About Us

Comparing Team Capabilities to Market Demand in Data & Analytics Roles

Dinara Zhorabek

Hung Nguyen

Nhat Tran

December 12, 2025

---

title: “Home”

Overview

This website presents a structured analysis of the 2024 U.S. job market using a combination of **Lightcast job postings**, **FRED macroeconomic indicators**, and insights from academic research. Our goal is to understand how economic conditions, skill demand, and organizational behavior shape opportunities for job seekers entering the market in 2024.

The analysis integrates quantitative data (job postings, salaries, geographic distribution, required skills) with qualitative insights drawn from peer-reviewed literature on hiring dynamics, gender disparities, and political context. By combining these perspectives, we provide a more holistic and actionable understanding of labor-market patterns.

Data Sources

Our project leverages two primary sources:

Lightcast Job Postings (2024)

This dataset includes thousands of job postings across the United States, containing information on:

- job titles
- employer details

- required skills
- salaries where reported
- geographic location
- posting dates

After cleaning and standardization, this dataset serves as the backbone of our empirical analysis, allowing exploration of hiring demand, wage patterns, and regional variations.

FRED Macroeconomic Indicators

To contextualize shifts in hiring, we incorporate macroeconomic time series from the Federal Reserve, including:

- unemployment rates
- labor-force participation
- CPI and inflation measures
- sector-level economic activity

These indicators help clarify why specific industries expand or contract over time.

Why It Matters

Navigating the job market has become increasingly complex for recent graduates and career-switchers. Economic uncertainty, rapid technological development, and evolving skill requirements mean that job seekers must adapt to conditions that shift year to year.

Three guiding questions shape our analysis:

1. **Which skills are most in demand in 2024, and how have they evolved?**
2. **How do geographic and sectoral differences influence job opportunities and salary outcomes?**
3. **How do broader social factors, such as gender representation and organizational culture affect hiring?**

These questions connect quantitative job-posting data with research on labor inequality, organizational behavior, and workforce development.

Website Structure

This website synthesizes findings across several analytical components:

- **Research Background**

  Academic perspectives on gender disparities, labor dynamics, and political influences.

- **Analysis**

  - **Gender Dispartities Overview**
  - **Data Cleaning & Preparation** : Standardizing, restructuring, and validating Lightcast job postings.
  - **Exploratory Data Analysis (EDA)** : Hiring patterns, salary distributions, geographic variation, and skills demand.
  - **Gender Dominance in Job Postings** :
  - **Machine Learning Models** : Clustering and predictive modeling of job-market patterns.
  - **NLP Analysis** : Extracting insights from job descriptions, including skills and thematic patterns.
  - **Skill Gap Analysis** : Comparison between team skill sets and market expectations.
- **Career Strategy**
- **About Us**

Collectively, these analyses provide a comprehensive, data-driven picture of the 2024 job market and inform actionable career strategies.

---

title: “Gender Disparities in Hiring & Political Influence” authors: “Hung Nguyen, Nhat Tran, Dinara Zhorabek” date: today bibliography: references.bib csl: csl/econometrica.csl format: html: toc-depth: 3

Introduction

The following section provides a research-based overview of gender disparities in hiring and leadership across academic, organizational, and political contexts. These insights serve as the conceptual foundation for our broader job-market analysis, helping us connect large-scale labor trends with the social and structural dynamics that influence real hiring outcomes. Without altering the original findings, this introduction contextualizes

why an understanding of gendered patterns remains crucial for interpreting hiring behaviors and market inequalities in 2024.

Gender disparity in academic positions and STEM discipline

Gender disparities go beyond who holds positions of power, raising deeper questions about the value and compensation assigned to women's work. In academic STEM disciplines, these pay inequalities occur alongside ongoing underrepresentation, which becomes more pronounced at senior levels. According to the research conducted by (Danielle J. Galvin ,Susan C. Anderson ,Chelsi J. Marolf ,Nikole G. Schneider ,Andrea L. Liebl (2024)), of over 20,000 faculty positions across 127 universities revealed that while women constitute approximately 50% of assistant professors in STEM fields, their representation declines significantly to 45.4% at the associate level and just 32.8% among full professors. Geographic variations are noticeable: though the Pacific and East North Central regions show the highest levels of gender representation, yet women in these areas account for only about 25% of tenure-track positions. Whereas mountain region showed visibly lower levels at just 19%. These regional differences seem to reflect the influence of state-level policies such as equal pay laws and paid family leave, as well as local cultural norms and institutional support systems.

These inequality patterns closely tied to the "leaky pipeline" effect, whereas women are more likely to leave STEM careers over time as barriers and disadvantages accumulate throughout their career path. When women are hired into STEM positions, they often receive smaller initial packages than male colleagues, placing them at an immediate disadvantage in launching their researches. Once employed, women often take on a heavier load of mentoring and service work without extra pay, tedious and repetitive tasks that are frequently overlooked in tenure evaluations. This imbalance may persist because women are often viewed as "caretakers," while men are seen as more "task-driven." Particularly, states with stronger equal pay laws and better comprehensive family-leave policies, including California, Massachusetts, Oregon, and Washington, illustrate smaller gaps in both representation and compensation. Understanding how wage disparities intersect with structural barriers and cultural perceptions is essential for addressing why qualified women continue to face obstacles in attaining leadership roles and receive recognition that matches their contributions.

Gender stereotypes in organizational structure

Hiring women into top leadership roles, like CEOs or board members, can change the way organizations talk about women (Lawson et al. (2022)). By analyzing over 43,000 company documents with advanced language tools, the researchers found that when women are appointed to senior positions, the language used by those organizations starts to describe women with more leadership-related traits such as confidence and decisiveness. Importantly, this shift doesn't reduce how often women are described as kind or caring, it simply expands how they are portrayed. In other words, bringing women into leadership helps reshape how society links "being a woman" with "being a leader."

The research is based on the idea that women's underrepresentation in leadership comes partly from language and stereotypes that associate men with power and women with warmth. Previous solutions like diversity training often have short-term results. This study suggests that a more lasting change can happen through representation itself: when women hold visible positions of power, they naturally help transform the language, attitudes, and stereotypes that have limited them in the first place.

Specific case: A newly selected Female Prime Minister in Japan

Although gender representation in leadership has improved in recent decades, perceptions of authority often remain unequal. Research consistently shows that women and men can perform equally well as leaders, yet they are judged differently because of deeply rooted social expectations. (Eagly and Karau (2002))'s role congruity theory suggests that society continues to associate effective leadership with traits that are traditionally viewed as masculine, such as assertiveness, decisiveness, and authority. As a result, when women display these same qualities, they are often perceived as competent but less likable or overly dominant. A recent experiment published in (Schneider, Altmann, and Langen (2023)) also found that identical leadership decisions were rated as more legitimate when made by men than by women, even after controlling for all other variables. Taken together, these findings highlight a subtle yet persistent perception gap in which authority remains unconsciously linked to gendered expectations. This perception gap helps explain why women, despite achieving comparable results, still encounter skepticism and resistance in positions of power.

This bias is not confined to the workplace but extends into politics and global governance. In 2025, Japan elected Sanae Takaichi as its first female prime minister, marking a historic moment for a country that has long struggled with gender equality in leadership. As reported by (TIME Magazine (2025)), public and media reactions reflected both celebration and hesitation, combining admiration for this milestone with ongoing scrutiny of her leadership style and decision-making approach. Commentators frequently described her potential leadership through gendered narratives, questioning whether she would govern with "feminine sensitivity" or "masculine toughness," expressions rarely used to evaluate male leaders. This reaction illustrates how even moments of progress can be accompanied by persistent stereotypes that frame women's authority through outdated cultural norms. Ultimately, addressing this issue requires more than improving access to leadership opportunities; it calls for a broader cultural shift in how societies define and value leadership, recognizing competence, vision, and authority as qualities that transcend gender.

Conclusion

Together, these three perspectives highlight how gender disparities emerge and persist across multiple domains of public life. They demonstrate that inequality is reinforced not only by structural and economic barriers but also by cultural narratives and expectations that shape how leaders are perceived. This research background provides the conceptual

framing that informs our later analysis of job-market data, helping us connect macro-level labor trends with the deeper social patterns that influence hiring, advancement, and representation in 2024.

–>

---

title: “Data Cleaning & Pre-Processing” subtitle: “Preparing and Standardizing Lightcast Job Postings Data for Analysis” bibliography: references.bib csl: csl/econometrica.csl format: html: code-fold: true code-tools: true

---

Objective

Data cleaning is the most critical phase in any analytics workflow.
Following the AD688 project requirements, our goals for this section are:

- inspect the raw dataset structure,
- identify and remove redundant or outdated columns,
- fix inconsistent formatting (e.g., stringified list fields),
- **standardize salary fields and preserve them even when sparsely reported**,
- standardize datetime fields and create analysis-ready variables,
- handle missing values using appropriate statistical strategies,
- remove duplicate job postings,
- export a clean, consistent dataset used across EDA, ML, and NLP sections.

This ensures accuracy, reproducibility, and coherence across the entire site.

Initial Dataset Overview

```
import pandas as pd
import numpy as np
import ast
import re

# Load raw Lightcast job postings
```

```
df = pd.read_csv("data/lightcast_job_postings.csv", low_memory=False)

# Basic structural overview
# df.head()
# df.info()
# list(df.columns)
```

Dropping Redundant or Outdated Columns

Lightcast contains multiple versions of NAICS/SOC classifications, plus metadata fields (tracking URLs, flags, timestamps). We remove clearly redundant metadata and older classification versions while keeping the 2022 NAICS codes and 2021 SOC codes.

```
columns_to_drop = [
    "ID", "URL", "ACTIVE_URLS",
    "DUPLICATES", "LAST_UPDATED_TIMESTAMP", "NAICS2",
    "NAICS3", "NAICS4", "NAICS5", "NAICS6",
    "SOC_2", "SOC_3", "SOC_5"
]

df.drop(
    columns=columns_to_drop,
    inplace=True,
    errors="ignore"
    )
# df.head()
```

Cleaning Messy List-Type Columns

Many Lightcast columns store lists as text such as:

[“Job Board”] [“disabledperson.com”, “dejobs.org”]

So, we convert these into clean comma-separated strings.

```
def clean_list(x):
    """
    Convert stringified Python lists into comma-separated strings and strip
noise.
    If parsing fails, fall back to a lightly cleaned string.
    """
    if pd.isna(x):
        return ""
    x = str(x).replace("\n", " ").strip()
    try:
        obj = ast.literal_eval(x)
```

```
        if isinstance(obj, list):
            return ", ".join(str(i).strip() for i in obj)
    except Exception:
        pass
    # Fallback: remove stray quotes if present
    x = re.sub(r'["\']', "", x)
    return x.strip()

list_cols = [
    "SOURCE_TYPES", "SOURCES",
    "SKILLS_NAME", "SPECIALIZED_SKILLS_NAME",
    "COMMON_SKILLS_NAME", "SOFTWARE_SKILLS_NAME",
    "CERTIFICATIONS_NAME"
]

for col in list_cols:
    if col in df.columns:
        df[col] = df[col].apply(clean_list)

# display(df.head(3))
```

Salary Fields: Cleaning and Preservation

Salary information is central to our job-market analysis, so we explicitly clean and preserve all salary-related fields, even when they are sparsely populated.

