Project Title

Name I, First Name I Name II, First Name II
Name III, First Name III

2025-10-15

This is the abstract of the report. It should be a short summary of the project, the data, the analysis and the results. It should be concise and to the point. It should not be longer than 250 words.

❗ Interpretation > Visualization

Plots alone won't earn you a good grade. What matters most is **interpreting your findings**.
For every result you present:

- **Explain** what the data shows
- **Interpret** what it means for your research questions
- **Discuss** implications and connect to domain knowledge

Quality of insight > Quantity of plots. Your instructors can read plots—show them you *understand* what the data reveals.

Quarto Guide (Remove After)

💡 Two Report Writing Options

Option 1 - Modular (Recommended for Teams):

- Each section in a separate `.qmd` file in `report/sections/`
- Files prefixed with `_` (e.g., `_introduction.qmd`) are auto-included
- Better for collaboration (fewer merge conflicts)
- Render `report.qmd` to build the complete report

Option 2 - Single File:

- Write everything in `report.qmd`
- Simpler but harder to collaborate
- Delete `report/sections/` folder if using this approach

See Quarto includes documentation for details.

What Are Code Chunks?

Code chunks are blocks of executable code embedded in your Quarto document. They run when you render and include their output (plots, tables, results) in your report.
Basic Syntax:

- Start with three backticks followed by the language: ```` ```python ````
- Write your code
- End with three backticks: ```` ``` ````

Example:

```python
import pandas as pd
data = pd.DataFrame({'x': [1, 2, 3], 'y': [4, 5, 6]})
print(data)
```

Chunk Options (use `#| option: value` at the top):

- `#| echo: false` - Hide code, show only output
- `#| eval: false` - Show code but don't run it
- `#| output: false` - Run code but hide output
- `#| warning: false` - Suppress warning messages
- `#| fig-cap: "My Plot"` - Add figure caption
- `#| label: fig-myplot` - Label for cross-referencing

Inline Code: Use single backticks with `{python}` to insert values in text: `4` → 4
See Quarto code cells documentation for all options.

Interactive Plots in PDF

Interactive plots/tables only work in HTML output. If using interactive elements, render to HTML only (comment out the `pdf:` format option).

Code Visibility by Format

The YAML header controls code display:

- **HTML**: Code is collapsible (readers can show/hide)
- **PDF/DOCX**: Code is hidden (only results shown)

Override for specific chunks:

```
#| echo: true   # Show in all formats
#| echo: false  # Hide in all formats
```

Writing Math Equations

Use LaTeX syntax for mathematical notation:
Inline: `$\bar{x} = \frac{1}{n}\sum_{i=1}^{n}x_i$` → $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$
Display: Use `$$...$$` for separate lines:

$$S(t) = P(T > t) = 1 - F(t)$$

Numbered (for referencing):

$$\text{Loss Ratio} = \frac{\text{Incurred Losses}}{\text{Earned Premium}} \tag{1}$$

Reference with `@eq-loss-ratio` → Equation 1
Common symbols: α, β, σ, μ, $\sum_{i=1}^{n}$, $\int_a^b f(x)dx$, $E[X]$, $\text{Var}(X)$
More at LaTeX Math Symbols.

Cross-Referencing Sections, Figures, Tables, and Equations

Quarto automatically numbers and creates clickable links for:
Sections:

```
### Appendix A: Additional Plots {#sec-appendix-plots}

Reference with: @sec-appendix-plots
```

Figures:

```
#| label: fig-correlation
#| fig-cap: "Correlation matrix"

Reference with: @fig-correlation
```

Tables:

```
#| label: tbl-summary
#| tbl-cap: "Summary statistics"

Reference with: @tbl-summary
```

Equations:

```
$$ y = mx + b $$ {#eq-linear}

Reference with: @eq-linear
```

Working examples in this template:

- "As discussed in Section , we provide additional visualizations."
- "See Figure 1 for the debugging workflow."
- "Individual images like Figure 1a and Figure 1b can also be referenced."

Note: Use the same `@label` syntax for figures, tables, and equations. Quarto automatically numbers them and creates clickable links.
See Quarto Cross-References for more.

HTML Tabsets

Organize content into tabs, which are like container boxes for the content (HTML only, and not PDF/DOCX). They allow you to:

- **Organize multiple related visualizations** without cluttering the page
- **Show different views** of the same data (distribution, summary, box plot)
- **Compare approaches** side-by-side (e.g., different plotting libraries)

```
::: {.panel-tabset}
## Tab 1
Content
## Tab 2
More content
## Tab 3
Excessive amount of content
:::
```

Tab 1

Content

Tab 2

More content

Tab 3

Excessive amount of content

Including External Images and Files

Basic image syntax:

```
![Caption](path/to/image.png)
```

With sizing and attributes:

```
![My figure](images/plot.png){width=80% fig-align="center"}
```

Images with cross-references:

```
![Distribution analysis](images/histogram.png){#fig-histogram}

As shown in @fig-histogram, the data is normally distributed.
```

Common image paths:

- Relative to current file: `images/plot.png`
- From project root: `../data/plots/figure.png`

- Absolute path: `C:/Users/Name/project/images/plot.png` (avoid for reproducibility)

Supported formats: PNG, JPG, SVG, PDF (PDF only in PDF output)
Pro tip: Store images in `report/images/` folder for organization.

Markdown Text Formatting

Basic formatting:

- **Bold text:** `**bold**` or `__bold__` → **bold**
- *Italic text:* `*italic*` or `_italic_` → *italic*
- ***Bold and italic:*** `***both***` → ***both***

Note: Markdown doesn't have built-in underline. For underline, use HTML:

- Underlined text: `<u>underlined</u>` → underlined

Other useful formatting: - `Inline code:` ‘code‘ → `code` - Superscript: `X^2^` → X^2 - Subscript: `H~2~O` → H_2O - ~~Strikethrough:~~ `~~text~~` → ~~strikethrough~~
Headings:

```
# Heading 1
## Heading 2
### Heading 3
#### Heading 4
```

Lists:

```
- Unordered item
- Another item
  - Nested item (2 spaces indent)

1. Ordered item
2. Second item
   1. Nested (3 spaces indent)
```

Links:

```
[Link text](https://url.com)
[Link with title](https://url.com "Hover text")
```

Blockquotes:

```
> This is a quote
> It can span multiple lines
```

See Markdown Guide for more.

