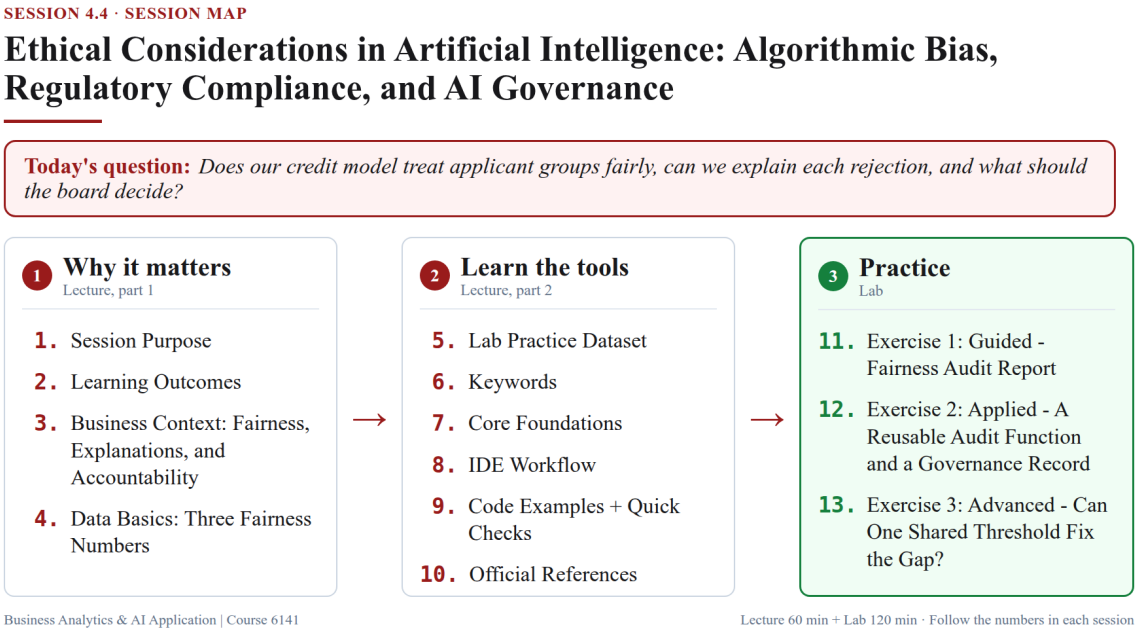
Session 4.4: Ethical Considerations in Artificial Intelligence: Algorithmic Bias, Regulatory Compliance, and AI Governance

Course 6141: Business Analytics & AI Application | Reading for Week 16



Schedule: Week 16 | In-person classes (1:2 Delivery Ratio: 60 Min Lecture + 120 Min Lab)

1. Session Purpose

- A consumer lender approves loan applications automatically when its model's approval score is `0.70` or higher
- Before the model's annual review, the board's risk committee must confirm that it treats applicant groups fairly and that every rejection can be explained to the applicant
- The chief risk officer asks: does our credit model treat applicant groups fairly, can we explain each rejection, and what should the board decide? Today, you audit the model for fairness, explain its decisions, and prepare the facts the board needs

2. Expected Learning Outcomes (By the end of this session, you will be able to)

1. "Are the groups treated alike?": measure approval rates by group and the disparate impact ratio, and test it against the four-fifths rule
2. "Why was this applicant rejected?": explain each decision with feature contributions (how much each feature pushed the score up or down)
3. "What should the board decide?": compare one shared threshold's fairness and risk, write a governance record, and fix a `NameError` caused by a variable that only exists inside a function