We:

```
•   identify key salary-like columns: SALARY_FROM, SALARY_TO, SALARY,
ORIGINAL_PAY_PERIOD,
•   strip currency symbols, commas, and non-numeric characters from numeric
fields,
•   create a standardized PAY_PERIOD,
•   convert everything to an annualized ANNUAL_SALARY field suitable for EDA
and modeling.
```

```
# Key salary-related columns we want to preserve
salary_cols = [
    "SALARY_FROM",
    "SALARY_TO",
    "SALARY",
    "ORIGINAL_PAY_PERIOD"
    ]

# Ensure the columns exist
for col in salary_cols:
    if col not in df.columns:
        df[col] = np.nan
```

```
# Helper to clean salary values, including ranges like "90k-120k"
def clean_salary_range(x):
    if pd.isna(x):
        return np.nan
    x = str(x)

    # Keep digits, commas, dot, minus, and en-dash
    x = re.sub(r"[^\d,\.\-\–]", "", x)

    # If there is a range, split and average
    if "-" in x or "–" in x:
        parts = re.split(r"[--]", x)
        nums = []
        for p in parts:
            p = p.replace(",", "")
            if p == "":
                continue
            try:
                nums.append(float(p))
            except Exception:
                pass
        if len(nums) >= 2:
            return np.mean(nums)
        elif len(nums) == 1:
            return nums[0]
        return np.nan

    # Otherwise, treat as a single value
    x = x.replace(",", "")
    if x == "":
        return np.nan
    try:
        return float(x)
    except Exception:
        return np.nan

# Clean numeric salary fields (FROM, TO, SALARY)
numeric_salary_fields = [
    "SALARY_FROM", "SALARY_TO", "SALARY"
    ]

for col in numeric_salary_fields:
    df[col] = df[col].apply(clean_salary_range)

# Choose the best base salary for each row
def get_base_salary(row):
    # 1. If SALARY exists, treat it as already standardized
    if pd.notna(row["SALARY"]):
```

```
        return row["SALARY"]
    # 2. If both FROM and TO exist, average them
    if pd.notna(row["SALARY_FROM"]) and pd.notna(row["SALARY_TO"]):
        return (row["SALARY_FROM"] + row["SALARY_TO"]) / 2
    # 3. Fallback to whichever exists
    if pd.notna(row["SALARY_FROM"]):
        return row["SALARY_FROM"]
    if pd.notna(row["SALARY_TO"]):
        return row["SALARY_TO"]
    return np.nan

# Normalize pay period strings
def normalize_period(x):
    if pd.isna(x):
        return "year"
    x = str(x).lower()

    if any(k in x for k in ["year", "yr", "annual"]):
        return "year"
    if any(k in x for k in ["month", "mo", "per_month"]):
        return "month"
    if "biweek" in x:
        return "biweek"
    if any(k in x for k in ["week", "wk", "per_week"]):
        return "week"
    if any(k in x for k in ["day", "daily"]):
        return "day"
    if any(k in x for k in ["hour", "hr"]):
        return "hour"
    return "year"

df["PAY_PERIOD"] = df["ORIGINAL_PAY_PERIOD"].apply(normalize_period)

# Convert base salary to annual salary
def convert_to_annual(row):
    salary = get_base_salary(row)
    period = row["PAY_PERIOD"]

    if pd.isna(salary):
        return np.nan

    if period == "year":
        return salary
    if period == "month":
        return salary * 12
    if period == "week":
        return salary * 52
    if period == "biweek":
        return salary * 26
```

```
    if period == "day":
        return salary * 260   # approx. work days in a year
    if period == "hour":
        return salary * 2080  # 40 hours * 52 weeks
    return salary

df["ANNUAL_SALARY"] = df.apply(convert_to_annual, axis=1)

# Remove implausible values (e.g., < $10k or > $700k per year)
df.loc[df["ANNUAL_SALARY"] < 10000, "ANNUAL_SALARY"] = np.nan
df.loc[df["ANNUAL_SALARY"] > 700000, "ANNUAL_SALARY"] = np.nan

df["ANNUAL_SALARY"] = df["ANNUAL_SALARY"].astype(float)

df[["SALARY_FROM",
    "SALARY_TO",
    "SALARY",
    "PAY_PERIOD",
    "ANNUAL_SALARY"]
    ].head(3)
```

	SALARY_FROM	SALARY_TO	SALARY	PAY_PERIOD	ANNUAL_SALARY
0	NaN	NaN	NaN	year	NaN
1	NaN	NaN	NaN	year	NaN
2	NaN	NaN	NaN	year	NaN

Handling Missing Values

Understanding missingness is important for determining how to impute values and whether key fields are usable.

We focus on essential analytical fields:

- salary fields (now cleaned),
- experience requirements,
- location fields,
- duration.

We drop extremely sparse columns (more than 50% missing) but never drop salary-related fields, even if they exceed this threshold.

```
# Inspect the most-missing columns
missing_summary = df.isna().mean().sort_values(ascending=False).head(15)
# display(missing_summary)

# Critical columns that must be preserved
critical_cols = [
    c for c in ([
```

```
        "SALARY_FROM",
        "SALARY_TO",
        "SALARY",
        "ORIGINAL_PAY_PERIOD",
        "ANNUAL_SALARY",
        "PAY_PERIOD"
        ])
    if c in df.columns
]

threshold = len(df) * 0.5

# Drop columns with >50% missing, except critical salary / pay-period fields
cols_to_drop_missing = [
    c for c in df.columns
    if c not in critical_cols and df[c].isna().sum() > threshold
]

df.drop(columns=cols_to_drop_missing, inplace=True)

# Fill numeric values with median (EXCLUDING salary-related fields)
num_cols = df.select_dtypes(include=["float", "int"]).columns
num_exclude = set(["SALARY_FROM", "SALARY_TO", "SALARY", "ANNUAL_SALARY"])
num_fill = [c for c in num_cols if c not in num_exclude]

df[num_fill] = df[num_fill].fillna(df[num_fill].median())

# Leave salary-related fields as NaN when missing

# Fill categorical values with "Unknown"
cat_cols = df.select_dtypes(include="object").columns
df[cat_cols] = df[cat_cols].fillna("Unknown")

print("Shape after missing-value handling:", df.shape)
# df.info()
```

```
Shape after missing-value handling: (72498, 114)
```

Standardizing the POSTED Date and Creating a Month Column

Datetime consistency is essential for trend analysis and for joining with FRED macroeconomic indicators.

```
# df["POSTED"] = pd.to_datetime(df["POSTED"], errors="coerce")
# df["month"] = df["POSTED"].dt.to_period("M").astype(str)
df = df.assign(
    POSTED = pd.to_datetime(df["POSTED"], errors="coerce"),
```

```
    month = lambda x: x["POSTED"].dt.to_period("M").astype(str)
)
display(df[["POSTED", "month"]].head())
```

	POSTED	month
0	2024-06-02	2024-06
1	2024-06-02	2024-06
2	2024-06-02	2024-06
3	2024-06-02	2024-06
4	2024-06-02	2024-06

Removing Duplicate Job Postings

Duplicate postings can overcount job demand and distort salary statistics. We remove duplicates based on:

- job title
- company
- location
- original posting date

```
subset_cols = [
    c for c in [
        "TITLE_NAME",
        "COMPANY_NAME",
        "LOCATION",
        "POSTED"]
        if c in df.columns]

if subset_cols:
    before = df.shape[0]
    df = df.drop_duplicates(subset=subset_cols, keep="first")
    after = df.shape[0]
    print(f"Removed {before - after} duplicate postings.")
else:
    print("Duplicate removal skipped: key columns missing.")

print(df.shape)
```

```
Removed 3300 duplicate postings.
(69198, 115)
```

Exporting the Final Cleaned Dataset

Exporting the Final Cleaned Dataset

This cleaned dataset will be used consistently across:

- Exploratory Data Analysis (EDA),
- Skill Gap Analysis,
- ML Methods,
- NLP Methods.

Exporting ensures team alignment and prevents inconsistent results.

```
df.to_csv("data/lightcast_cleaned.csv", index=False)
# df.head()
print(df.shape)
```

```
(69198, 115)
```

Summary

This cleaned dataset establishes the foundation for all downstream analysis.

By explicitly cleaning and preserving salary fields, selectively dropping only non-essential sparse columns, standardizing dates, and removing duplicates, we ensure:

- accuracy and reproducibility,
- consistent salary-based insights,
- reduced rendering load for GitHub Pages,
- and smooth integration with the ML and NLP pipelines.

This completes the Data Cleaning & Pre-Processing phase.

EDA Overview

This page explores hiring trends, salary patterns, geographic variation, remote work availability, and software skill demands using the **cleaned Lightcast dataset** produced earlier.

Each visualization is selected to answer job seeker–focused questions:

- **Which industries are hiring the most?**
- **How do salaries vary across sectors?**

- **How common are remote roles?**
- **How has demand changed over time?**
- **Which software skills are most requested?**

Load the Cleaned Dataset

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import numpy as np
import re

df = pd.read_csv("data/lightcast_cleaned.csv", low_memory=False)
#df.head()
#df.info()
#list(df.columns)
```

Job Postings by Industry (Top Sectors Hiring)

```
industry_candidates = [
    "NAICS_2022_2_NAME",
    "INDUSTRY_NAME",
    "NAICS_2022_3_NAME"]
industry_col = next(
    (c for c in industry_candidates if c in df.columns),
    None)
industry_col
```

```
'NAICS_2022_2_NAME'
```

```
if industry_col:
    industry_counts = (
        df[industry_col]
        .value_counts()
        .head(15)
        .reset_index()
    )
    industry_counts.columns = ["Industry", "Postings"]

    fig = go.Figure()
    fig.add_trace(go.Bar(
        x=industry_counts["Postings"],
        y=industry_counts["Industry"],
        orientation="h"
    ))
```

```
    fig.update_layout(
        title=f"Top 15 Industries by Job Postings ({industry_col})",
        plot_bgcolor="white",
        margin=dict(l=10, r=10, t=60, b=10),
        height=500,
        xaxis=dict(
            showgrid=True,
            gridcolor="#eaeaea",
        ),
        yaxis=dict(
            categoryorder='total ascending',
        ))
    fig.show()

else:
    print("No industry column found.")
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

Insight:

Professional, Scientific & Technical Services and Administrative & Support Services account for the largest share of job postings, indicating sustained demand for analytical, operational, and client-facing roles. These sectors typically exhibit continuous hiring cycles driven by project-based work, business expansion, and comparatively high turnover. For job seekers, they represent strong entry points into the labor market with broad role variety and relatively frequent openings.

Annual Salary - Trend Over Time

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots

df2 = df.copy()
df2 = df2.dropna(subset=["ANNUAL_SALARY", "month"])

# Group by month (string)
agg = (
    df2.groupby("month")
    .agg(
        postings=("ANNUAL_SALARY", "size"),
        avg_salary=("ANNUAL_SALARY", "mean")
        ).reset_index().sort_values("month")
)

# Dual-axis chart
```

```
# fig = make_subplots(specs=[[{"secondary_y": True}]])
fig = go.Figure()
fig.add_trace(go.Bar(
    x=agg["month"],
    y=agg["postings"],
    name="Job Postings",
    marker_color="#ff7043",
    yaxis="y1",
    hovertemplate="Job postings: %{y:,}<extra></extra>"
))
fig.add_trace(go.Scatter(
    x=agg["month"],
    y=agg["avg_salary"],
    name="Annual Salary",
    mode="lines+markers",
    yaxis="y2",
    line=dict(color="#1f77b4", width=3),
    hovertemplate="Salary: $%{y:,.0f}<extra></extra>"
))

fig.update_layout(
    title="Annual Salary - Trend Over Time",
    template="plotly_white",
    hovermode="x unified",
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=1.02,
        xanchor="right",
        x=1),
    xaxis=dict(
        title="Month",
        tickformat="%b %Y",
        showgrid=True,
        gridcolor="#eaeaea"
    ),
    yaxis=dict(
        title="Job Postings",
        showgrid=True,
        gridcolor="#eaeaea"
    ),
    yaxis2=dict(
        title="Annual Salary($)",
        overlaying="y",
        side="right",
        tickprefix="$",
        showgrid=False
    )
)
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

Insight:

The chart shows a clear seasonal pattern in the job market. Job postings stay fairly strong from May through September, although there is a small dip in July before demand rises again in late summer. However, advertised salaries move very differently. They drop steadily from May to July and reach their lowest point mid-summer. Then, in September, salaries jump sharply to the highest level in the entire period.

This contrast suggests that early summer is dominated by lower-paying or junior roles, while late summer and early fall bring a return of higher-value positions. The most important takeaway is that September offers both strong hiring volume and significantly higher salaries, making it a particularly favorable time for job seekers aiming for better-paid opportunities.