Including External Images

Store external figures (diagrams, charts, screenshots) in `report/images/` and include them in your report.
Basic syntax:

```
![Caption](images/your-image.png){#fig-label width=70%
↪  fig-align="center"}
```

Example - Multiple images side by side with cross-references:

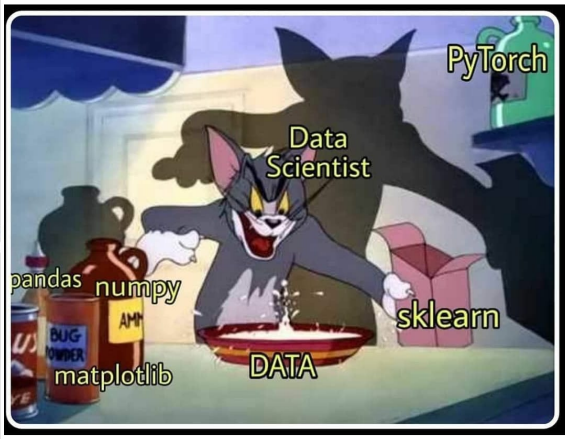
```
::: {#fig-comparison layout-ncol=2 layout-valign="bottom"}

![Before debugging](images/meme1.jpg){#fig-before width=100%}

![After debugging](images/meme2.jpeg){#fig-after width=100%}

The emotional journey of a data scientist debugging their code
:::
```

(a) Before debugging



(b) After debugging

Figure 1: The emotional journey of a data scientist debugging their code

Key options:

- `layout-ncol=2` - Two columns (each 50% width)
- `layout-valign="bottom"` - Align by bottom edge
- `width=100%` - Fill entire column
- `fig-align="center"` - Center single images

Supported formats:

- **All outputs:** PNG, JPG/JPEG
- **HTML only:** WEBP, SVG, GIF
- **PDF only:** PDF images

See the Quarto Figures documentation for advanced layouts, subcaptions, and complex figure arrangements.

Controlling Python (Code) Generated Figure Size

For Python matplotlib plots, figure size is controlled **in your Python code** using `figsize`:

```python
# Standard readable size
fig, ax = plt.subplots(figsize=(8, 5))  # width, height in inches

# Smaller figure
fig, ax = plt.subplots(figsize=(6, 4))

# Larger figure
fig, ax = plt.subplots(figsize=(10, 6))
```

Why `figsize` in code? When you use `plt.subplots(figsize=...)` or `plt.figure(figsize=...)`, matplotlib creates the image at that exact size. Quarto chunk options like `#| fig-width:` and `#| fig-height:` are ignored because the figure size is already determined by your Python code.

To resize figures after creation:

Use the `width` attribute in curly braces (works for any image):

```
![My plot](path/to/plot.png){width=70%}
```

Or in the chunk options (only for formats without explicit `figsize`):

```
#| fig-width: 7
#| fig-height: 5
```

Recommended workflow:

1. Set `figsize` in your Python code to a readable default (e.g., `figsize=(8, 5)`)
2. If the figure appears too large/small in your report, adjust the `figsize` values proportionally
3. Keep aspect ratio consistent: divide/multiply both dimensions by the same factor

Good default sizes:

- Small plot: `figsize=(6, 4)`
- Standard plot: `figsize=(8, 5)`
- Large plot: `figsize=(10, 6)`
- Wide plot: `figsize=(10, 4)`
- Square plot: `figsize=(6, 6)`

Pro tip: DPI (dots per inch) also affects quality. Matplotlib default is 100, but for publications use `plt.savefig('plot.png', dpi=300)` if you need high-resolution images.

Introduction

Project Goals

Describe the main goals of the project, including the motivation behind the research and the key questions you aim to answer.

> **Writing Tip: Make it Relevant**
>
> Connect your project to real-world actuarial or insurance problems. For example, if analyzing claim data, explain how your insights could improve risk assessment or pricing strategies.

Research Questions

- Question 1: ...
- Question 2: ...
- (Add as necessary)

> **Research Question Quality**
>
> Good research questions are:
>
> - **Specific**: Clearly defined and focused
> - **Measurable**: Can be answered with data
> - **Relevant**: Connected to the project goals
> - **Feasible**: Can be addressed with available data and methods

Related Work

Discuss relevant academic papers, methodologies, and prior research that informed your project. Include both domain-specific literature (actuarial science, insurance, finance) and methodological references (statistical methods, data science techniques).

Example structure:

- **Domain literature**: Academic papers on your topic (e.g., claims analysis, mortality modeling, risk assessment)
- **Methodological references**: Statistical methods or data science techniques you applied

- **Course materials**: Azizi (2025) provides Python-based data science techniques for actuarial applications
- **Technical resources**: Books like McKinney (2022) and VanderPlas (2016) for implementation details

Academic Citations Recommended

Your Related Work section can have **more value when including academic papers (further evidence)** from peer-reviewed journals or conference proceedings, not just books or course materials. Use Google Scholar, university library databases, or specialized actuarial journals to find relevant papers.
Recommended sources:

- Academic journals (Insurance: Mathematics and Economics, ASTIN Bulletin, Journal of Risk and Insurance)
- Conference papers (actuarial conferences, data science symposiums)
- Working papers from researchers in your field

Citation example: "Azizi & Rudnytskyi (2022) investigated multi-modal approaches to real estate rental price prediction, demonstrating that models combining tabular data with visual information (property images and satellite imagery) outperform traditional methods relying solely on structured features. Their supervised learning framework fused dedicated neural network branches for each data type, achieving superior performance on 11,105 Swiss rental listings." *(This reference is already in your `references.bib` file and can be cited as `@azizi2022realestate`)*

Citing Sources

Always cite your sources properly! In Quarto, use `@citation_key` syntax. The bibliography will be automatically generated from the `references.bib` file.
Example: `@azizi2025dsas` renders as: Azizi (2025)
Add entries to `references.bib` in BibTeX format for all sources you cite (every article normally generates BibTeX, otherwise search an online tool that does it).