3. Business Context: Fairness, Explanations, and Accountability

- A model learns patterns from past data. If the past was unfair, for example because some groups had less access to credit, the model can repeat that unfairness at scale, without anyone intending it.
- Two kinds of unfair treatment:
  - **Disparate treatment:** using a person's group directly in the decision, such as a different threshold for each group. In lending, this is prohibited in many countries, including under the US Equal Credit Opportunity Act (ECOA).
  - **Disparate impact:** a rule that treats everyone the same on paper, but approves one group far less often. It is a warning sign that must be investigated and justified.
- The four-fifths rule: if one group's approval rate is less than 80% (four-fifths) of the most-approved group's rate, the gap needs review. It comes from US employment guidelines (1978) and is widely borrowed as a first screening test. Failing it is a reason to investigate, not proof of discrimination.
- Explanations are required: US lenders must give rejected applicants the main reasons (an adverse action notice), and the EU AI Act (European Union Artificial Intelligence Act) treats credit scoring of people as high-risk, with duties for documentation, human oversight, and transparency.
- Governance: the board stays accountable for automated decisions. It needs a written record of what was tested, what was found, and who decided what.

4. Data Basics: Three Fairness Numbers

- Approval rate of a group = approved applicants in the group / all applicants in the group.
- Disparate impact ratio = the lower group's approval rate / the higher group's approval rate. `1.0` means equal rates; below `0.80` fails the four-fifths rule.
- Feature contribution (for a model that adds up features, like linear regression) = coefficient x (applicant's value - average value). A negative contribution pulled the score down. For such a model, this number is also the feature's SHAP value (SHapley Additive exPlanations, a widely used explanation method).
- Our approval scores come from a model whose formula we were not given. We fit a linear regression that copies it from the scores (a surrogate model) and explain that copy. It reproduces the scores well (`R-squared 0.975`), so its explanations are trustworthy for this model.

5. Lab Practice Dataset

- File Link: `data/session_04_04_dataset.csv`
- Audited 16-row loan application dataset (fictional teaching data), one row per applicant, 8 in each group:

  - `Applicant_ID`: Applicant code (e.g., `APP-6001`).
  - `Demographic_Group`: An anonymized protected group, `Group_A` or `Group_B`. A protected group is defined by a characteristic such as race, sex, or age that the law forbids using against an applicant. The model does not use this column; auditors use it only to test fairness.
  - `Credit_Score_Count`: Credit score from a credit bureau (a company that records how people repay debts); higher is better.
  - `Debt_To_Income_Pct`: Monthly debt payments as a share of monthly income (DTI); lower is safer.
  - `Algorithmic_Approval_Score`: The model's score from 0 to 1; approved at `0.70` or higher.
  - `Adverse_Action_Notice_Flag`: `1` if the applicant was rejected and must receive an adverse action notice, else `0`.

6. Keywords (Core Programming Concepts)

- `Disparate Impact`:
  - What: a rule that approves one group much less often than another, even though it never uses the group
  - Why: the most common way bias enters automated decisions
- `Disparate Impact Ratio` and the `Four-Fifths Rule`:
  - What: the lower approval rate divided by the higher one, tested against `0.80`
  - Why: a simple, widely used first test of fairness
- `Disparate Treatment`:
  - What: deciding differently because of a person's group
  - Why: prohibited in lending in many countries, so it can never be the fix
- `Explainability` (feature contributions, SHAP):
  - What: showing how much each feature pushed a decision up or down
  - Why: applicants, regulators, and managers need reasons, not just scores
- `Surrogate Model`:
  - What: a simple model fitted to copy a model whose formula is hidden
  - Why: makes a hidden model explainable, as long as the copy is accurate
- `AI Governance`:
  - What: the rules, records, and people that keep AI decisions accountable
  - Why: the board, not the model, answers for the decisions

7. Core Programming & Computational Foundations

- **7.1 Approval rates by group: the average of a 0/1 column**

- *The idea:* Fairness checks start with a simple question: what share of each group was approved? If approved applicants are marked `1` and rejected ones `0`, the average of that column is the share of `1`s, so `df.groupby("Demographic_Group")["Approved"].mean()` gives each group's approval rate in one line.
- *See it on real data:* threshold `0.70`.