Salary Distribution by Industry

Salary variation across industries answers: “Which sectors pay more, and how unequal are wages within each industry?” We use the annualized salary field (ANNUAL_SALARY), which consolidates raw salary data from SALARY_FROM, SALARY_TO, and SALARY and converts it into a consistent yearly format based on reported pay period. This provides a clean, comparable measure of compensation across all industries.

```
industry_col = industry_col
salary_col = "ANNUAL_SALARY" if "ANNUAL_SALARY" in df.columns else None

if industry_col and salary_col:

    # Filter valid salaries
    df_salary = df[[industry_col, salary_col]].dropna()

    # Remove noise categories that distort analysis
    df_salary = df_salary[df_salary[industry_col] != "Unclassified Industry"]

    # Cap extreme values at 300k for readability (still accurate for 95% of
postings)
    df_salary[salary_col] = np.where(
        df_salary[salary_col] > 300000,
        300000,
        df_salary[salary_col]
        )

    # Compute median salary per industry for ordering
    medians = (
        df_salary.groupby(industry_col)[salary_col]
        .median()
        .sort_values(ascending=True)
```

```
    )

    ordered_categories = medians.index.tolist()

    fig = px.box(
        df_salary,
        y=industry_col,
        x=salary_col,
        category_orders={industry_col: ordered_categories},
        title="Annual Salary Distribution by Industry",
        color=industry_col,
        color_discrete_sequence=px.colors.qualitative.Vivid,
        height=900
    )

    fig.update_layout(
        showlegend=False,
        xaxis_title="Annual Salary (USD)",
        yaxis_title="Industry",
        margin=dict(l=120)
    )

    fig.show()

else:
    print("Salary chart not generated: Missing industry or salary column.")
```

```
Unable to display output for mime type(s): text/html
```

Insight:

Annual salary levels differ substantially across industries, with Finance, Information, and Professional Services exhibiting the highest median compensation. The wide dispersion of salaries within these sectors reflects variation in seniority, specialization, and credential requirements. In contrast, service-oriented industries such as Retail and Accommodation & Food Services show lower but more compressed salary ranges, consistent with narrower variation in role complexity. These patterns suggest that targeted upskilling toward high-skill roles in Finance, Tech, and Professional Services is likely to yield the greatest earnings uplift.

Remote vs. On-Site Job Patterns

Remote work availability is a crucial factor for job seekers.
This chart shows whether industries lean toward remote, hybrid, or in-person work. Here, [None] represents postings where Lightcast did not provide a remote-work tag, which typically corresponds to on-site or unspecified arrangements.

```
remote_candidates = ["REMOTE_TYPE_NAME", "REMOTE_TYPE"]
remote_col = next((c for c in remote_candidates if c in df.columns), None)
```

```
if remote_col:
    remote_counts = df[remote_col].value_counts().reset_index()
    remote_counts.columns = ["Work Arrangement", "Postings"]

    fig = px.pie(
        remote_counts,
        names="Work Arrangement",
        values="Postings",
        title="Remote vs On-Site Job Postings",
        color_discrete_sequence=px.colors.qualitative.Pastel1
    )
    fig.show()
else:
    print("No remote work field found.")
```

```
Unable to display output for mime type(s): text/html
```

Insight:

The majority of postings either do not explicitly tag remote work or are implicitly on-site, with only a modest share labeled as Remote or Hybrid. Where remote options are available, they are concentrated in data, IT, and knowledge-intensive roles that can be performed digitally. For job seekers prioritizing location flexibility, this suggests focusing on analytically oriented roles and employers that have formal remote-work policies rather than assuming remote options are widespread across all occupations.

Geographic Distribution of Jobs (by State)

Geographic EDA answers: “Which states have the most job opportunities?” We use STATE_NAME (cleaned) to compute the total number of postings per state.

```
STATE_CENTROIDS = {
    "AL": (32.806, -86.791), "AK": (64.200, -149.493), "AZ": (34.049, -
111.094),
    "AR": (35.201, -91.832), "CA": (36.778, -119.418), "CO": (39.550, -
105.782),
    "CT": (41.603, -73.087), "DE": (38.910, -75.527), "DC": (38.907, -
77.037),
    "FL": (27.664, -81.516), "GA": (32.165, -82.900), "HI": (19.896, -
155.582),
    "ID": (44.068, -114.742), "IL": (40.633, -89.398), "IN": (40.267, -
86.134),
    "IA": (41.878, -93.098), "KS": (39.012, -98.484), "KY": (37.839, -
84.270),
    "LA": (30.984, -91.963), "ME": (45.254, -69.445), "MD": (39.045, -
76.641),
    "MA": (42.407, -71.382), "MI": (44.314, -85.602), "MN": (46.729, -
94.685),
    "MS": (32.355, -89.398), "MO": (37.964, -91.832), "MT": (46.879, -
110.362),
```

```
    "NE": (41.492, -99.901), "NV": (38.803, -116.419), "NH": (43.193, -
71.572),
    "NJ": (40.058, -74.406), "NM": (34.519, -105.870), "NY": (43.000, -
75.000),
    "NC": (35.759, -79.019), "ND": (47.551, -101.002), "OH": (40.417, -
82.907),
    "OK": (35.468, -97.516), "OR": (43.804, -120.554), "PA": (41.203, -
77.194),
    "RI": (41.580, -71.477), "SC": (33.837, -81.164), "SD": (43.969, -
99.901),
    "TN": (35.518, -86.580), "TX": (31.968, -99.901), "UT": (39.321, -
111.094),
    "VT": (44.558, -72.577), "VA": (37.431, -78.657), "WA": (47.751, -
120.740),
    "WV": (38.597, -80.454), "WI": (43.785, -88.787), "WY": (43.076, -
107.290)
}
STATE_NAMES = {
    "AL": "Alabama", "AK": "Alaska", "AZ": "Arizona",
    "AR": "Arkansas", "CA": "California", "CO": "Colorado",
    "CT": "Connecticut", "DE": "Delaware",
    "DC": "District of Columbia", "FL": "Florida",
    "GA": "Georgia", "HI": "Hawaii", "ID": "Idaho",
    "IL": "Illinois", "IN": "Indiana", "IA": "Iowa",
    "KS": "Kansas", "KY": "Kentucky", "LA": "Louisiana",
    "ME": "Maine", "MD": "Maryland", "MA": "Massachusetts",
    "MI": "Michigan", "MN": "Minnesota", "MS": "Mississippi",
    "MO": "Missouri", "MT": "Montana", "NE": "Nebraska",
    "NV": "Nevada", "NH": "New Hampshire", "NJ": "New Jersey",
    "NM": "New Mexico", "NY": "New York", "NC": "North Carolina",
    "ND": "North Dakota", "OH": "Ohio", "OK": "Oklahoma",
    "OR": "Oregon", "PA": "Pennsylvania", "RI": "Rhode Island",
    "SC": "South Carolina", "SD": "South Dakota", "TN": "Tennessee",
    "TX": "Texas", "UT": "Utah", "VT": "Vermont",
    "VA": "Virginia", "WA": "Washington", "WV": "West Virginia",
    "WI": "Wisconsin", "WY": "Wyoming"
}
STATE_NAME_TO_CODE = {
    "Alabama": "AL", "Alaska": "AK", "Arizona": "AZ", "Arkansas": "AR",
    "California": "CA", "Colorado": "CO", "Connecticut": "CT", "Delaware":
"DE",
    "District of Columbia": "DC", "Florida": "FL", "Georgia": "GA",
    "Hawaii": "HI", "Idaho": "ID", "Illinois": "IL", "Indiana": "IN",
    "Iowa": "IA", "Kansas": "KS", "Kentucky": "KY", "Louisiana": "LA",
    "Maine": "ME", "Maryland": "MD", "Massachusetts": "MA", "Michigan": "MI",
    "Minnesota": "MN", "Mississippi": "MS", "Missouri": "MO", "Montana":
"MT",
    "Nebraska": "NE", "Nevada": "NV", "New Hampshire": "NH", "New Jersey":
"NJ",
    "New Mexico": "NM", "New York": "NY", "North Carolina": "NC",
```

```
    "North Dakota": "ND", "Ohio": "OH", "Oklahoma": "OK", "Oregon": "OR",
    "Pennsylvania": "PA", "Rhode Island": "RI", "South Carolina": "SC",
    "South Dakota": "SD", "Tennessee": "TN", "Texas": "TX", "Utah": "UT",
    "Vermont": "VT", "Virginia": "VA", "Washington": "WA",
    "West Virginia": "WV", "Wisconsin": "WI", "Wyoming": "WY"
}

df = df[~df["CITY_NAME"].str.contains("Unknown", na=False)].copy()
df_city = (
    df
    .dropna(subset=["CITY_NAME", "STATE_NAME"])
    .groupby(["CITY_NAME", "STATE_NAME"], as_index=False)
    .agg(total_job_postings=("CITY_NAME", "size"))
)
df_city["STATE_CODE"] = df_city["STATE_NAME"].map(STATE_NAME_TO_CODE)
df_city = df_city.dropna(subset=["STATE_CODE"])
df_city = df_city.nlargest(100, "total_job_postings")

df_state = (
    df_city.groupby("STATE_CODE", as_index=False)
           .agg(total_postings=("total_job_postings", "sum"))
)
# city coordinates
lats, lons = [], []
counts = df_city.groupby("STATE_CODE")["CITY_NAME"].transform("count")
idxs = df_city.groupby("STATE_CODE").cumcount()

for st_code, k, idx in zip(df_city["STATE_CODE"], counts, idxs):
    base = STATE_CENTROIDS.get(st_code, (37.0, -96.0))  # fallback center of
US
    angle = 2 * np.pi * (idx / max(k, 1))
    radius = 0.8

    lats.append(base[0] + radius * np.cos(angle))
    lons.append(base[1] + radius * np.sin(angle))

df_city["lat"] = lats
df_city["lon"] = lons

# Bubble size scaling
max_total = float(df_city["total_job_postings"].max())
size_scale = 40.0 / np.sqrt(max_total) if max_total > 0 else 1.0

fig = go.Figure()
fig.add_trace(go.Choropleth(
    locations=df_state["STATE_CODE"],
    z=df_state["total_postings"],
    locationmode="USA-states",
    colorscale="Blues",
```

```
    marker_line_color="white",
    marker_line_width=0.6,
    colorbar_title="Total postings",
    hovertemplate="<b>%{location}</b><br>Total: %{z:,}<extra></extra>",
    zmin=0,
    zmax=df_state["total_postings"].max()
))

# City bubbles
fig.add_trace(go.Scattergeo(
    lon=df_city["lon"],
    lat=df_city["lat"],
    text=df_city["CITY_NAME"],
    mode="markers",
    marker=dict(
        size=np.sqrt(df_city["total_job_postings"]) * size_scale,
        sizemin=5,
        opacity=0.85,
        color="#22c55e",
        line=dict(width=0.8, color="white")
    ),
    customdata=df_city["total_job_postings"],
    hovertemplate=(
        "<b>%{text}</b><br>"
        "Total postings: %{customdata:,}"
        "<extra></extra>"
    ),
    showlegend=False,
    name="City postings"
))

# State labels
label_lats, label_lons, label_names = [], [], []
for code in df_state["STATE_CODE"]:
    if code in STATE_CENTROIDS:
        lat, lon = STATE_CENTROIDS[code]
        label_lats.append(lat)
        label_lons.append(lon)
        label_names.append(STATE_NAMES.get(code, code))

fig.add_trace(go.Scattergeo(
    lat=label_lats,
    lon=label_lons,
    mode="text",
    text=label_names,
    textfont=dict(size=9, color="rgba(0,0,0,0.75)"),
    hoverinfo="skip",
    showlegend=False
))
```

```
fig.update_layout(
    title="Job postings by Location",
    geo=dict(
        scope="usa",
        projection=go.layout.geo.Projection(type="albers usa"),
        showland=True,
        landcolor="#fafafa"
    ),
    # margin=dict(l=100, r=20, t=60, b=60),
    paper_bgcolor="white",
    plot_bgcolor="white"
)
fig.update_traces(
    selector=dict(type="choropleth"),
    colorbar=dict(
        orientation="h",
        x=0.5,
        xanchor="center",
        y=-0.01,
        yanchor="top",
        thickness=12,
        len=0.6
    )
)
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

Insight:

Texas, California, and Florida emerge as national hiring hubs, reflecting their large and diversified economies, population growth, and strong business ecosystems. Many of the remaining top states, such as Virginia, New York, New Jersey, and North Carolina are anchored by federal agencies, financial centers, or technology clusters. Concentrating job search efforts in these high-volume states can materially increase the number of suitable opportunities available to candidates.

Top Software Skills in Job Postings

To support the Skill Gap Analysis page, we identify which software tools are requested most often. This uses the cleaned SOFTWARE_SKILLS_NAME column produced in the Data Cleaning step.