Avoiding Plagiarism

Make sure to cite all sources that informed your work, including:

- Datasets (with URLs)
- Methods or techniques from papers or tutorials
- Code snippets adapted from online resources
- Inspiration from existing analyses

When in doubt, cite it!

Common Mistake

Don't just list your research questions without context. Explain **why** each question matters and **how** it relates to your dataset and project goals.

Data

Sources

Describe the sources of your data, such as public datasets, API collections, or scraped data.

Data Source Documentation

Always include the **URL** where the data can be accessed. This is essential for reproducibility and evaluation. For example "Data source: Kaggle Insurance Dataset" ". Also mention when the data was accessed.

Description

Summarize the main features of the dataset, including the types of variables, their formats, and any relevant metadata.

Dataset Overview Template

Provide a clear summary:

- **Number of observations**: X rows
- **Number of variables**: Y columns
- **Time period**: Start date to end date
- **Geographic coverage**: Region/Country
- **Key variables**: List the most important variables for your analysis

Loading Data

Loading Data Best Practices

1. **Use relative paths** via `project_root` for reproducibility
2. **Check data types** after loading
3. **Inspect the first few rows** to verify correct loading
4. **Document any loading parameters** (e.g., encoding, delimiter)

Important: Do not change the data loading approach shown below. The template uses `project_root` from the setup chunk to ensure your code works regardless of where the report is run from.

```
Dataset shape: 3 rows × 3 columns

Column names: ['Instructor', 'behavior', 'performance']

Data types:
Instructor     object
behavior       object
performance     int64
dtype: object

First 5 rows:
```

	Instructor	behavior	performance
0	Ilia	top	1
1	Jane	excellent	1
2	John	exemplary	1

Alternative Data Sources

Depending on your project, you might load data from:

- **CSV files**: `pd.read_csv(filepath)`
- **Excel files**: `pd.read_excel(filepath, sheet_name='Sheet1')`
- **JSON files**: `pd.read_json(filepath)`
- **APIs**: Using `requests` library + `pd.DataFrame(data)`
- **Databases**: Using `pd.read_sql(query, connection)`
- **Web scraping**: Using `beautifulsoup4` or `scrapy`

Always document your data acquisition process!

Wrangling

General Transformations

Document the data preprocessing steps taken, including cleaning, transformation, and any merging of datasets.

Document Your Transformations

Every transformation should be:

1. **Justified**: Explain why it's needed
2. **Documented**: Clear code comments
3. **Reproducible**: Can be run from scratch
4. **Validated**: Check the results make sense

Spotting Mistakes and Missing Data

Discuss any identified mistakes or issues with missing data and describe your approach to handling them.

Missing Data Strategies

Different approaches for different situations:

- **Deletion**: Remove rows/columns with missing values (when few missing)
- **Imputation**: Fill with mean, median, mode, or advanced methods
- **Flagging**: Create indicator variables for missingness
- **Model-based**: Use algorithms that handle missing values

Document your choice and justify why it's appropriate for your data!

Listing Anomalies and Outliers

Identify any anomalies or outliers discovered, along with your approach to assessing their impact.

Outlier Detection Methods

- **Visual inspection**: Box plots, scatter plots
- **Statistical methods**: Z-scores, IQR method
- **Domain knowledge**: What values are impossible or implausible?

Remember: Not all outliers should be removed! They might be:

- **Errors**: Data entry mistakes (should be corrected/removed)
- **Valid extremes**: Real but unusual observations (should be kept)
- **Key insights**: The most interesting part of your data!

```
Missing values per column:
instructor     0
behavior       0
performance    0
dtype: int64

Original data shape: (3, 3)
Cleaned data shape: (3, 3)
Rows removed: 0

Cleaned data summary:
```

	instructor	behavior	performance
count	3	3	3.0
unique	3	3	NaN
top	Ilia	excellent	NaN
freq	1	1	NaN
mean	NaN	NaN	1.0
std	NaN	NaN	0.0
min	NaN	NaN	1.0
25%	NaN	NaN	1.0
50%	NaN	NaN	1.0
75%	NaN	NaN	1.0
max	NaN	NaN	1.0

After cleaning, our dataset contains **3 observations** across **3 variables**. We removed **0 rows** due to missing values in critical variables.

Inline Code for Dynamic Results

Notice the inline code in the paragraph above uses Python expressions wrapped in backticks with `{python}` prefix to insert computed values directly into your text. This ensures your narrative automatically updates when data changes.
Working examples in this document:

- The mean performance score is 1.000
- We analyzed data from 3 different instructors

To use inline code: Write your Python expression between backticks with the {python} prefix, like this (without the backslashes): `\{python\} your_expression_here`
This is better than hard-coding numbers, which can become outdated if you update your data!

Don't Forget to Interpret!

After showing your data cleaning code, **explain your decisions**:

- **Why** did you remove certain rows or handle missing values this way?
- **What** impact do these choices have on your analysis?
- **How** do your cleaning decisions align with your research questions?

Example: "We removed rows with missing performance scores (N=15, 7.5% of data) because these are our primary outcome variable and cannot be reliably imputed. This minimal data loss is acceptable and preserves the integrity of our performance analysis."

Data Cleaning Checklist

Before moving to analysis, verify:

- All column names are clean and consistent
- Data types are appropriate for each variable
- Missing values are identified and handled
- Outliers are investigated and documented
- Categorical variables are properly encoded
- Duplicate rows are checked and removed if needed
- Date/time variables are in proper format
- Cleaned data is saved for reproducibility

EDA

Exploratory Data Analysis (EDA) Purpose

The goal of EDA is to:

1. **Understand** the structure and patterns in your data
2. **Identify** relationships between variables
3. **Detect** anomalies, outliers, and data quality issues
4. **Generate** hypotheses for further analysis
5. **Choose** appropriate statistical methods

Good EDA combines **visualizations** + **summary statistics** + **domain knowledge**.