Group	Applicants	Approved	Approval rate
`Group_A`	`8`	`7`	`0.875`
`Group_B`	`8`	`2`	`0.250`

- *Common mistake:* counting approvals instead of rates. If one group had 80 applicants and another 8, raw counts would hide the comparison; rates put groups of any size on the same scale.
- *More examples:*

Code	Result	What it tells you
`df["Approved"].mean()`	`0.5625`	9 of all 16 applicants approved
`df.groupby("Demographic_Group")["Approved"].sum()`	A `7`, B `2`	approvals per group
`df.groupby("Demographic_Group")["Approved"].count()`	A `8`, B `8`	applicants per group
rate = sum / count	A `0.875`, B `0.250`	the same rates as `.mean()`

- *Remember:* the mean of a 0/1 column is a rate.

- **7.2 The disparate impact ratio and the four-fifths rule**
  - *The idea:* Divide the lower approval rate by the higher one. `1.0` means equal treatment in outcomes; the lower the number, the bigger the gap. Under the four-fifths rule, a ratio below `0.80` is a warning that must be investigated.
  - *See it on real data:* `0.250 / 0.875 = 0.286`. To pass, Group B would need an approval rate of at least `0.80 x 0.875 = 0.70`: 6 of its 8 applicants instead of 2.
  - *Common mistake:* treating the rule as a verdict. Failing it does not prove discrimination, and passing it does not prove fairness. It tells you where to look.
  - *More examples:* the ratio at other shared thresholds (Exercise 3 computes the full table).

Threshold	Group A rate	Group B rate	Ratio	Passes?
`0.45`	`1.000`	`1.000`	`1.000`	yes
`0.50`	`1.000`	`0.875`	`0.875`	yes
`0.60`	`1.000`	`0.500`	`0.500`	no
`0.70`	`0.875`	`0.250`	`0.286`	no
`0.75`	`0.875`	`0.000`	`0.000`	no

- *Remember:* lower rate / higher rate; below `0.80`, investigate.

- **7.3 Where a gap comes from: the features**
  - *The idea:* Our model never sees the group column. But if the groups differ on the features the model uses, its scores and approvals will differ too. This is how disparate impact appears without anyone intending it.
  - *See it on real data:*

Group average	Group A	Group B
Credit score	730.62	670.00
Debt-to-income (DTI)	26.58%	36.05%
Approval score	0.86	0.60
Lowest and highest score	0.66 to 0.94	0.49 to 0.72

  - *The real question:* do these feature differences reflect real repayment risk, or past unequal access to credit? A credit score depends partly on how long someone has had access to credit. Data alone cannot settle this; people must investigate.
  - *More examples:*

Measure	Group A	Group B
lowest and highest credit score	690 to 760	640 to 705
share with DTI above 35%	0.0	0.5
correlation of score with credit score (all 16)	0.987	
correlation of score with DTI (all 16)	-0.982	

  - *Remember:* a model can be unfair without using the group column; look at the features.
- **7.4 Explaining one decision: feature contributions**
  - *The idea:* For a model that adds up its features, like linear regression, each feature's contribution is coefficient x (applicant's value - average value). The contributions show how much each feature pushed this applicant's score above or below the average score. They add up exactly: average score + all contributions = the applicant's score.
  - *See it on real data:* our surrogate model has coefficients 0.00292 per credit point and -0.00734 per DTI point; the averages are a credit score of 700.31, a DTI of 31.31%, and a score of 0.7275.

Applicant	Credit effect	DTI effect	Average + effects	Decision
APP-6007 (640, 41.2%)	-0.176	-0.073	0.7275 - 0.176 - 0.073 = 0.479	rejected
APP-6012 (690, 34.0%)	-0.030	-0.020	0.678	rejected, just below 0.70
APP-6006 (760, 22.0%)	+0.174	+0.068	0.970	approved

    For APP-6007, the low credit score is the main reason (-0.176), more than twice the effect of the high debt ratio.