```
if "SOFTWARE_SKILLS_NAME" in df.columns:
    skills = (
        df["SOFTWARE_SKILLS_NAME"]
        .astype(str)
        .str.split(",")
        .explode()
```

```
        .str.strip()
    )
    skills = skills[(skills != "") & (skills != "nan")]

    top_skills = skills.value_counts().head(15).reset_index()
    top_skills.columns = ["Skill", "Postings"]

    fig = go.Figure()
    fig.add_trace(go.Bar(
        x=top_skills["Postings"],
        y=top_skills["Skill"],
        orientation="h"
    ))
    fig.update_layout(
        title="Top 15 Software Skills",
        plot_bgcolor="white",
        margin=dict(l=10, r=10, t=60, b=10),
        height=500,
        xaxis=dict(
            showgrid=True,
            gridcolor="#eaeaea",
        ),
        yaxis=dict(
            categoryorder='total ascending',
        ),
        legend=dict(
            orientation="h",
            yanchor="bottom",
            y=1.02,
            xanchor="right",
            x=1
        ))
    fig.show()
else:
    print("Software skills column not available.")
```

```
Unable to display output for mime type(s): text/html
```

Insight:

SQL, Python, and Excel appear as the most universally requested tools, underscoring their status as core competencies for analytics, BI, and operations roles. Tableau and Power BI, together with dashboarding skills more broadly, highlight the premium placed on communicating data insights visually. Cloud and platform skills such as AWS, Azure, and Oracle Cloud are less frequent but still prominent, indicating growing demand for candidates who can work across both traditional analytics stacks and modern cloud-based architectures.

EDA Summary

The exploratory analysis reveals five consistent themes in the 2024 job market:

- **Industry concentration**: Hiring is heavily concentrated in Professional Services, Administrative & Support Services, and key service sectors, which together generate a large share of postings.
- **Compensation differences**: Median annual salaries vary sharply by industry, with Finance, Information, and Professional Services offering the strongest earnings potential.
- **Remote work structure**: Explicitly remote and hybrid roles remain a minority but are disproportionately found in digital and analytics-intensive occupations.
- **Seasonality**: Job posting activity dips around late June and rebounds sharply in late summer, highlighting the importance of timing in job search strategies.
- **Skill demand**: SQL, Python, Excel, BI tools, and cloud platforms dominate technical requirements, defining a clear "baseline stack" for aspiring data and business analytics professionals.

These insights directly feed into the subsequent Skill Gap Analysis, ML Methods, and Career Strategy sections by identifying which industries, skills, and time windows matter most for job seekers in 2024.

---

title: "Gender Dominance Analysis" subtitle: "Distribution of Job Postings and Median Salaries Across Gender-Dominance Categories" bibliography: references.bib csl: csl/econometrica.csl format: html: code-fold: true # code-tools: true

---

Objective

This section aims to clean and prepare the Lightcast job postings dataset, focusing on industries with clearly defined classifications. It then categorizes industries by gender dominance and analyzes job postings, median salaries, and distributions across male-, female-, and mixed-dominated sectors. The visualizations provide insights into how gender dominance correlates with job availability and compensation patterns.

Load the Cleaned Dataset

```
import pandas as pd
import numpy as np
import plotly.graph_objects as go
import plotly.io as pio
import plotly.express as px
```

```
import os

# Load raw Lightcast job postings
df = pd.read_csv("data/lightcast_cleaned.csv", low_memory=False)
#df.head()
```

Industry Data Preparation

We clean the dataset by removing rows where the industry classification is either "Unknown" or "Unclassified Industry" to ensure the analysis focuses on clearly defined sectors. Before dropping these records, we calculate the percentage they represent in the dataset to document their impact on data quality.

```
# Clean the dataset to analyze available industries
# Remove rows where NAICS2_NAME is "Unknown"
df = df[df["NAICS2_NAME"] != "Unknown"]

# Use unclassified_count function to see % of Unclassified Industry in the
dataset
# Count rows before dropping
before_count = len(df)

# Count how many are "Unclassified Industry"
unclassified_count = (df["NAICS2_NAME"] == "Unclassified Industry").sum()

# Percentage of Unclassified Industry rows
percentage = unclassified_count / before_count * 100
print(f"Percentage of Unclassified Industry rows: {percentage:.2f}%")
# Drop those rows
df = df[df["NAICS2_NAME"] != "Unclassified Industry"]

after_count = len(df)
removed_count = before_count - after_count
print("Rows removed:", removed_count)
print("Rows before:", before_count)
print("Rows after:", after_count)
```

```
Percentage of Unclassified Industry rows: 13.31%
Rows removed: 9205
Rows before: 69181
Rows after: 59976
```

Exploring Unique Industry Categories (NAICS2_NAME)

We retrieve the list of unique industry categories by extracting all distinct values from the NAICS2_NAME column after removing missing entries.

```
# Get list of uniques NAICS2_NAME
naics2_name = df["NAICS2_NAME"].dropna().unique().tolist()
# print(naics2_name)
```

Classifying Industries by Gender Dominance

We classify each NAICS2_NAME industry into one of three gender-dominance categories based on U.S. labor statistics:

- **Male-dominated industries:** Agriculture, Forestry, Fishing and Hunting; Mining, Quarrying, and Oil and Gas Extraction; Utilities; Construction; Manufacturing; Wholesale Trade; Transportation and Warehousing; Professional, Scientific, and Technical Services; Information; Management of Companies and Enterprises.
- **Female-dominated industries:** Finance and Insurance; Educational Services; Health Care and Social Assistance; Accommodation and Food Services; Other Services (except Public Administration).
- **Mixed industries:** Retail Trade; Administrative and Support and Waste Management and Remediation Services; Real Estate and Rental and Leasing; Arts, Entertainment, and Recreation; Public Administration.

```
gender_dom_map = {
    # Male-dominated Industries
    "Agriculture, Forestry, Fishing and Hunting": "Male-dominated",
    "Mining, Quarrying, and Oil and Gas Extraction": "Male-dominated",
    "Utilities": "Male-dominated",
    "Construction": "Male-dominated",
    "Manufacturing": "Male-dominated",
    "Wholesale Trade": "Male-dominated",
    "Transportation and Warehousing": "Male-dominated",
    "Professional, Scientific, and Technical Services": "Male-dominated",
    "Information": "Male-dominated",
    "Management of Companies and Enterprises": "Male-dominated",

    # Female-dominated Industries
    "Finance and Insurance": "Female-dominated",
    "Educational Services": "Female-dominated",
    "Health Care and Social Assistance": "Female-dominated",
    "Accommodation and Food Services": "Female-dominated",
    "Other Services (except Public Administration)": "Female-dominated",

    # Mixed Industries
    "Retail Trade": "Mixed",
    "Administrative and Support and Waste Management and Remediation
Services": "Mixed",
    "Real Estate and Rental and Leasing": "Mixed",
    "Arts, Entertainment, and Recreation": "Mixed",
    "Public Administration": "Mixed"
}

# Create a GENDER_DOMINANCE column in the dataset
df["GENDER_DOMINANCE"] = df["NAICS2_NAME"].map(gender_dom_map)
```

```
gender_dom_summary = (
    df["GENDER_DOMINANCE"]
    .value_counts()
    .reset_index()
)
print("Total Job Posting Counts in Dataset:", len(df))
print("Total Job Posting Counts by Gender Dominance:")
display(gender_dom_summary)
```

```
Total Job Posting Counts in Dataset: 59976
Total Job Posting Counts by Gender Dominance:
```

	GENDER_DOMINANCE	count
0	Male-dominated	35159
1	Female-dominated	13013
2	Mixed	11804

Out of 59,976 total job postings, the dataset shows 35,159 job postings in male-dominated industries, followed by 13,013 postings in female-dominated industries, while Mixed industries account for 11,804 postings. These counts provide a clear view of how job postings are distributed across the three gender-dominance categories. The larger volume in male-dominated fields suggests that most hiring activity in this dataset is concentrated in those sectors, with smaller but still significant demand in female-dominated and Mixed industries. Together, this distribution gives a straightforward snapshot of where job opportunities are most active.

Median Salary by Industry with Job Count

```
# Drop rows with missing salary values
df_salary = df.dropna(subset=["SALARY"])
# Save new df as lighcast_gender.csv to local folder
df_salary.to_csv("./data/lightcast_gender.csv", index=False)

# Compute job counts
industry_counts = (df_salary.groupby("NAICS2_NAME")["SALARY"]
    .count()
    .rename("JOB_COUNT"))

industry_median_salary = (
    df_salary.groupby("NAICS2_NAME")["SALARY"]
    .median()
    .rename("MEDIAN_SALARY")
    .to_frame()
    .join(industry_counts)
    .sort_values("MEDIAN_SALARY", ascending=False)
    .reset_index()
)
display(industry_median_salary)
```

	NAICS2_NAME	MEDIAN_SALARY	JOB_COUNT
0	Accommodation and Food Services	144560.0	212
1	Information	132550.0	2166
2	Professional, Scientific, and Technical Services	130000.0	8530
3	Retail Trade	120000.0	755
4	Construction	118097.5	284
5	Manufacturing	117450.0	1552
6	Finance and Insurance	115239.0	3567
7	Utilities	114206.5	306
8	Wholesale Trade	101597.0	851
9	Management of Companies and Enterprises	101400.0	39
10	Mining, Quarrying, and Oil and Gas Extraction	100600.0	36
11	Administrative and Support and Waste Managemen...	98800.0	3584
12	Transportation and Warehousing	97739.0	203
13	Health Care and Social Assistance	95285.0	1280
14	Agriculture, Forestry, Fishing and Hunting	87500.0	28
15	Other Services (except Public Administration)	85000.0	332
16	Real Estate and Rental and Leasing	85000.0	416
17	Arts, Entertainment, and Recreation	82950.0	85
18	Public Administration	79092.0	691
19	Educational Services	75919.0	950

```
# Group by gender dominance
gender_dom_summary = (
    df_salary.groupby("GENDER_DOMINANCE")
    .agg(
        JOB_COUNT=("SALARY", "count"),
        MEDIAN_SALARY=("SALARY", "median")
    )
    .reset_index()
    .sort_values("MEDIAN_SALARY", ascending=False)
)
display(gender_dom_summary)
```

	GENDER_DOMINANCE	JOB_COUNT	MEDIAN_SALARY
1	Male-dominated	13995	125900.0
0	Female-dominated	6341	101500.0
2	Mixed	5531	95000.0

Chart 1: Median Salary by Gender Dominance Category

```
# Build a color palette for gender dominance
colors = {
    "Male-dominated": "rgba(137, 176, 255, 0.8)",
    "Female-dominated": "rgba(255, 179, 207, 0.8)",
    "Mixed": "rgba(199, 168, 255, 0.8)"
}

fig_gd = go.Figure()
fig_gd.add_trace(
    go.Bar(
        x=gender_dom_summary["GENDER_DOMINANCE"],
        y=gender_dom_summary["MEDIAN_SALARY"],
        text=[
            f"${v:,.0f}" for v in gender_dom_summary["MEDIAN_SALARY"]
            ],
        textposition="outside",
        marker_color=[
            colors[cat] for cat in gender_dom_summary["GENDER_DOMINANCE"]
            ]
    )
)
fig_gd.update_layout(
    title="Median Salary by Gender Dominance Category",
    yaxis_title="Median Salary",
    xaxis_title="Gender Dominance"
    # width=700,
    # height=500
)
fig_gd.show()
```