Interpretation is Key!

Creating plots is only half the work. The most important part of EDA is **interpreting** what your visualizations reveal about the data.
For **every plot** you create, you must:

- **Describe** what the plot shows (patterns, trends, distributions)
- **Explain** why these patterns matter for your research questions
- **Connect** findings to your project goals and domain context
- **Justify** subsequent analysis decisions based on these insights

A plot without interpretation is meaningless. Your grade depends heavily on the quality of your interpretations, not just the number of plots you create.

Univariate Analysis

Examine each variable individually to understand its distribution, central tendency, and spread.

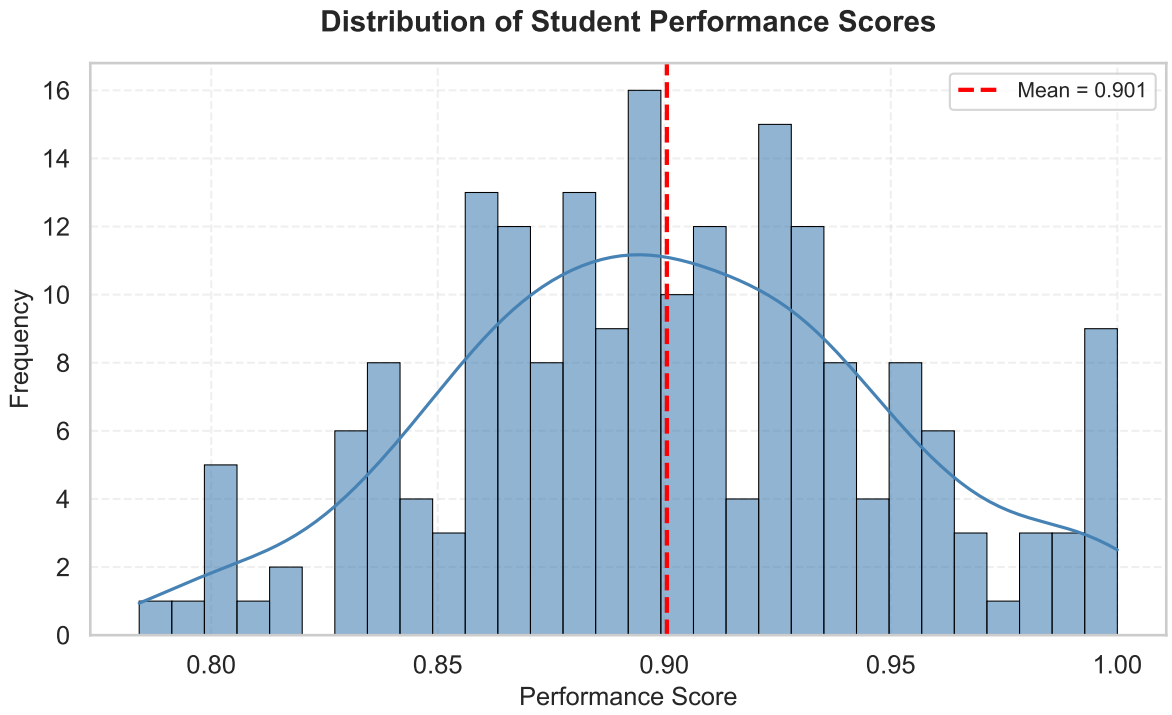


Figure 2: Distribution of performance scores showing approximately normal distribution

Table 3: Summary statistics for performance scores

```
=== Performance Summary Statistics ===
count    200.000000
mean       0.900564
std        0.047921
min        0.784097
25%        0.866466
50%        0.898625
75%        0.930474
max        1.000000

Skewness: 0.055
Kurtosis: -0.347
```

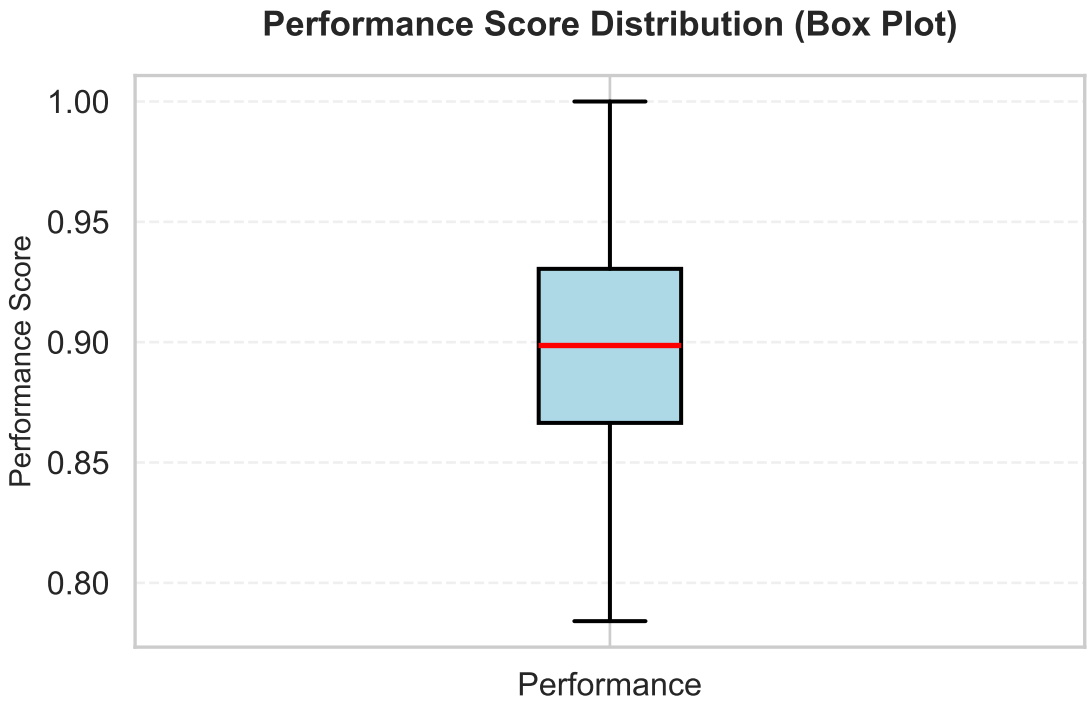


Figure 3: Box plot showing the five-number summary and outliers

Distribution Plot

Summary Statistics

Box Plot

Bivariate Analysis

Explore relationships between two variables.