  - *Common mistake:* explaining a model you have not checked. A surrogate's explanations are only as good as its fit; ours reproduces the scores with an R-squared of `0.975`.
  - *More examples:*

Applicant	Credit effect	DTI effect	Decision
`APP-6000` (720, 28.5%)	`+0.058`	`+0.021`	approved
`APP-6005` (705, 29.5%)	`+0.014`	`+0.013`	approved, close to average
`APP-6013` (700, 30.5%)	`-0.001`	`+0.006`	approved, almost exactly average
`APP-6001` (680, 34.2%)	`-0.059`	`-0.021`	rejected: credit score is the main reason

  - *Remember:* contribution = coefficient x (value - average); the largest negative one is the main reason.

- **7.5 Fair process: what a fix may and may not do**
  - *The idea:* When a model fails a fairness test, the fix itself must be lawful and responsible.

Possible fix	Allowed?	Why
A lower threshold for one group	no	deciding by group is disparate treatment, prohibited in lending in many countries
One lower threshold for everyone	only with care	Exercise 3 shows it approves applicants above the risk limit
Investigate whether the features measure risk equally well for both groups	yes	addresses the cause
Add data such as rent and utility payment history	yes	gives applicants with thin credit records a fair hearing
Human review of rejections near the threshold, documented	yes	a person checks the borderline cases

  - *More examples:*

A governance record states	Example from this session
the fairness test and its result	ratio `0.286`, fails the four-fifths rule
the threshold in use	`0.70`, the lender's current approval rule
the explanation method and its fit	surrogate model, R-squared `0.975`
the actions decided and their owners	feature investigation; human review near the threshold

  - *Remember:* never decide by group; fix the cause, add oversight, and document every decision.

8. Development Workflow in Google Antigravity IDE

- `[1]` File Explorer: create your script with your name and the session code (for example, `Reacher_S44_demo_1.py`), and check the dataset is in `data/`.
- `[2]` Code Editor: write a few lines, run, repeat. Small steps keep errors small.
- `[3]` AI Agent Chat: on an error, capture the traceback (Python's error report) with `Win + Shift + S` (macOS: `Cmd + Shift + 4`), paste it with `Ctrl + V`, and ask why it failed.
- `[4]` Terminal: run `python` followed by your script name. Return code `0` means the script finished with no errors.

9. Hands-On Coding Micro-Examples

- Every example has three parts: **1. Input**, **2. Process**, and **3. Output**. Name each file with your own name and the session code, for example `Reacher_S44_demo_1.py` (pattern: `YourName_S44_demo_1.py`).
- *Micro-Example 1 (Approval Rates and the Four-Fifths Rule):* `YourName_S44_demo_1.py`

```python
# 1. Input: the 16 applications
import pandas as pd
df = pd.read_csv("data/session_04_04_dataset.csv")

# 2. Process: decisions at 0.70, approval rate per group, and the ratio
df["Approved"] = (df["Algorithmic_Approval_Score"] >= 0.70).astype(int)
rates = df.groupby("Demographic_Group")["Approved"].mean()
ratio = rates.min() / rates.max()

# 3. Output: rates, ratio, and the four-fifths test
print(rates)
print(f"Disparate impact ratio: {ratio:.3f}")
print("Passes the four-fifths rule:", ratio >= 0.80)
```

  - *How it works, line by line:*
    - `(df["Algorithmic_Approval_Score"] >= 0.70).astype(int)` turns each score into `1` (approved) or `0` (rejected).
    - `df.groupby("Demographic_Group")["Approved"].mean()` gives each group's approval rate (section 7.1).
    - `rates.min() / rates.max()` divides the lower rate by the higher, whichever group it is (section 7.2).
    - `ratio >= 0.80` gives `True` or `False`: the four-fifths test.
  - *Read the output:*

```
Demographic_Group
Group_A    0.875
Group_B    0.250
Name: Approved, dtype: float64
Disparate impact ratio: 0.286
```

```
Passes the four-fifths rule: False
```

  - 7 of 8 Group A applicants are approved, but only 2 of 8 in Group B.
  - The ratio, `0.286`, is far below `0.80`: the gap must be investigated before the model's next review.