```
Unable to display output for mime type(s): text/html
```

Looking at the job postings, male-dominated industries lead in both number and pay, with 13,995 positions and a median salary of $125,900. Female-dominated sectors have 6,341 postings at a median of $101,500, while Mixed industries have the fewest openings, 5,531, with a median of $95,000. Overall, this points to a clear link between gender dominance in an industry and salary levels, with male-heavy fields offering both more opportunities and higher pay.

Chart 2: Salary Distribution by Gender Dominance

```
fig_box = go.Figure()
for category in df_salary["GENDER_DOMINANCE"].unique():
    fig_box.add_trace(
        go.Box(
            y=df_salary[
                df_salary["GENDER_DOMINANCE"] == category
                ]["SALARY"],
```

```
            name=category,
            marker_color=colors.get(category, "lightgray"),
            boxmean=True
        )
    )

fig_box.update_layout(
    title="Salary Distribution by Gender Dominance",
    yaxis_title="Salary",
    xaxis_title="Gender Dominance"
    # width=800,
    # height=550
)
fig_box.show()
```

```
Unable to display output for mime type(s): text/html
```

The boxplot shows that Male-dominated industries consistently offer higher salaries, with both a higher median and greater variation at the upper end. On the other hand, female-dominated industries display lower median salaries and a tighter distribution, suggesting fewer opportunities for better salary offers. The Mixed industries fall right in-between, with moderate median pay and some high-earning positions, though not as many as in male-dominated fields.

Chart 3: Job Count vs. Median Salary by Gender Dominance Category

```
bubble_fig = px.scatter(
    gender_dom_summary,
    x="JOB_COUNT",
    y="MEDIAN_SALARY",
    size="JOB_COUNT",
    color="GENDER_DOMINANCE",
    color_discrete_map=colors,
    hover_name="GENDER_DOMINANCE",
    size_max=80
)

bubble_fig.update_layout(
    title="Job Count vs. Median Salary by Gender Dominance Category",
    xaxis_title="Job Count",
    yaxis_title="Median Salary"
    # width=850,
    # height=550
)
bubble_fig.show()
```

```
Unable to display output for mime type(s): text/html
```

Male-dominated industries lead in both job count and median salary, with the largest number of openings and the highest pay. Female-dominated industries have fewer

positions and lower median salaries, while Mixed industries fall in between, offering moderate opportunities and pay. Overall, higher male representation in an industry is associated with more jobs and higher salaries.

Chart 4: Median Salary and Job Count Across Male-Dominated Industries

```
# Filter for male-dominated industries
male_df = df_salary[df_salary["GENDER_DOMINANCE"] == "Male-dominated"]

# Compute industry-level median salary
male_salary_summary = (
    male_df.groupby("NAICS2_NAME")["SALARY"]
       .median()
    .sort_values(ascending=False)
)

# Compute industry-level job counts
male_job_counts = (male_df.groupby("NAICS2_NAME")["SALARY"]
    .count()
    .reindex(male_salary_summary.index))

# Pastel blue for bars
pastel_blue = "rgba(137, 176, 255, 0.8)"
fig_male = go.Figure()
fig_male.add_trace(
    go.Bar(
        x=male_salary_summary.index,
        y=male_salary_summary.values,
        text=[
            f"${v:,.0f}" for v in male_salary_summary.values
            ],
        textposition="outside",
        marker_color=pastel_blue,
        name="Median Salary"
    )
)

# Add trend line for male-dominated industry job counts
fig_male.add_trace(
    go.Scatter(
        x=male_salary_summary.index,
        y=male_job_counts.values,
        mode="lines+markers",
        name="Job Count",
        yaxis="y2"
    )
)

fig_male.update_layout(
```

```
    title="Median Salary and Job Count Across Male-Dominated Industries",
    xaxis_title="Industry",
    yaxis_title="Median Salary",
    yaxis2=dict(
        title="Job Count",
        overlaying="y",
        side="right",
    ),
    xaxis_tickangle=40,
    # width=1100,
    height=700,
    margin=dict(l=50, r=70, t=80, b=150),
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=1.02,
        xanchor="right",
        x=1)
)
fig_male.show()
```

```
Unable to display output for mime type(s): text/html
```

The plot shows that male-dominated industries tend to offer relatively high median salaries, with Information ($132,550) and Professional/Scientific/Technical Services ($130,000) leading the group. However, job availability varies widely across some high-paying sectors: Information with a comparatively modest 2166 postings, while Professional/Scientific/Technical Services offers both high pay and large availability of 8530 postings. Thus, hands-on, strongly male-dominated fields like construction, mining, and agriculture show noticeably lower median salaries compared to many other industries overall, highlighting that physically intensive sectors don't necessarily correspond to higher earnings.

Chart 5: Median Salary and Job Count Across Female-Dominated Industries

```
# Filter for female-dominated industries
female_df = df_salary[df_salary["GENDER_DOMINANCE"] == "Female-dominated"]

# Compute industry-level median salary
female_salary_summary = (
    female_df.groupby("NAICS2_NAME")["SALARY"]
    .median()
    .sort_values(ascending=False)
)

# Compute industry-level job counts
female_job_counts = (female_df.groupby("NAICS2_NAME")["SALARY"]
    .count()
    .reindex(female_salary_summary.index))
```

```
# Pastel pink for bars
pastel_pink = "rgba(255, 179, 207, 0.8)"
fig_female = go.Figure()
fig_female.add_trace(
    go.Bar(
        x=female_salary_summary.index,
        y=female_salary_summary.values,
        text=[
            f"${v:,.0f}" for v in female_salary_summary.values
            ],
        textposition="outside",
        marker_color=pastel_pink,
        name="Median Salary"
    )
)

# Add trend line for female-dominated industry job counts
fig_female.add_trace(
    go.Scatter(
        x=female_salary_summary.index,
        y=female_job_counts.values,
        mode="lines+markers",
        name="Job Count",
        yaxis="y2"
    )
)

fig_female.update_layout(
    title="Median Salary and Job Count Across Female-Dominated Industries",
    xaxis_title="Industry",
    yaxis_title="Median Salary",
    yaxis2=dict(
        title="Job Count",
        overlaying="y",
        side="right",
    ),
    xaxis_tickangle=40,
    # width=1100,
    height=700,
    margin=dict(l=50, r=70, t=80, b=150),
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=1.02,
        xanchor="right",
        x=1)
)
fig_female.show()
```

```
Unable to display output for mime type(s): text/html
```

Female-dominated industries show a wider salary spread, ranging from about $76k to $145k, with Accommodation and Food Services unexpectedly offering the highest median pay ($144,560) despite only 212 job counts. Finance and Insurance stands out with strong salaries ($115,239) and by far the largest employment (3567), while Health Care and Social Assistance provides mid-range pay ($95,285) with substantial job availability (1280). Other Services and Educational Services offer lower salaries ($75,919) and moderate job counts (950), reflecting more service-oriented, lower-wage career tracks within this sector.

Chart 6: Median Salary and Job Count Across Mixed Industries

```
# Filter for Mixed industries
balanced_df = df_salary[df_salary["GENDER_DOMINANCE"] == "Mixed"]

# Compute industry-level median salary
balanced_salary_summary = (
    balanced_df.groupby("NAICS2_NAME")["SALARY"]
    .median()
    .sort_values(ascending=False)
)

# Compute industry-level job counts
balanced_job_counts = (balanced_df.groupby("NAICS2_NAME")["SALARY"]
    .count()
    .reindex(balanced_salary_summary.index))

# Pastel purple for bars
pastel_purple = "rgba(199, 168, 255, 0.8)"
fig_balanced = go.Figure()
fig_balanced.add_trace(
    go.Bar(
        x=balanced_salary_summary.index,
        y=balanced_salary_summary.values,
        text=[f"${v:,.0f}"
            for v in balanced_salary_summary.values
            ],
        textposition="outside",
        marker_color=pastel_purple,
        name="Median Salary"
    )
)

# Add trend line for Mixed industry job counts
fig_balanced.add_trace(
    go.Scatter(
        x=balanced_salary_summary.index,
        y=balanced_job_counts.values,
        mode="lines+markers",
```

```
        name="Job Count",
        yaxis="y2"
    )
)

fig_balanced.update_layout(
    title="Median Salary and Job Count Across Mixed Industries",
    xaxis_title="Industry",
    yaxis_title="Median Salary",
    yaxis2=dict(
        title="Job Count",
        overlaying="y",
        side="right",
    ),
    xaxis_tickangle=40,
    # width=1100,
    height=800,
    margin=dict(l=50, r=70, t=80, b=150),
    legend=dict(
        orientation="h",
        yanchor="bottom",
        y=1.02,
        xanchor="right",
        x=1)
)
fig_balanced.show()
```

```
Unable to display output for mime type(s): text/html
```

Mixed industries show a narrower salary range - between $80k and $120k. Retail Trade offers the highest median pay ($120,000) despite moderate job availability (755), while Administrative and Support Services has the largest job count (3584) but lower median wages ($95,800). Real Estate, Public Administration and Arts/Entertainment maintain moderate salaries with smaller job counts, reflecting niche but steady career paths. Overall, these industries appear more stable across pay and job availability, without the sharp disparities seen in male-dominated sectors.

Conclusion

The analysis shows that male-dominated industries have the highest number of job postings and median salaries, while female-dominated industries generally offer fewer positions and lower median pay, with some exceptions in high-paying sectors like Finance and Accommodation. Mixed industries display moderate job counts and salaries, suggesting more balanced opportunities. Overall, gender dominance in an industry is strongly associated with both job availability and compensation patterns, highlighting structural differences across sectors.

---

title: “Machine Learning Models” subtitle: “Modeling the Impact of Industry Gender Composition on Salaries” bibliography: references.bib csl: csl/econometrica.csl format: html: code-fold: true # code-tools: true

---

Overview

This analysis examines how industry-level gender dominance relates to advertised salaries across job postings. To incorporate this factor into our models, we assigned each NAICS 2-digit industry code to one of three gender representation categories based on U.S. labor statistics:

- **Male-dominated**
- **Mixed**
- **Female-dominated**

These categories reflect broad workforce participation trends across major sectors. Male-dominated sectors typically include labor-intensive industries such as construction and transportation, while female-dominated sectors include health care, education, and administrative or professional services. Mixed sectors show more balanced gender representation or significant role-based variation.

Integration into Modeling

These three groups were then encoded numerically as:

- `0` → Male-dominated
- `1` → Mixed
- `2` → Female-dominated

This encoding allows the gender composition of industries to be used directly as a structured feature inside regression and machine learning models.

By examining salary differences across these groups—while controlling for experience requirements, employment type, remote status, staffing-company involvement, and geographic factors—we evaluate whether certain industry gender compositions correspond to systematically higher or lower advertised wages within the professional and technical job postings present in our dataset.

```
import pandas as pd
import numpy as np
```

```
lightcast_jp = pd.read_csv(
    "data/lightcast_gender.csv",
    low_memory=False
)

gender_encoding = {
    "Male-dominated": 0,
    "Mixed": 1,
    "Female-dominated": 2,
}
lightcast_jp["GENDER_DOMINANCE_CODE"] = lightcast_jp[
    "GENDER_DOMINANCE"
    ].map(gender_encoding)
```

Linear Regression

The goal of this model is to answer the question:

> How does the gender dominance of an industry affect the offered salary in job postings?

We trained a linear regression model using the following features:

- Gender dominance
- Years of Experience
- Employment type
- Remote type
- Internship indicator
- Staffing company indicator
- State
- NAICS code

```
features = [
    "SALARY", "GENDER_DOMINANCE_CODE", "MIN_YEARS_EXPERIENCE",
    "EMPLOYMENT_TYPE", "REMOTE_TYPE", "IS_INTERNSHIP",
    "COMPANY_IS_STAFFING", "STATE_NAME", "NAICS_2022_2"]

df_model = lightcast_jp[features]
# df_model.isna().sum()

df_model = pd.get_dummies(
    df_model,
    columns=["IS_INTERNSHIP", "COMPANY_IS_STAFFING", "STATE_NAME"],
    drop_first=True
)
```

After preparing the dataset and encoding categorical features, we trained a linear regression model using an 80/20 train–test split.

The coefficient for GENDER_DOMINANCE represents the expected change in salary when moving from one dominance group to the next (male → mixed → female).

Model Training

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X = df_model.drop("SALARY", axis=1)
y = df_model["SALARY"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=79
)

model = LinearRegression()
model.fit(X_train, y_train)

coef = model.coef_[list(X.columns).index("GENDER_DOMINANCE_CODE")]
print(f"Salary change per category step: {coef:.3f}")
```

```
Salary change per category step: -6998.673
```

Moving from a male-dominated to a mixed industry is associated with an average salary decrease of about $7K.
Moving from mixed to female-dominated shows an additional decrease of roughly the same magnitude.