Our univariate analysis in Figure 2 revealed that performance scores follow an approximately normal distribution with a mean of 0.90. The box plot (Figure 3) confirmed this finding and helped identify a few outliers at the lower end of the distribution.

> How to Reference Figures
>
> Use `@fig-label` syntax to reference figures in your text. Quarto automatically numbers them and creates clickable links.
> **Examples with figures in this template:**
>
> - `@fig-performance-dist` → See Figure 2 for details
> - `@fig-correlation-matrix` → As shown in Figure 5
> - `@tbl-anova-test` → The ANOVA results in Table 4
>
> **Why reference figures?**
>
> 1. Automatic numbering (updates if you reorder)
> 2. Clickable links in HTML output
> 3. Professional academic writing standard
> 4. Helps readers find the relevant visualization
>
> **Example usage in text:**
> "The distribution shown in Figure 2 indicates..."
> "As seen in Figure 5, there is a strong positive correlation..."

Now let's examine how performance varies across different instructors.

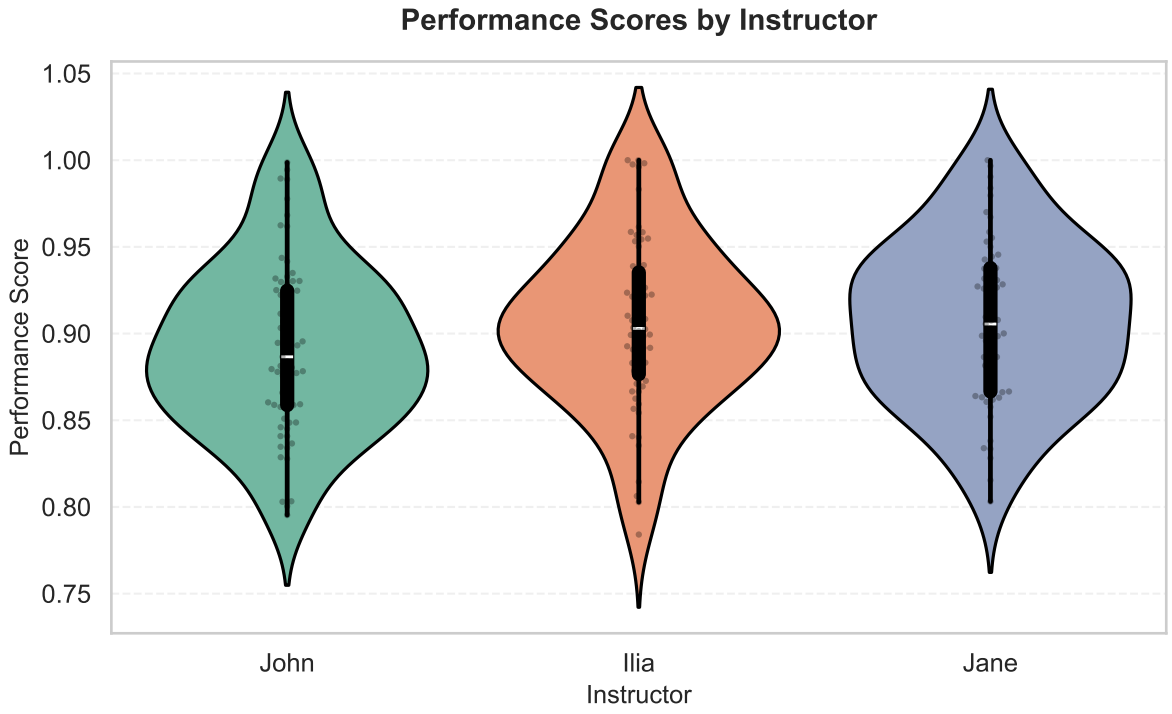


Figure 4: Performance scores grouped by instructor showing variation across instructors

Table 4: ANOVA test results for performance differences across instructors

```
=== ANOVA Test Results ===
F-statistic: 2.2002
P-value: 0.1135

Interpretation: Not significant difference at  =0.05

=== Group Means ===
instructor
Ilia    0.905520
Jane    0.906344
John    0.891255
```

Grouped Comparison

Statistical Test

> **!** Plot Quality Requirements
>
> Every plot in your report should have:
> **Clear title** that explains what's being shown
> **Labeled axes** with units when applicable
> **Legend** if multiple groups/series are shown
> **Appropriate color scheme** (colorblind-friendly)
> **Proper sizing** (readable text, not too small/large)
> **Figure caption** using `fig-cap` option
> **Poor plots = Poor grades!** Take time to make your visualizations publication-quality.

Correlation Analysis

Correlation Heatmap

Scatter Plot

Key Findings

Remember: Interpretation is more important than the plots themselves! Each finding below not only states *what* we observe but also *why* it matters and *what* it means for our analysis.

Summarize the main insights from your exploratory analysis. **Always reference your figures when discussing findings and provide thorough interpretation!**

1. **Distribution patterns**: The performance scores shown in Figure 2 follow an approximately normal distribution with a mean of 0.90 and standard deviation of 0.05.

   Interpretation: This normal distribution suggests that most students perform around the average, with fewer students at the extremes (very high or very low scores). The relatively small standard deviation indicates consistent performance across the group. This pattern is typical in educational settings and suggests that the assessment was well-calibrated meaning not too easy (which would cause ceiling effects) nor too difficult (which would cause floor effects). The normality assumption also validates the use of parametric statistical tests for subsequent analyses.