- *Quick Check 1 (`.map()`):*
  - New tool: `column.map({1: "Approved", 0: "Rejected"})` replaces each value using a dictionary. Why: tables for the board should use words, not codes.
  - Task: add two lines to part **3. Output**: `df["Decision"] = df["Approved"].___({1: "Approved", 0: "Rejected"})` and `print(pd.crosstab(df["Demographic_Group"], df["Decision"]))`
  - Check: Group A: `7` approved, `1` rejected; Group B: `2` approved, `6` rejected.

- *Micro-Example 2 (Where Does the Gap Come From?):* `YourName_S44_demo_2.py`

```python
# 1. Input: the 16 applications
import pandas as pd
df = pd.read_csv("data/session_04_04_dataset.csv")

# 2. Process: average features and score by group
profile = df.groupby("Demographic_Group")[
    ["Credit_Score_Count", "Debt_To_Income_Pct", "Algorithmic_Approval_Score"]
].mean().round(2)

# 3. Output: the two group profiles
print(profile)
```

  - *How it works, line by line:*
    - `df.groupby("Demographic_Group")[[...]].mean()` averages the two features and the score in each group (section 7.3).
    - The group column is used here only by the auditor, to compare groups; the model never sees it.
  - *Read the output:*

```
                  Credit_Score_Count  ...  Algorithmic_Approval_Score
Demographic_Group                     ...
Group_A                       730.62  ...                        0.86
Group_B                       670.00  ...                        0.60
```

    - The `...` hides the debt-to-income column: Group A `26.58`%, Group B `36.05`%. Quick Check 2 shows the full table.
    - Group B has lower credit scores and higher debt ratios on average, so it gets lower approval scores. The gap in approvals follows from the gap in features.

- *Quick Check 2 (`.T`):*
  - New tool: `table.T` flips a table so that rows become columns. Why: with many measures and few groups, the flipped table is easier to read in a report.

  - Task: in part **3. Output**, change the line to `print(profile._)`
  - Check: the measures are now rows and `Group_A` and `Group_B` are the two columns.

- *Micro-Example 3 (Explain Each Decision):* `YourName_S44_demo_3.py`

```
# 1. Input: the 16 applications and the two features
import pandas as pd
from sklearn.linear_model import LinearRegression
df = pd.read_csv("data/session_04_04_dataset.csv")
features = ["Credit_Score_Count", "Debt_To_Income_Pct"]

# 2. Process: copy the score with a linear model, then contributions
surrogate = LinearRegression().fit(df[features], df["Algorithmic_Approval_Score"])
contributions = (df[features] - df[features].mean()) * surrogate.coef_
contributions.columns = ["Credit_Score_Effect", "Debt_Ratio_Effect"]
report = pd.concat([df[["Applicant_ID", "Algorithmic_Approval_Score"]],
                    contributions.round(3)], axis=1)

# 3. Output: how well the copy fits, then the explanations
print("Surrogate R-squared:", round(surrogate.score(df[features],
 ↪  df["Algorithmic_Approval_Score"]), 3))
print(report[df["Algorithmic_Approval_Score"] < 0.70])
```

  - *How it works, line by line:*

    * `LinearRegression().fit(df[features], df["Algorithmic_Approval_Score"])` fits a simple model that copies the scores from the two features: the surrogate (section 7.4).
    * `(df[features] - df[features].mean()) * surrogate.coef_` computes every applicant's contribution for both features at once: each column's average is subtracted, and each column is multiplied by its own coefficient.
    * `contributions.columns = [...]` gives the two results readable names.
    * `pd.concat([...], axis=1)` puts the IDs, scores, and contributions side by side.
    * `report[df["Algorithmic_Approval_Score"] < 0.70]` keeps the rejected applicants only.