Model Performance Evaluation

To assess the performance of the linear regression model, we compute the R^2 score using the test dataset. Our model explains approximately 24.3% of the variation in advertised salaries. Meaning that the model captures some underlying structure—especially differences driven by experience, industry, and location—but much of the salary variation remains unexplained under a linear framework.

```
from sklearn.metrics import r2_score, mean_squared_error

y_pred_lr = model.predict(X_test)
lr_r2 = r2_score(y_test, y_pred_lr)
lr_rmse = mean_squared_error(y_test, y_pred_lr) ** 0.5
print(f"R²: {lr_r2:.4f}")
print(f"RMSE: {lr_rmse:.4f}")
```

```
R²: 0.2432
RMSE: 39206.1659
```

Predicted Salary by Dominance Group

Using the average job posting profile in our dataset, we generated predicted salaries for each gender dominance category. The results show a **downward trend in salary as industries become more female-dominated**, even after controlling for experience, employment type, remote status, staffing firm involvement, and state-level differences. In other words, job postings in male-dominated industries are associated with higher advertised salaries, while female-dominated sectors tend to offer lower salaries for otherwise comparable postings in our dataset.

```
base = X_train.mean().copy()

pred = {}
for group_code in [0, 1, 2]:
    base["GENDER_DOMINANCE_CODE"] = group_code
    pred[group_code] = model.predict(base.to_frame().T)[0]

merged = pd.DataFrame({
    "GENDER_DOMINANCE_CODE": list(pred.keys()),
    "PREDICTED_SALARY": list(pred.values())
})
merged["GENDER_DOMINANCE"] = merged["GENDER_DOMINANCE_CODE"].map(
    {v: k for k, v in gender_encoding.items()}
    )
merged = merged[["GENDER_DOMINANCE", "PREDICTED_SALARY"]]
print(merged)
```

```
   GENDER_DOMINANCE  PREDICTED_SALARY
0    Male-dominated     123564.708984
1             Mixed     116566.035994
2  Female-dominated     109567.363003
```

Random Forest

The Random Forest model is used to capture non-linear relationships between job posting characteristics and advertised salary. Unlike linear regression, which assumes a straight-line effect for each feature, Random Forests build many decision trees and combine their predictions. This allows the model to detect more complex interactions across features such as industry, remote work status, experience requirements, and gender dominance.

Model Training

We specify the number of trees, allow trees to grow to full depth, and set a random seed to ensure reproducible results. Using multiple decision trees makes the model more robust and reduces overfitting.

During training, the forest collectively learns how salary varies with experience level, industry code, employment type, remote type, and other factors.

```
from sklearn.ensemble import RandomForestRegressor
```

```
rf_model = RandomForestRegressor(
    n_estimators=300,
    max_depth=None,
    random_state=79,
    n_jobs=-1
)

rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
```

Model Performance Evaluation

We evaluate the model using two metrics:

- **R^2:** Measures how much variance in salary the model explains.
- **RMSE:** Measures average prediction error in dollar terms.

The model achieves an R^2 of approximately 0.44, indicating moderate predictive power given the complexity and noise typical of job posting salary data.

```
rf_r2 = r2_score(y_test, y_pred_rf)
rf_rmse = mean_squared_error(y_test, y_pred_rf) ** 0.5

print(f"R²: {rf_r2:.4f}")
print(f"RMSE: {rf_rmse:,.4f}")
```

```
R²: 0.4381
RMSE: 33,780.1637
```

Feature Importance Analysis

Random Forest provides an estimate of each feature's importance based on how much it reduces prediction error across the forest of trees. In our results, experience and industry code are the strongest predictors, while gender dominance plays a smaller but measurable role.

```
rf_importances = (
pd.Series(rf_model.feature_importances_, index=X.columns)
.sort_values(ascending=False)
)
print(rf_importances.head(10))
```

```
MIN_YEARS_EXPERIENCE        0.375126
NAICS_2022_2                0.157024
REMOTE_TYPE                 0.068725
GENDER_DOMINANCE_CODE       0.056740
EMPLOYMENT_TYPE             0.040388
IS_INTERNSHIP_True          0.025948
STATE_NAME_California       0.025122
COMPANY_IS_STAFFING_True    0.023374
```

```
STATE_NAME_Texas            0.014256
STATE_NAME_New York         0.013872
dtype: float64
```

Predicted Salary by Dominance Group

The Random Forest model shows substantial salary differences across gender-dominance categories. These predictions suggest that male-dominated industries carry a significant salary premium, with advertised salaries nearly $56,000 higher than those in mixed or female-dominated industries. In contrast, the salaries for mixed and female-dominated sectors are almost identical.

```
base_rf = X_train.mean().copy()

pred_rf = {}
for group_code in [0, 1, 2]:
    base_rf["GENDER_DOMINANCE_CODE"] = group_code
    pred_rf[group_code] = rf_model.predict(base_rf.to_frame().T)[0]

rf_merged = pd.DataFrame({
"GENDER_DOMINANCE_CODE": list(pred_rf.keys()),
"PREDICTED_SALARY_RF": list(pred_rf.values())
})
rf_merged["GENDER_DOMINANCE"] = rf_merged["GENDER_DOMINANCE_CODE"].map(
{v: k for k, v in gender_encoding.items()}
)
rf_merged = rf_merged[["GENDER_DOMINANCE", "PREDICTED_SALARY_RF"]]
print(rf_merged)
```

```
   GENDER_DOMINANCE  PREDICTED_SALARY_RF
0    Male-dominated        128001.509402
1             Mixed         71684.453611
2  Female-dominated         72021.826390
```

---

title: “NLP Methods” subtitle: “Text Mining and Skill Extraction from Job Descriptions” bibliography: references.bib csl: csl/econometrica.csl format: html: code-fold: true # code-tools: true

---

Overview

Natural Language Processing (NLP) techniques allow us to extract patterns, themes, and linguistic signals embedded in job descriptions.
While structured data captures industry, salary, skills, and location, unstructured job text reveals employers’ expectations, behavioral traits, and role-specific competencies that are not always encoded in Lightcast’s structured fields.

In this section, we:

- clean and normalize job description text
- tokenize and filter low-value words
- compute word frequencies and visualize them
- apply TF–IDF to identify high-information terms
- quantify the presence of technical keywords in descriptions

These insights complement earlier EDA and Skill Gap Analysis results by highlighting **how employers talk** about data and analytics roles in practice.

Load the Dataset

```
import pandas as pd
import numpy as np
import re
from collections import Counter
import plotly.express as px
from wordcloud import WordCloud
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer

df = pd.read_csv("data/lightcast_cleaned.csv", low_memory=False)

# Keep only the columns we need for NLP
df_text = df[["TITLE_NAME", "BODY", "NAICS_2022_2_NAME",
"NAICS_2022_2"]].copy()
# df_text.head()
```

Text Cleaning Pipeline

We apply a lightweight but robust text cleaning pipeline: 1. Convert all text to lowercase 2. Remove punctuation, digits, and symbols 3. Collapse repeated whitespace 4. Tokenize into individual words 5. Remove very common “filler” words using a custom stopword list 6. Keep only alphabetic tokens with length ≥ 4

This produces a semantically meaningful set of tokens for each job description while avoiding heavy external dependencies.

```
# Simple tokenizer using regex + Python only (no NLTK)
def simple_tokenize(text: str):
    text = str(text).lower()
```

```
    text = re.sub(r"[^a-z\s]", " ", text)   # keep only letters and spaces
    text = re.sub(r"\s+", " ", text).strip()
    tokens = text.split()
    return [w for w in tokens if len(w) > 3]

# Custom stopword list (focused on generic English terms)
custom_stopwords = {

"this","that","with","from","about","there","their","which","have","were",
    "been","also","into","such","they","them","your","will","would","could",

"should","other","than","some","more","when","what","where","these","those",
    "just","here","very","much","many","most","over","under","while","after",
    "before","still","next","only","each","every","then","because","within",
    "including","using","across","through"
}

def clean_text(t: str) -> str:
    t = str(t).lower()
    t = re.sub(r"[^a-z\s]", " ", t)
    t = re.sub(r"\s+", " ", t).strip()
    return t

# Clean + tokenize BODY text
df_text["clean_body"] =
df_text["BODY"].fillna("").astype(str).apply(clean_text)

df_text["tokens"] = df_text["clean_body"].apply(
    lambda text: [w for w in simple_tokenize(text) if w not in
custom_stopwords]
)

df_text[["TITLE_NAME", "tokens"]].head()
```

	TITLE_NAME	tokens
0	Enterprise Analysts	[enterprise, analyst, merchandising, dorado, a...
1	Oracle Consultants	[oracle, consultant, reports, augusta, maine, ...
2	Data Analysts	[taking, care, people, heart, everything, star...
3	Management Analysts	[role, wells, fargo, looking, platform, tools,...
4	Unclassified	[comisiones, semana, comiensa, rapido, modesto...

Word Frequency Extraction

We aggregate a global vocabulary across all postings and compute the most frequently occurring terms.

```
all_words = []
df_text["tokens"].apply(all_words.extend)

word_freq = Counter(all_words).most_common(20)
freq_df = pd.DataFrame(word_freq, columns=["word", "count"])
freq_df.head()
```

	word	count
0	data	512950
1	experience	361495
2	business	301725
3	work	234020
4	skills	181358

The bar chart below highlights the 20 most common lexical terms appearing across job descriptions.

```
fig_freq = px.bar(
    freq_df.sort_values("count", ascending=True),
    x="count",
    y="word",
    orientation="h",
    title="Most Common Terms in Job Descriptions",
    labels={"count": "Frequency", "word": "Term"}
)
fig_freq.update_layout(
    xaxis=dict(
        showgrid=True,
        gridcolor="#eaeaea",
    ),
    yaxis=dict(
        categoryorder='total ascending',
    ),
    plot_bgcolor="white",
    paper_bgcolor="white",
    showlegend=False
)
fig_freq.show()
```

```
Unable to display output for mime type(s): text/html
```

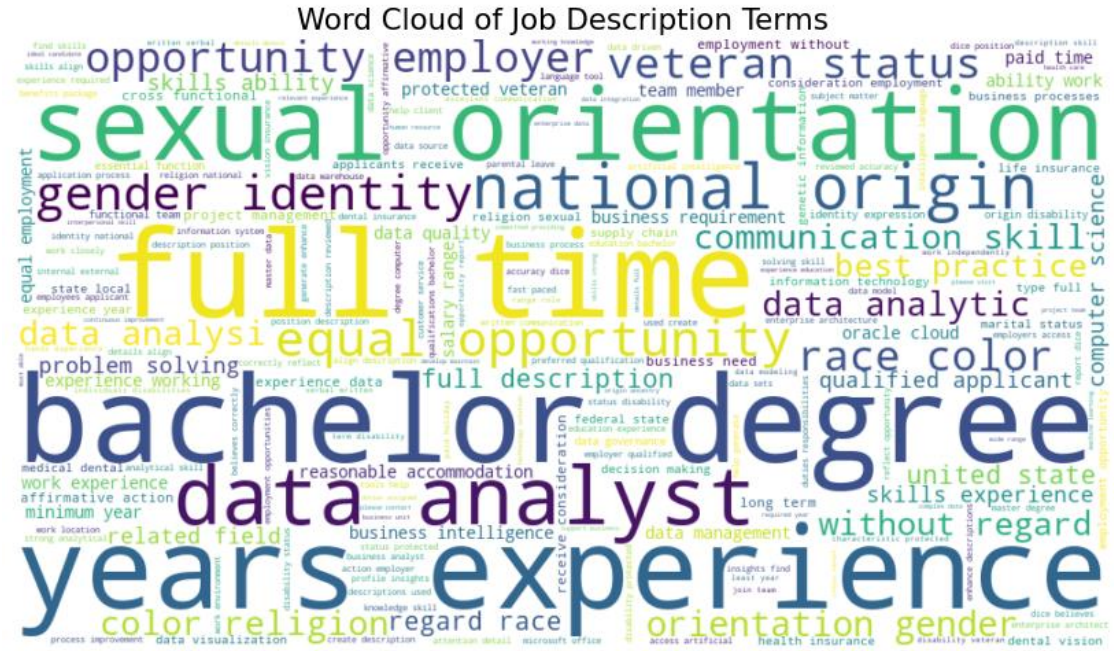
Word Cloud

A word cloud provides a quick, intuitive view of recurring terms. Large words appear more frequently in text, giving a qualitative impression of employer emphasis.

```
wc = WordCloud(
    width=900,
```

```
    height=500,
    background_color="white",
    colormap="viridis"
).generate(" ".join(all_words))

plt.figure(figsize=(12, 6))
plt.imshow(wc, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud of Job Description Terms", fontsize=16)
plt.show()
```



TF–IDF: High-Information Terms Across Job Descriptions

Raw word counts can be dominated by generic terms (for example, “team,” “support,” “experience”). To focus on high-information terms that distinguish postings, we apply TF–IDF (Term Frequency–Inverse Document Frequency).