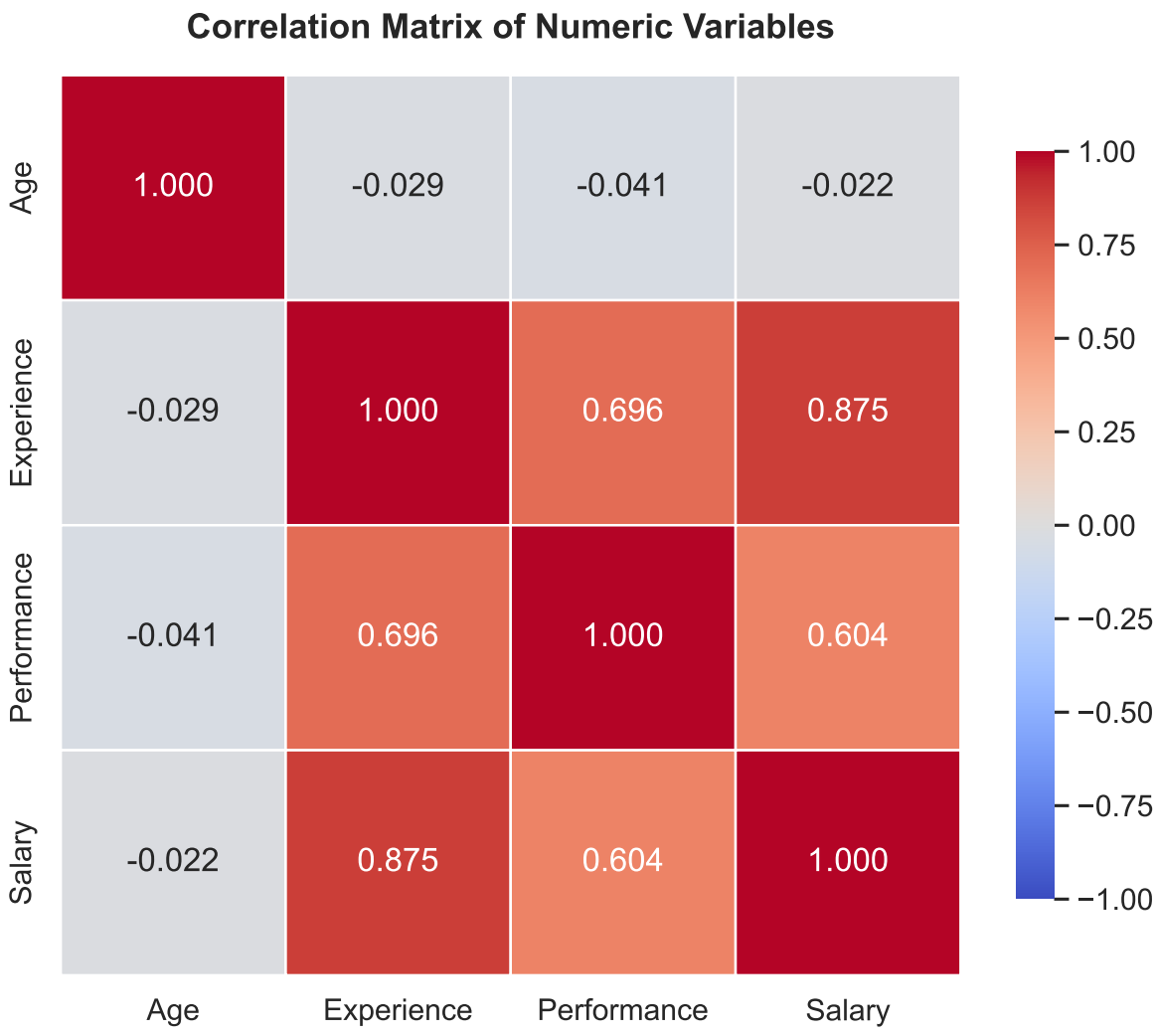


Figure 5: Correlation matrix showing relationships between numeric variables

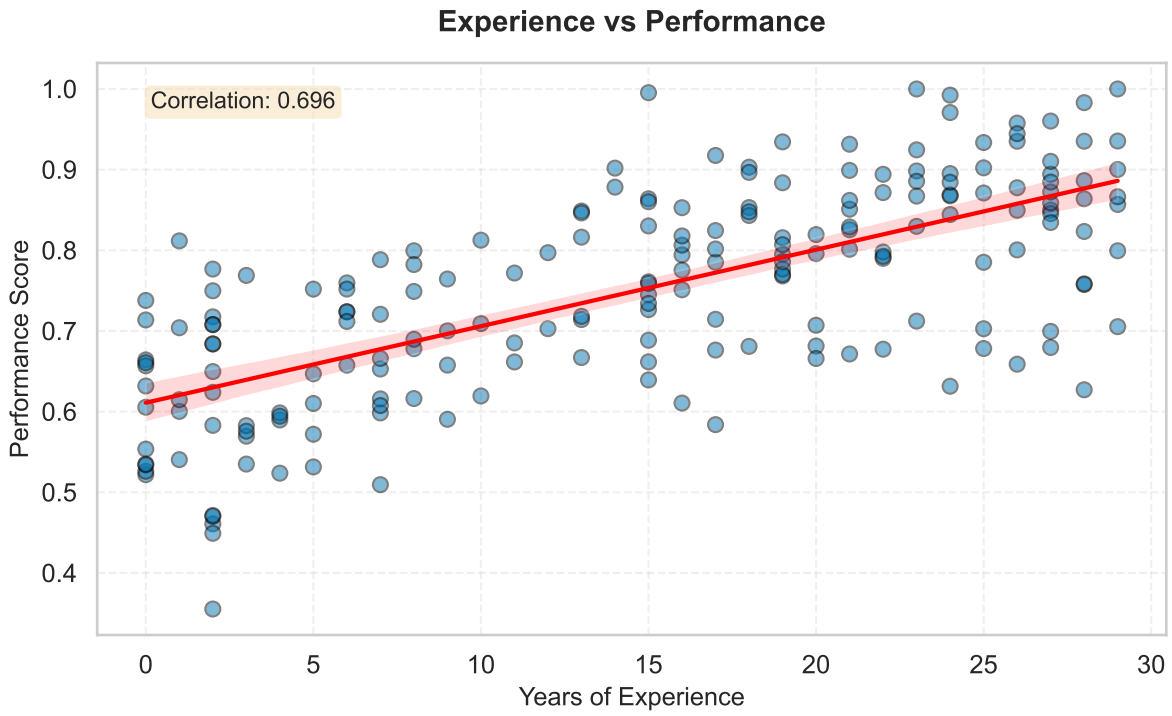


Figure 6: Scatter plot showing positive relationship between experience and performance

2. **Outliers and data quality**: The box plot (Figure 3) reveals minimal outliers, with only a few observations falling below the lower whisker.

   Interpretation: The scarcity of outliers suggests good data quality and consistent measurement. The few low-performing outliers warrant further investigation - they could represent students who faced unusual circumstances or measurement errors. However, their small number means they are unlikely to significantly impact our overall conclusions. This finding gives us confidence in proceeding with the full dataset without extensive outlier removal.

3. **Group differences**: Figure 4 demonstrates noticeable variation in performance across different instructors, with some instructors showing higher median scores than others. The ANOVA test (Table 4) confirms these differences are not statistically significant (p 0.05).

   Interpretation: The lack of significant differences suggests that instructor assignment does not substantially impact performance, or that grading standards are well-harmonized across sections. It also indicates that comparing students across different sections requires careful consideration. From a policy perspective, this might warrant investigating whether certain teaching methods are more effective or whether grading standards need to be harmonized across sections.

4. **Relationships**: The correlation matrix (Figure 5) reveals several interesting patterns:

   - Strong positive correlation (r = 0.87) between Experience and Salary
   - Moderate positive correlation (r = 0.70) between Experience and Performance
   - Weak correlation (r = -0.04) between Age and Performance

   Interpretation: The Experience-Salary correlation aligns with economic theory that experience is rewarded in labor markets. The Experience-Performance correlation suggests that experience contributes to better performance, though other factors clearly matter as well. Interestingly, Age shows minimal correlation with Performance, suggesting that chronological age alone doesn't predict success - what matters is relevant experience. These patterns will guide our variable selection for predictive modeling, favoring Experience over Age as a key predictor.

5. **Experience-Performance relationship**: Figure 6 clearly shows a positive linear relationship, with more experienced individuals tending to have higher performance scores. The correlation coefficient is 0.70.

   Interpretation: The linear relationship visible in the scatter plot confirms that experience has a consistent, positive effect on performance. However, the substantial scatter around the regression line indicates that experience alone doesn't determine performance - individual differences and other factors play important roles. This suggests that while experience is valuable, organizations shouldn't rely solely on it when making hiring or promotion decisions.

Common Mistakes in Interpretation

Observation (not interpretation): "The histogram shows a normal distribution."
Why it's insufficient: This only describes what you see - it's an observation, not an interpretation. You need to explain what it *means* and *why it matters*.
