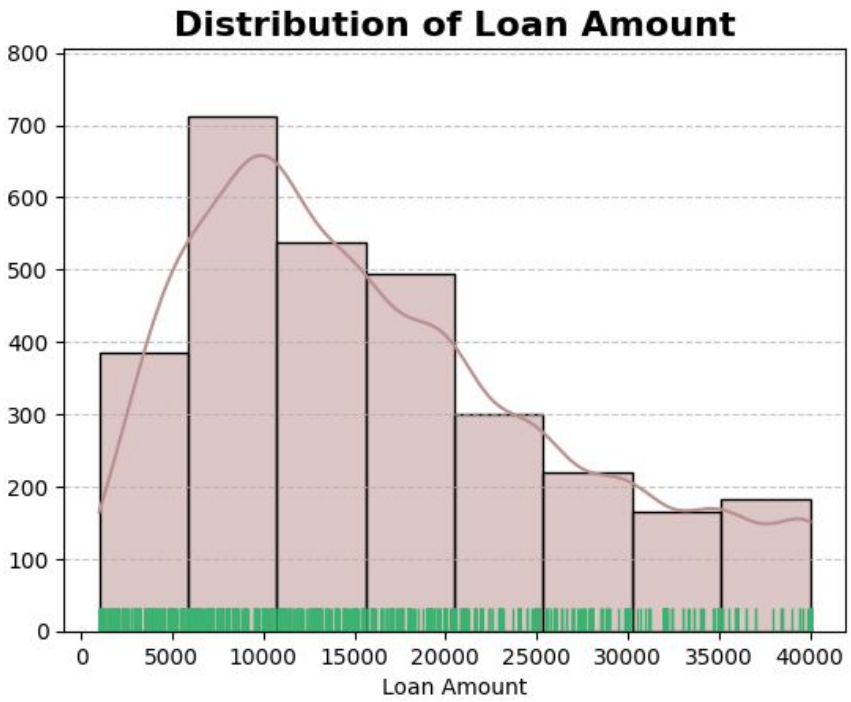
Combining charts

DeepLearning.AI

Matplotlib subplots

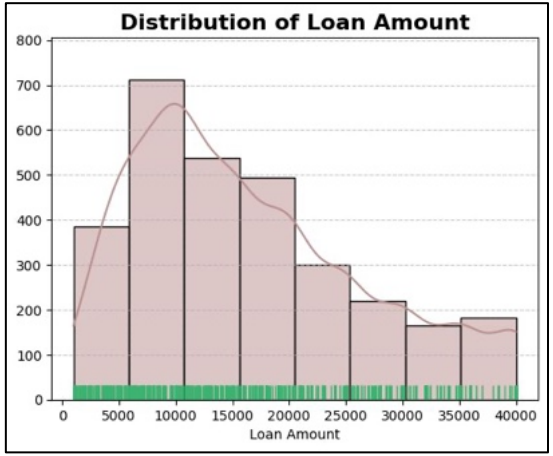
You can:

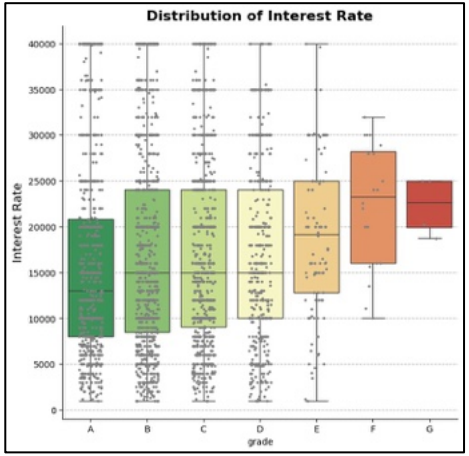
- Stack commands from seaborn and matplotlib to create a chart
- Stack plots on top of each other
  - Use technique to:
    - Stack complementary plots
    - Plot multiple distributions together on the same axes

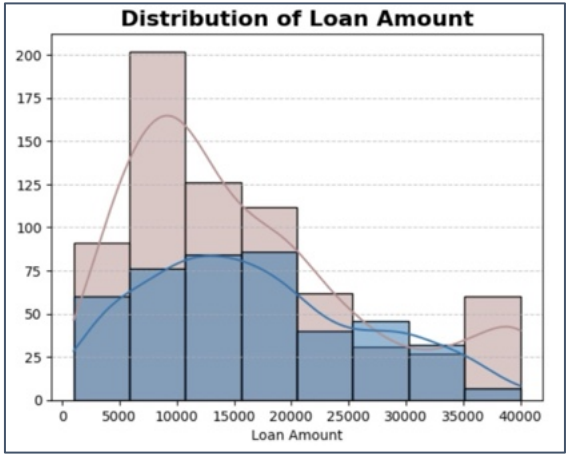


Recap: Subplots

1. Create a rugplot on histogram
   - Disaggregated plot of individual values
2. Combine a stripplot with a boxplot
3. Overlay histograms on the same chart to compare distributions
   - Accomplished by creating two plots in the same figure







Data Visualization

Matplotlib subplots

DeepLearning.AI

Scenario

🏆 **Goal**: Explore number of open credit lines customers have

- Credit cards
- Home equity
- Business line of credit

🎯 **Task**: Segmenting paid interest based on loan grade and open lines of credit

- A lot of possible values for line of credit
- Need to create a lot of graphs

matplotlib

Create a grid of empty subplots

+

Seaborn

Fill up with appealing plots

Recap: Matplotlib subplots

- Create the figure

Canvas Width, Height

```
plt.figure(figsize = (5,5))
```

- Use the matplolib .subplot()

Rows Col. Plot

```
plt.subplot( 2,   3,   4 )
```

1	2	3
4	5	6

- Create a plot

```
sns.barplot(filtered_df, hue="grade", y="paid_interest", palette="RdYlGn_r")
```

Data Visualization

Looping with subplots

DeepLearning.AI

Recap: Looping with subplots

Use **for** loop to iterate through subplots

- Loop through a range:

  Starting: 1

  Ending: 1 after the last number

- Use to help improve code's efficiency and readability

```
plt.figure(figsize=(15,15))
for i in range(1,10):
    plt.subplot(3, 3, i)
    filtered_df = df[df["open_credit_lines"] == i]
    sns.barplot( filtered_df,
                 hue="grade",
                 y="paid_interest",
                 palette="RdYlGn_r")
plt.title(i)
plt.savefig("9graphs.png")
```

Data Visualization

Seaborn pairplot

DeepLearning.AI

You're almost done with EDA! 🎉

- **Next steps**: Create an appendix with many distributional plots
- **Why**: For clients interested in looking for their own insights
- **How**: Use Seaborn's `pairplot` tool to create many plots quickly

Recap: Seaborn pairplot

How to use `sns.pairplot()` to:

- Create scatterplots between features and histogram for each feature individually

```
sns.pairplot(df[["loan_amount", "annual_income", "interest_rate", "paid_interest"]])
```

- Add to the title of the entire figure

```
plt.suptitle("Pairplot of Loan Amount, Annual Income, Interest Rate, and Paid Interest")
```