  - *Read the output:*

```
Surrogate R-squared: 0.975
   Applicant_ID  ...  Debt_Ratio_Effect
1      APP-6001  ...             -0.021
3      APP-6003  ...             -0.053
7      APP-6007  ...             -0.073
...
15     APP-6015  ...             -0.067
```

    * The surrogate explains 97.5% of the score differences, so its explanations can be trusted.
    * All 7 rejected applicants have negative effects. The hidden `Credit_Score_Effect` column is larger in every case, so the credit score is the main reason in each adverse action notice (Quick Check 3).

- *Quick Check 3 (`.idxmin(axis=1)`):*

- New tool: `contributions.idxmin(axis=1)` returns, for each row, the column with the most negative value. Why: that column is the main reason to print on the adverse action notice.
- Task: add one line at the end of part **2. Process**: `report["Main_Reason"] = contributions._____(axis=1)`
- Check: the rejected applicants all show `Credit_Score_Effect` as the main reason.

10. Official Python Documentation & References

- European Commission: Regulatory Framework for Artificial Intelligence (EU AI Act).
- US Consumer Financial Protection Bureau: Equal Credit Opportunity Act (Regulation B).
- Fairlearn: A Python Toolkit for Assessing and Improving Fairness in AI.
- Barocas, Hardt, Narayanan: Fairness and Machine Learning: Limitations and Opportunities.

11. Exercise 1: Guided Practice - Fairness Audit Report

- *Step 1: Open Script:* In Google Antigravity IDE, create a new script named `YourName_S44_governance.py`, replacing `YourName` with your own name (for example, `Reacher_S44_governance.py`).
- *Step 2: Audit the Current Threshold:* Type the following code:

```
# 1. Input: the 16 applications
import pandas as pd
df = pd.read_csv("data/session_04_04_dataset.csv")

# 2. Process: decisions, rates, ratio, and a check of the notice flags
df["Approved"] = (df["Algorithmic_Approval_Score"] >= 0.70).astype(int)
rates = df.groupby("Demographic_Group")["Approved"].mean()
ratio = rates.min() / rates.max()
notices_match = ((1 - df["Approved"]) == df["Adverse_Action_Notice_Flag"]).all()

# 3. Output: the audit findings
print(f"Group A approval: {rates['Group_A'] * 100:.1f}% | "
      f"Group B approval: {rates['Group_B'] * 100:.1f}%")
print(f"Disparate impact ratio: {ratio:.3f}")
if ratio < 0.80:
    print("ALERT: fails the four-fifths rule; investigate before the next
    ↪  review.")
else:
    print("Passes the four-fifths rule.")
print("Every rejected applicant is flagged for a notice:", notices_match)
```

  - `1 - df["Approved"]` turns approved (`1`) into `0` and rejected (`0`) into `1`, matching the notice flag.

- *Step 3: Interpret:* In two comment lines, explain why failing the four-fifths rule calls for an investigation, and why it does not by itself prove the model discriminates.

- *Step 4: Execute in Terminal:* Run `python YourName_S44_governance.py` in Panel `[4]` and confirm return code `0`.
- *Success Check:* `Group A approval: 87.5% | Group B approval: 25.0%`; ratio `0.286`; the alert line prints; `Every rejected applicant is flagged for a notice: True`.