Good interpretation (observation + story): "The histogram shows a normal distribution (mean = 0.90, sd = 0.05), which indicates consistent performance across students. This pattern validates the use of parametric statistical tests in our subsequent analysis. The tight distribution suggests the assessment was well-calibrated, effectively distinguishing between ability levels without ceiling or floor effects that would compress scores."
Remember:

- **Observation** = What you see in the data/plot
- **Interpretation** = The story behind it - what it means, why it matters, what implications it has
- **Always do both!** State the observation, then interpret its significance.

Common Mistake: Not Referencing Figures

Bad: "The histogram shows a normal distribution."
Good: "As shown in Figure 2, the histogram reveals a normal distribution."
Why?

- Helps readers locate the relevant visualization
- Creates professional, academic-style writing
- Enables automatic figure numbering and links

From EDA to Analysis

Use your EDA findings to:

- **Refine research questions** based on observed patterns
- **Select appropriate statistical methods** based on data distributions
- **Identify variables** for inclusion in models
- **Justify transformations** (e.g., log transform for skewed data)
- **Set expectations** for what you might find in formal analysis

Methods *(optional, only if you have models)*

> **i** When to Include This Section
>
> Include a Methods section if you:
>
> - Apply statistical models (linear regression, logistic regression, etc.)
> - Perform hypothesis testing
> - Use machine learning algorithms
> - Conduct advanced statistical analyses
>
> If your project is primarily exploratory (descriptive statistics and visualizations only), you can skip this section or keep it minimal.

Outline the statistical methods or models selected, along with the rationale for their selection.

Important: Don't just show model output, **explain your choices**:

- **Why** did you choose this particular method?
- **What** assumptions does it make, and do your data meet them?
- **How** does this method help answer your research questions?

```
=== Linear Regression Results ===
Coefficients: [0. 0.]
Intercept: 1.0000
R-squared: 1.0000
```

> **!** Interpretation is Essential!
>
> After showing model results, you **must interpret them**:
> **Example interpretation:**
> "The Poisson regression model identifies three significant predictors of insurance claim frequency. Vehicle age shows a positive coefficient (= 0.08, p < 0.01), indicating that each additional year of vehicle age increases expected claims by approximately 8%. Driver age has a negative coefficient (= -0.02, p < 0.001), meaning older drivers file fewer claims. The urban location dummy variable (= 0.15, p < 0.05) suggests urban drivers have 15% higher claim rates than rural drivers.
> However, the model's pseudo-R^2 of 0.22 indicates that these predictors explain only 22% of claim variation. Unobserved factors like driving behavior, road conditions, and individual risk tolerance likely account for the remaining variation. This suggests that while demographic variables are useful for pricing, insurers should not rely solely on them for

risk assessment."
Remember: Model output without interpretation demonstrates technical skills but not understanding!