This gives higher scores to words that

- appear frequently within a posting, but
- do not appear in every other posting.

```
# For performance, optionally sample a subset if the dataset is huge
MAX_DOCS = 15000
if len(df_text) > MAX_DOCS:
    df_tfidf = df_text.sample(n=MAX_DOCS, random_state=42).copy()
else:
    df_tfidf = df_text.copy()

tfidf_vectorizer = TfidfVectorizer(
```

```
    max_features=3000,
    min_df=5,
    max_df=0.6,
    stop_words=None  # custom stopwords already applied at the text level
)

tfidf_matrix = tfidf_vectorizer.fit_transform(df_tfidf["clean_body"])
feature_names = np.array(tfidf_vectorizer.get_feature_names_out())
# tfidf_matrix.shape

# Compute average TF-IDF score per term across all sampled documents
avg_tfidf = tfidf_matrix.mean(axis=0).A1

tfidf_df = (
    pd.DataFrame({"term": feature_names, "score": avg_tfidf})
    .sort_values("score", ascending=False)
)
# tfidf_top = tfidf_df.head(25)
tfidf_df.head(3)
```

	term	score
2394	sap	0.053556
1899	oracle	0.035554
2995	your	0.031992

```
fig_tfidf = px.bar(
    tfidf_df.head(25).sort_values("score", ascending=True),
    x="score",
    y="term",
    orientation="h",
    title="Top 25 High-Information Terms (TF-IDF)",
    labels={"score": "Average TF-IDF Score", "term": "Term"},
    # color="term",  # categorical coloring
    # color_discrete_sequence=px.colors.qualitative.Set3,  # vibrant, premium
palette
    # height=700
)
fig_tfidf.update_layout(
    xaxis=dict(
        showgrid=True,
        gridcolor="#eaeaea",
    ),
    yaxis=dict(
        categoryorder='total ascending',
    ),
    plot_bgcolor="white",
    paper_bgcolor="white",
    showlegend=False
```

```
)
fig_tfidf.show()
```

```
Unable to display output for mime type(s): text/html
```

TF–IDF by Industry (Macro Lens)

To connect language patterns with industry context, we compare TF–IDF profiles across a few major NAICS sectors:

- Information (51)
- Finance and Insurance (52)
- Professional, Scientific, and Technical Services (54)

```
# Map NAICS_2022_2 to readable sector labels
sector_map = {
    51.0: "Information",
    52.0: "Finance & Insurance",
    54.0: "Professional, Scientific & Technical"
}

df_text["NAICS_2022_2"] = df["NAICS_2022_2"].astype(float)
df_text["sector"] = df_text["NAICS_2022_2"].map(sector_map)

df_sector = df_text.dropna(subset=["sector", "clean_body"]).copy()
df_sector["sector"].value_counts()
```

```
sector
Professional, Scientific & Technical    22339
Finance & Insurance                      6990
Information                              3771
Name: count, dtype: int64
```

```
# Build separate corpora per sector
sector_texts = (
    df_sector
    .groupby("sector")["clean_body"]
    .apply(lambda s: " ".join(s.tolist()))
)

sector_texts
```

```
sector
Finance & Insurance                     taking care of people is at the heart
of every...
Information                             about lumen lumen connects the world
we are ig...
Professional, Scientific & Technical    sr marketing analyst united states ny
new york...
Name: clean_body, dtype: object
```

```
# Separate vectorizer for sector-level TF-IDF
sector_vectorizer = TfidfVectorizer(
    max_features=2000,
    min_df=1,    # allow terms that appear in at least one sector corpus
    max_df=1.0,  # do not drop terms that appear across multiple sectors
    stop_words=None
)

sector_tfidf = sector_vectorizer.fit_transform(sector_texts.values)
sector_terms = np.array(sector_vectorizer.get_feature_names_out())

sector_tfidf.shape
```

```
(3, 2000)
```

```
# For each sector, extract its top 15 TF-IDF terms
rows = []
for i, sector_name in enumerate(sector_texts.index):
    row_vec = sector_tfidf[i].toarray().ravel()
    top_idx = row_vec.argsort()[-15:][::-1]
    for idx in top_idx:
        rows.append({
            "sector": sector_name,
            "term": sector_terms[idx],
            "score": row_vec[idx]
        })

sector_tfidf_df = pd.DataFrame(rows)
sector_tfidf_df.head()
```

	sector	term	score
0	Finance & Insurance	and	0.671229
1	Finance & Insurance	to	0.368418
2	Finance & Insurance	the	0.301178
3	Finance & Insurance	of	0.272638
4	Finance & Insurance	in	0.171988

```
fig_sect = px.bar(
    sector_tfidf_df,
    x="score",
    y="term",
    color="sector",
    barmode="group",
    orientation="h",
    title="High-Information Terms by Sector (TF-IDF – Top 15 per Sector)",
    labels={"score": "TF-IDF Score", "term": "Term", "sector": "Sector"}
    # height=800
)
fig_sect.update_layout(
```

```
    xaxis=dict(
        showgrid=True,
        gridcolor="#eaeaea",
    ),
    yaxis=dict(
        categoryorder='total ascending',
    ),
    legend=dict(orientation="v", yanchor="top", y=0.25, xanchor="left",
x=0.75)
    )
fig_sect.show()
```

```
Unable to display output for mime type(s): text/html
```

Technical Terms Directly from Descriptions

Although Lightcast provides structured software skill fields, job descriptions often repeat these terms inside free text. To quantify this, we scan cleaned descriptions for common technical keywords.

```
technical_terms = [
    "python", "sql", "excel", "tableau", "powerbi", "power bi",
    "aws", "azure", "cloud", "machine learning",
    "analytics", "analysis", "business intelligence"
]

tech_counts = {}
for term in technical_terms:
    pattern = term.replace(" ", "")  # basic normalization for "power bi" ->
"powerbi"
    tech_counts[term] = df_text["clean_body"].str.contains(pattern,
regex=False).sum()

tech_df = (
    pd.DataFrame.from_dict(tech_counts, orient="index", columns=["count"])
    .sort_values("count", ascending=True)
    .reset_index()
    .rename(columns={"index": "term"})
)
tech_df.head()
```

	term	count
0	machine learning	22
1	business intelligence	56
2	powerbi	3115
3	power bi	3115
4	azure	5580

```
fig_tech = px.bar(
    tech_df,
    x="count",
    y="term",
    orientation="h",
    title="Technical Terms Found in Job Descriptions",
    labels={"count": "Mentions", "term": "Term"},
    # color="count",
    # color_continuous_scale=px.colors.sequential.GnBu,
    # height=600
)
fig_tech.update_layout(
    xaxis=dict(
        showgrid=True,
        gridcolor="#eaeaea",
    ),
    yaxis=dict(
        categoryorder='total ascending',
    ))
fig_tech.show()
```

```
Unable to display output for mime type(s): text/html
```

Interpretation & Insights

- Dominant Themes in Employer Language
    - Global frequency and word clouds show recurring emphasis on experience, support, team, management, and responsibilities.
    - This indicates that employers value not only technical competence but also the ability to operate in collaborative, process-oriented environments.
- High-Information Terms (TF–IDF)
    - TF–IDF surfaces more specialized vocabulary (e.g., “pipeline,” “analytics,” “visualization,” “governance”) that differentiates advanced analytics roles from generic postings.
    - These terms often correspond to specific project responsibilities or technical domains within data teams.
- Sector-Specific Vocabulary
    - In Finance & Insurance, TF–IDF highlights concepts related to risk, portfolios, credit, and regulatory reporting.
    - In Professional, Scientific & Technical Services, terms linked to modeling, experimentation, and client delivery become more prominent.

    - In Information, we see emphasis on platforms, content, and digital products.

    - This demonstrates how sector context shapes the language of “data work.”
- Technical Signal in Text
    - The repeated presence of terms such as SQL, Python, Excel, Tableau, Power BI, AWS, and Azure inside descriptions reinforces the structured skill patterns observed in the EDA section.

    - These technologies function as “linguistic anchors” that clearly distinguish analytics roles from general business positions.
- Implications for Job Seekers
    - Candidates aligning with these text-level signals (SQL + Python + BI + cloud) are better positioned for data-intensive roles.

    - Beyond tools, employers repeatedly emphasize concepts related to analysis, insights, decision-making, and stakeholder communication — confirming that soft skills and interpretation capabilities matter as much as raw coding ability.

    - Understanding how employers write about roles helps job seekers tailor resumes, cover letters, and LinkedIn profiles using the same vocabulary that appears in successful job descriptions.

Taken together, the NLP results provide a qualitative, language-based complement to our structured EDA and Skill Gap Analysis, strengthening the case for a hybrid skill profile: technical depth, sector awareness, and communication skills that translate data into decisions.

Objective

This section evaluates how our team’s current technical capabilities compare to the skills most frequently demanded in the job market for Data & Analytics roles.

Using Lightcast job postings, we extracted software and technical skills commonly requested by employers. We then compared these industry requirements to the team’s self-assessed proficiency levels across key tools such as SQL, Python, Tableau, Power BI, R, and AWS.

The goal is to identify strengths, highlight capability gaps, and provide insights that can guide learning plans and role alignment within the team.

Team Skill Matrix

The heatmap below shows each team member's self-assessed proficiency (1–5) across selected analytics and data engineering skills.

Darker shades indicate higher proficiency.

```
team_members = ["Dinara", "Nhat", "Leo"]
skills = [
    "SQL (Programming Language)",
    "Python (Programming Language)",
    "Tableau (Business Intelligence Software)",
    "Power BI",
    "Microsoft Excel",
    "R (Programming Language)",
    "Amazon Web Services",
]

df_skills = pd.DataFrame(
    {
        "Name": team_members,
        "SQL (Programming Language)": [5, 3, 3],
        "Python (Programming Language)": [4, 3, 4],
        "Tableau (Business Intelligence Software)": [3, 4, 2],
        "Power BI": [3, 5, 2],
        "Microsoft Excel": [5, 4, 4],
        "R (Programming Language)": [4, 3, 3],
        "Amazon Web Services": [4, 3, 3],
    }
).set_index("Name")

fig_heat = px.imshow(
    df_skills,
    text_auto=True,
    aspect="auto",
    color_continuous_scale="Blues",
    labels=dict(color="Skill Level"),
    title="Team Skill Matrix"
)
fig_heat.update_yaxes(title="Team Member")
fig_heat
```

```
Unable to display output for mime type(s): text/html
```

Market Alignment Analysis

Extracting Industry Demand

We filter job postings to focus exclusively on data-centric roles within data-heavy industries (NAICS sectors: Information, Finance & Insurance, and Professional Services).

These sectors employ a large proportion of data & analytics professionals, making them strong benchmarks for industry expectations.

We then identify postings related to analytics roles using keyword matching on job titles.

Finally, we extract software skills listed in job descriptions and compute how frequently each skill appears across the filtered postings.