12. Exercise 2: Applied Practice - A Reusable Audit Function and a Governance Record

- *Coding Challenge:* In `YourName_S44_practice.py`, turn the audit into a function and save a governance record for the board.
- *Task 1 (Audit Function):* Write `approval_audit(df, threshold)` that computes the approval rates and `ratio` at any threshold, exactly as in Exercise 1.
- *Task 2 (Read the NameError & AI Diagnosis):* Call `approval_audit(df, 0.70)`, then write `print(ratio)` on the next line, outside the function. The script stops with `NameError: name 'ratio' is not defined`. Capture the traceback with `Win + Shift + S`, paste it into Panel `[3]` (AI Agent Chat) with `Ctrl + V`, and ask why a variable made inside a function cannot be seen outside it.
- *Task 3 (Return, Record, Save):* Make the function end with `return rates, ratio`, and call it as `rates, ratio = approval_audit(df, 0.70)`. Build a one-row governance record with `pd.DataFrame([{...}])` and the keys `Model`, `Threshold`, `Group_A_Rate`, `Group_B_Rate`, `Disparate_Impact_Ratio`, `Passes_Four_Fifths`, and `Status`, where `Model` is `"Credit_Approval_Model_v1"` and `Status` is `"Board review required"` if the test fails. Save it to `data/board_ai_governance_audit.csv` with `index=False`, and print a 3-bullet memo to the board risk committee: finding, evidence, and requested decision.
- *Success Check:* the record shows `Group_A_Rate` `0.875`, `Group_B_Rate` `0.25`, `Disparate_Impact_Ratio` `0.286`, `Passes_Four_Fifths` `False`, and `Status` `Board review required`; the script ends with return code `0`.

13. Exercise 3: Advanced Practice - Can One Shared Threshold Fix the Gap?

- *Goal:* A different threshold for each group would be disparate treatment, so it is not an option. Test whether a **single threshold for everyone** can pass the four-fifths rule, and what it would cost in credit risk. Prepare the trade-off for the board.
- *What you need:*
  - File: `data/session_04_04_dataset.csv` (described in **5. Lab Practice Dataset**).
  - Script name: `YourName_S44_challenge.py`.
  - Tools you already know: your `approval_audit()` function from Exercise 2, `for` loops, lists of dictionaries, `pd.DataFrame()`, and Boolean masks.
- *Requirements (follow them in order):*
  1. For each shared threshold in `[0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85]`, compute: the number approved, the approval rate of each group, the disparate impact ratio (rounded to 3 decimals), and `Risky_Approvals`: approved applicants with a DTI above 35%, the lender's usual risk limit.
  2. Collect the results in one DataFrame with the columns `Threshold`, `Approved`, `Group_A_Rate`, `Group_B_Rate`, `DI_Ratio`, and `Risky_Approvals`.

3. Find the **highest** threshold that passes the four-fifths rule, and report its `Risky_Approvals`.
4. Find the **highest** threshold that passes the four-fifths rule **and** has `0` risky approvals. If there is none, print that clearly.
5. Write three comment lines for the board: what the threshold can and cannot fix, and two other actions the board could request instead.

- *New tools for this exercise:*
  - Combining two masks: `approved & (df["Debt_To_Income_Pct"] > 35)` is `True` for applicants who are approved **and** above the risk limit; `.sum()` counts them.
  - Filtering a results table with two conditions: `results[(results["DI_Ratio"] >= 0.80) & (results["Risky_Approvals"] == 0)]`. An empty result has `len()` of `0`.
  - Guarding a division: if a group has no approvals at all, its rate is `0`, and the ratio is `0`; check `rates.max() > 0` before dividing.
- *Hints:*
  - Reuse `approval_audit()` inside the loop, so every threshold is tested the same way.
  - Print the full table before answering requirements 3 and 4; the pattern is easier to see.
- *Self-Check (expected results):*
  - Ratios: `0.45` `1.000`, `0.50` `0.875`, `0.55` `0.625`, `0.60` `0.500`, `0.65` `0.250`, `0.70` `0.286`, and `0.000` from `0.75` upward.
  - Highest passing threshold: `0.50`, with 15 of 16 applicants approved and `3` risky approvals (DTI above 35%).
  - No threshold passes the rule with `0` risky approvals.
  - Business meaning: moving one threshold either fails the fairness test or approves applicants above the lender's risk limit. The threshold cannot fix the gap, because the gap comes from the features themselves. The board should instead ask for an investigation of the features (do credit scores and DTI measure real repayment risk equally well for both groups?), consider additional data such as rent and utility payment history, and require human review of rejected applications near the threshold, with every decision documented.