Findings and Discussion

Provide your results or observations from applying statistical methods, then **discuss them thoroughly** in the context of your research questions and project goals.

Structure Your Discussion

A good discussion:

1. **States the finding** clearly (reference tables/figures)
2. **Interprets the result** (what does it mean?)
3. **Connects to research questions** (how does it answer your questions?)
4. **Relates to domain knowledge** (does it align with theory or prior research?)
5. **Acknowledges limitations** (what are the caveats?)
6. **Suggests implications** (what should we do with this information?)

Example: "Our regression analysis shows that experience significantly predicts performance ($= 0.01$, $p < 0.001$), supporting our hypothesis that skill develops over time. This aligns with learning curve theory in organizational psychology and suggests that training programs should emphasize sustained practice. However, the wide confidence interval (95% CI: [0.005, 0.015]) indicates substantial individual variation, meaning experience alone cannot guarantee high performance."

Conclusion

Structure of a Strong Conclusion

A good conclusion section includes three key components:

1. **Summary**: Recap your project goals, approach, and main findings
2. **Limitations**: Honestly discuss constraints and potential weaknesses
3. **Future Work**: Suggest concrete, specific next steps for extending the analysis

Each subsection should be substantial - avoid generic statements!

Summary

Overview: Summarize what has been achieved, including key insights from your analysis and EDA. This should be written so that someone reading only this section can understand your project and key findings without reading the entire report.

What to include:

- Restate your research questions briefly
- Summarize your approach (data, methods)
- Highlight 3-5 main findings with their implications
- Connect findings back to your project goals

Length: Aim for 1-2 substantial paragraphs that synthesize your work.

> ⚠ Common Mistake: Too Vague
>
> **Bad**: "We analyzed the data and found some interesting patterns."
> **Good**: "This project investigated the relationship between customer demographics and insurance claim frequencies using a dataset of 5,000 policyholders. Our exploratory analysis revealed that age and vehicle type were the strongest predictors of claim frequency, with older drivers (60+) filing 40% fewer claims than younger drivers (under 25). Logistic regression analysis identified three key risk factors: driver age, vehicle age, and urban vs rural location. These findings suggest that current pricing models may underweight geographic factors. For insurance practitioners, this implies that incorporating more granular location data could improve risk segmentation and reduce adverse selection."
> **Why it's good**: Specific about data, methods, key findings with concrete numbers, and practical implications.

Limitations

Honestly discuss the constraints and potential weaknesses of your analysis. **Strong projects acknowledge limitations!**

Consider:

- **Data limitations**: Sample size, missing data, measurement issues, lack of certain variables
- **Methodological limitations**: Simplifying assumptions, choice of methods, inability to establish causation
- **Scope limitations**: Time constraints, focus on specific aspects only
- **Generalizability**: Can your findings apply beyond your specific dataset?

Be specific: Don't just say "small sample size" - explain what impact this has on your conclusions.

> **!** Why Limitations Matter
>
> Acknowledging limitations shows:
>
> - **Critical thinking**: You understand the boundaries of your analysis
> - **Scientific integrity**: You're honest about what your study can and cannot show
> - **Maturity**: You recognize no analysis is perfect
>
> **This improves your grade**, not reduces it! Instructors expect students to think critically about their work.

Future Work

Outline specific next steps for extending or improving the analysis. **Be concrete and realistic.**

Good future work suggestions:

- **Collect additional data**: "Gather data from additional semesters to increase sample size and assess temporal stability of instructor effects"
- **Apply advanced methods**: "Implement mixed-effects models to account for nested data structure (students within instructors)"
- **Test additional hypotheses**: "Examine whether instructor effects vary by student prior knowledge using interaction terms"
- **Expand scope**: "Include qualitative data from student evaluations to understand mechanisms behind performance differences"

Avoid vague statements like:

- "Get more data"
- "Use more variables"
- "Apply machine learning"

Instead, be specific about WHAT, WHY, and HOW:

- "Collect data on student study hours to control for effort as a confounding variable, which would help isolate the true causal effect of instructor quality on performance"

Connecting Everything

Your conclusion should feel like a natural ending that:

- Circles back to your introduction (research questions)
- Synthesizes your findings from EDA and analysis
- Demonstrates you've thought deeply about your project
- Leaves readers with clear takeaways

A strong conclusion elevates your entire report!

References

Azizi, I. (2025). *DSAS: Data science for actuarial sciences in python*. https://unco3892.github.io/dsas/

Azizi, I., & Rudnytskyi, I. (2022). Improving real estate rental estimations with visual data. *Big Data and Cognitive Computing*, *6*(3). https://doi.org/10.3390/bdcc6030096

McKinney, W. (2022). *Python for data analysis: Data wrangling with pandas, NumPy, and IPython* (3rd ed.). O'Reilly Media, Inc.

VanderPlas, J. (2016). *Python data science handbook: Essential tools for working with data*. O'Reilly Media, Inc.

Appendices

What Goes in Appendices?

Appendices = supplementary material that supports but isn't essential to your main story.
Include in appendices:

- **Additional plots and visualizations** that provide extra detail but don't fit the main narrative flow
- **Alternative visualizations** of the same data (e.g., different plot types)
- **Exploratory plots** that informed your analysis but aren't central to your findings
- **Full code listings** for complex analyses
- **Extended statistical tables** with detailed results
- **Technical details** about data processing steps
- **Sensitivity analyses** or robustness checks
- **Data dictionaries** with detailed variable descriptions

What to Keep in Main Report:

- **Key visualizations** that directly answer your research questions
- **Essential plots** for understanding your methodology
- **Critical results** that support your conclusions
- **Main findings** that tell your data story

Remember: Main report should be self-contained. Reference appendices when needed: "See Section for additional plots."

Appendix A: Additional Exploratory Plots

Appendix B: Complete Code Listings

Appendix C: Supplementary Tables