Higher skill frequency = greater demand = higher importance for employability.

```
naics_data_industries = [51, 52, 54]
df_data_industry = lightcast_jp[
    lightcast_jp["NAICS_2022_2"].astype(float).isin(naics_data_industries)
]

data_keywords = [
    "data", "analytics", "analysis", "analyst",
    "machine learning", "ml", "ai",
    "business intelligence", "cloud",
    "sql", "python"
]

df_data_roles = lightcast_jp[

lightcast_jp["TITLE_NAME"].str.lower().str.contains('|'.join(data_keywords),
na=False)
]

df_filtered = df_data_industry[

df_data_industry["TITLE_NAME"].str.lower().str.contains('|'.join(data_keyword
s))
]
```

Software Skills Extraction

Lightcast provides skill data as comma-separated text fields.

We parse and standardize these strings, explode them into individual skills, and aggregate counts across all postings. This produces a clean, frequency-based ranking of the most in-demand technologies in data-oriented careers.

```
df_filtered = df_filtered.copy()
df_filtered["software_skill"] = (
    df_filtered["SOFTWARE_SKILLS_NAME"]
        .fillna("")
        .str.split(",")
```

```
        .apply(lambda lst: [s.strip() for s in lst if s.strip() != ""])
)

df_sw_exploded = (
    df_filtered
        .explode("software_skill")
        .dropna(subset=["software_skill"])
)

industry_skill_counts = (
    df_sw_exploded["software_skill"]
        .value_counts()
        .to_frame(name="count")
)

industry_skill_counts.head()
```

	count
software_skill	
SQL (Programming Language)	7684
Python (Programming Language)	4620
Dashboard	4202
Tableau (Business Intelligence Software)	4085
Power BI	3695

Gap Calculation

To quantify alignment between our team and the market, we compare:

- Team Score - average proficiency ratings (1–5 scale) provided internally
- Industry Score - demand-derived skill importance (normalized from posting frequency)

The Gap metric is defined as: $Gap = IndustryScore - TeamScore$

- A positive gap means the team exceeds market expectations.
- A negative gap highlights opportunities for additional training or hiring.

```
team_avg = df_skills.mean().to_frame(name="team_score")

ind_skills = industry_skill_counts.rename_axis("Skill")
ind_skills = ind_skills.loc[skills]

gap_df = team_avg.join(ind_skills, how="left")

max_demand = gap_df["count"].max()
gap_df["industry_score"] = (gap_df["count"] / max_demand) * 5
```

```
gap_df["gap"] = gap_df["industry_score"] - gap_df["team_score"]
gap_df
```

	team_score	count	industry_score	gap
SQL (Programming Language)	3.666667	7684	5.000000	1.333333
Python (Programming Language)	3.666667	4620	3.006247	-0.660420
Tableau (Business Intelligence Software)	3.000000	4085	2.658121	-0.341879
Power BI	3.333333	3695	2.404347	-0.928987
Microsoft Excel	4.333333	3642	2.369859	-1.963474
R (Programming Language)	3.333333	2312	1.504425	-1.828909
Amazon Web Services	3.333333	1126	0.732691	-2.600642

Skill Gap Visualizations

Bar Chart Comparison

```
gap_long = (
    gap_df[["team_score", "industry_score"]]
    .reset_index(names="Skill")
    .melt(id_vars="Skill", var_name="Source", value_name="Score")
)
skill_order =
gap_df["industry_score"].sort_values(ascending=False).index.to_list()
fig_gap = px.bar(
    gap_long.sort_values("Score", ascending=False),
    x="Score",
    y="Skill",
    color="Source",
    barmode="group",
    orientation="h",
    title="Skill Gap Analysis",
    labels={"Score": "Score (1-5)", "Skill": "", "Source": ""},
    category_orders={"Skill": skill_order}
)
fig_gap.update_layout(
    legend_title_text="",
    xaxis=dict(range=[0, 5]),
    legend=dict(orientation="v", yanchor="top", y=0.15, xanchor="left",
```

```
x=0.75)
)
fig_gap
```

```
Unable to display output for mime type(s): text/html
```

Radar Chart Comparison

```
skills_list = gap_df.index.tolist()
industry_values = gap_df["industry_score"].tolist()

skills_list += [skills_list[0]]
industry_values += [industry_values[0]]

fig = go.Figure()
for member in df_skills.index:
    values = df_skills.loc[member].tolist()
    values_loop = values + [values[0]]

    fig.add_trace(go.Scatterpolar(
        r=values_loop,
        theta=skills_list,
        fill='toself',
        name=member,
        opacity=0.3,
        line=dict(width=1)
    ))

fig.add_trace(go.Scatterpolar(
    r=industry_values,
    theta=skills_list,
    fill='toself',
    name='Industry Demand',
    line=dict(color='rgba(255, 127, 14, 0.9)', width=3),
    fillcolor='rgba(255, 127, 14, 0.5)'
))
fig.update_layout(
    title="Skill Gap Analysis",
    polar=dict(radialaxis=dict(visible=True, range=[0, 5])),
    showlegend=True,
    legend=dict(orientation="h", y=-0.2)
)
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

Improvement Plan

How can the team collaborate to bridge skill gaps?

While our team demonstrates strong proficiency across most software skills, we still need more consistency to meet industry expectations. To address this, we developed a collaborative improvement plan where members with higher expertise will guide others through structured peer learning. The plan focuses on small but consistent practice activities, especially in the skills with the highest industry demand: Python, SQL, Tableau/Power BI, and Microsoft Excel. For example, we will practice SQL by solving LeetCode problems and joining SQL contests, and strengthen Power BI and Tableau skills by creating hands-on visualizations. We will also remain active on GitHub to document progress and maintain a strong professional portfolio.

---

title: “Career Strategy Plan”

Practical Applications

To convert our job market analytics into actionable career recommendations, we followed a structured three-step framework:

1. Market Demand Extraction
We analyzed Lightcast job postings for Data & Analytics roles across high-demand sectors (Information, Finance & Insurance, and Professional Services).

2. Team Capability Assessment
Each member evaluated their skill levels (1–5) across seven core analytics competencies.

3. Gap & Alignment Strategy
By comparing industry demand vs. team proficiency, we identified skill strengths, shortages, and role-alignment recommendations.

Individual Strategy Profiles

Dinara

Skill Profile (Interactive Radar Chart)

Dinara’s Improvement Plan

AWS Skills

Focus on expanding cloud fundamentals beyond EC2, next steps include S3, IAM, Lambda, and RDS. The goal is to build a small end-to-end pipeline on AWS and publish it with diagrams and documentation.

Visualization Tools (Tableau & Power BI)

Increase hands-on experience with dashboard tools by recreating datasets into visual stories. Dashboards will be shared on Tableau Public and Power BI, with short summaries of insights and methodology.

Portfolio Development

Strengthen the portfolio through project-based learning: data pipelines, SQL analyses, interactive dashboards, and business analytics mini-projects. Each project will be fully documented on GitHub and added to the website.

Python & SQL Consistency

Maintain steady practice in core programming skills through SQL challenges, exploratory notebooks, and small automation scripts. Emphasis on clean code, reproducibility, and clarity.

Professional Branding

Enhance visibility through consistent GitHub activity, project write-ups, and LinkedIn updates. Integrate learning progress and completed dashboards into the website to show growth over time.

Nhat

Skill Profile (Interactive Radar Chart)

Interpretation & Career Strategy

Cloud & Data Engineering Readiness

I plan to extend my cloud skills by moving into services that support analytics workflows, including S3 for storage, IAM for access management, and Lambda for automation. Applying these tools to a small pipeline inspired by the data work I've done before will help solidify the concepts. Clear diagrams and documentation will reinforce the design.

Visualization & Analytical Storytelling

To strengthen how I communicate insights, I'll continue developing dashboards that turn complex datasets into clear visual narratives. Rebuilding analyses in Tableau or Power BI and adding concise explanations of the methods and findings will help sharpen both structure and storytelling.

Business Analytics Portfolio Growth

I plan to expand my portfolio through projects that reflect my background in macro analysis and data-driven interpretation. Creating well-documented mini-studies on topics like inflation signals, labor patterns, or pricing behavior will highlight both technical ability and business reasoning. Each will include reproducible code and clear takeaways.

Python & SQL Development

I want to keep improving my coding habits through regular practice in both Python and SQL. Writing more organized scripts, refining queries, and automating small tasks related to my coursework or past analytics work will strengthen consistency and technical depth.

Professional Presence & Branding

I'll enhance my professional presence by sharing dashboards, code, and summaries on GitHub, LinkedIn, and my personal website. Bringing together my coursework, macro insights, and analytics projects in one place will help present a cohesive view of my development.

Hung (Leo)

Skill Profile (Interactive Radar Chart)

Interpretation & Recommendations

As I think about my current skill profile, I'm beginning to understand more clearly how my strengths connect with the kind of work I want to pursue in data and analytics. Skills like SQL, Python, and Excel have become the backbone of the way I approach problems, and they give me the confidence to work through analytical workflows as well as explore the early stages of data engineering. I'm also becoming more comfortable with AWS, and as I continue learning it, I can see how valuable this will be as cloud technologies become more standard across the industry.

At the same time, this project made it very clear where I need to grow next. Visualization tools such as Tableau and Power BI show up in almost every job posting I look at, and developing stronger skills in these areas would help me tell better stories with data and communicate my ideas more effectively. I also want to deepen my knowledge of AWS, especially services like S3, Lambda, Glue, and Redshift, because understanding how these pieces fit together will allow me to build more complete, automated, and scalable analytics pipelines.

Looking ahead, I can see myself moving toward a more technical analytics path, whether that means becoming a Technical Data Analyst, an Analytics Engineer, or even a Junior Data Engineer. To prepare for that future, I want to strengthen my abilities in Tableau and Power BI so I can improve the way I communicate insights, and I also want to advance my AWS proficiency by eventually earning the relevant certifications. I plan to start building more portfolio projects that combine SQL, Python, cloud services, and automation, and I also want to keep improving the way I translate technical findings into meaningful insights that people can act on.

Overall, this project helped me recognize not only the skills I already have, but also the areas where I want to push myself so I can grow into the professional I hope to become. By

focusing on visualization and deepening my cloud engineering experience, I feel more confident that I can continue developing into a strong, well-rounded analytics professional who can work across both technical and business environments.

Team Findings

Shared Strengths
- Strong analytical and SQL foundations
- Balanced Python and Excel experience
- Diverse BI + cloud skills across members

Shared Growth Opportunities
- AWS for all members
- Advanced SQL
- Dashboard portfolio development

Team 5

Graduate students in the Master of Applied Business Analytics program at Boston University,
completing this project for **AD 688 – Web Analytics**.

Team Members

Hung Nguyen Data wrangling, Quarto architecture, integration of macro/micro insights.

Nhat Tran FRED API integration, time-series visualization, macro interpretation.

Dinara Zhorabek Research framing, literature review, policy/gender analysis.

Tools & Technology Stack

- Python & PySpark
- Pandas, NumPy, Seaborn, Plotly
- FRED API and Lightcast Job Postings
- AWS (EC2)
- Quarto & GitHub Pages
- SQL and data preprocessing pipelines

Danielle J. Galvin ,Susan C. Anderson ,Chelsi J. Marolf ,Nikole G. Schneider ,Andrea L. Liebl. (2024): "Comparative analysis of gender disparity in academic positions based on u.s. Region and STEM discipline," *PLOSone*,.

Eagly, A. H., and S. J. Karau. (2002): "Role congruity theory of prejudice toward female leaders," *Psychological Review*, 109, 573–98.

Lawson, M. A., A. E. Martin, I. Huda, and S. C. Matz. (2022): "Hiring women into senior leadership positions is associated with a reduction in gender stereotypes in organizational language," *Proceedings of the National Academy of Sciences*, 119, e2026443119.

Schneider, M. B., T. Altmann, and A. I. Langen. (2023): "Gender differences in perceived legitimacy and status perception in a leadership role," *Frontiers in Psychology*, 14, 1088190.

TIME Magazine. (2025): "What japan's first female prime minister could mean for the country's gender politics," *TIME*,.