Loading Libraries and Data

Final Report

Kelly, Sabrina, Makenzie

October 15, 2025

from pyspark.sql import SparkSession  
import pandas as pd  
import matplotlib.pyplot as plt  
  
raw\_df = pd.read\_csv("data/lightcast\_job\_postings.csv")  
#raw\_df.columns.tolist()

/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_71870/1289692877.py:5: DtypeWarning:  
  
Columns (19,30) have mixed types. Specify dtype option on import or set low\_memory=False.

# Cleaning Data

columns\_to\_drop = [  
 "ID", "URL", "ACTIVE\_URLS", "DUPLICATES", "LAST\_UPDATED\_TIMESTAMP",  
 "NAICS2", "NAICS3", "NAICS4", "NAICS5", "NAICS6",  
 "SOC\_2", "SOC\_3", "SOC\_5"  
]  
raw\_df.drop(columns=columns\_to\_drop, inplace=True)  
  
# Fill missing values  
raw\_df["SALARY"].fillna(raw\_df["SALARY"].median(), inplace=True)  
raw\_df["NAICS\_2022\_6"].fillna("Unknown", inplace=True)  
  
# Drop columns with >50% missing values  
raw\_df.dropna(thresh=len(raw\_df) \* 0.5, axis=1, inplace=True)  
  
raw\_df = raw\_df.drop\_duplicates(subset=["TITLE", "COMPANY", "LOCATION", "POSTED"], keep="first")  
  
#raw\_df.head()

/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_71870/2470702404.py:9: FutureWarning:  
  
A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.  
  
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.  
  
  
  
/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_71870/2470702404.py:10: FutureWarning:  
  
A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.  
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.  
  
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.  
  
  
  
/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_71870/2470702404.py:10: FutureWarning:  
  
Setting an item of incompatible dtype is deprecated and will raise an error in a future version of pandas. Value 'Unknown' has dtype incompatible with float64, please explicitly cast to a compatible dtype first.

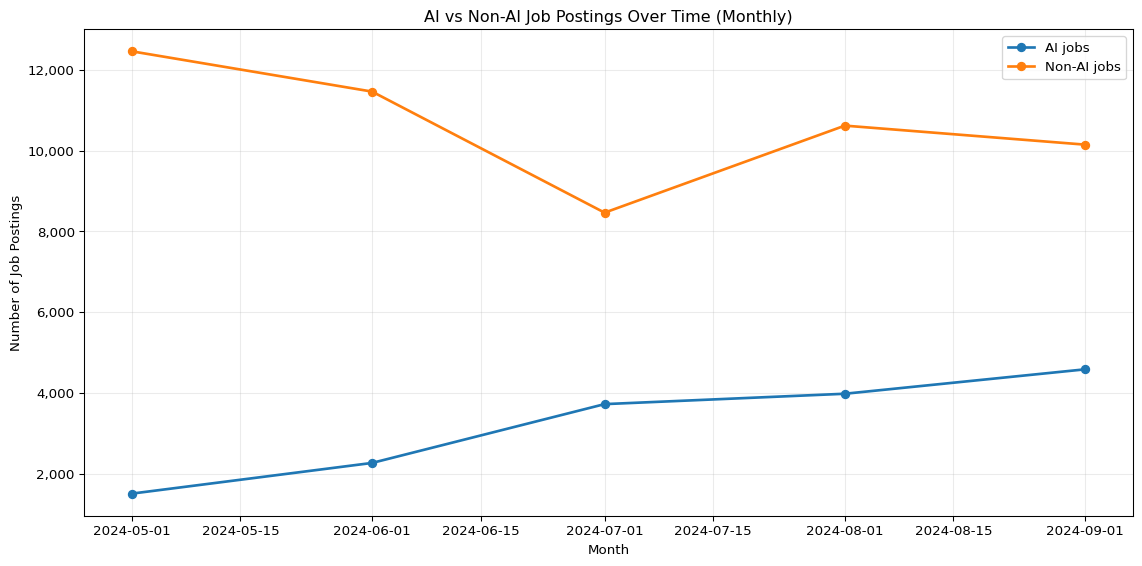
# Plot setup for AI vs Non-AI job posting count

import os, re  
import pandas as pd  
  
# Kelly’s cleaned dataframe must exist  
assert "raw\_df" in globals(), "raw\_df must exist (Kelly’s cleaned dataframe)."  
df = raw\_df.copy()  
  
# Parse date -> month  
if "POSTED" not in df.columns:  
 raise ValueError("Expected a POSTED column in raw\_df.")  
df["POSTED\_DT"] = pd.to\_datetime(df["POSTED"], errors="coerce")  
df = df.dropna(subset=["POSTED\_DT"])  
df["month"] = df["POSTED\_DT"].dt.to\_period("M").dt.to\_timestamp()  
  
# Combine likely text fields  
candidate\_text\_cols = [  
 "TITLE","TITLE\_CLEAN","TITLE\_NAME","BODY",  
 "SKILLS","SKILLS\_NAME",  
 "SPECIALIZED\_SKILLS","SPECIALIZED\_SKILLS\_NAME",  
 "SOFTWARE\_SKILLS","SOFTWARE\_SKILLS\_NAME",  
 "COMMON\_SKILLS","COMMON\_SKILLS\_NAME",  
 "CERTIFICATIONS\_NAME",  
]  
text\_cols = [c for c in candidate\_text\_cols if c in df.columns]  
if not text\_cols:  
 text\_cols = ["TITLE"] if "TITLE" in df.columns else []  
text\_series = (  
 df[text\_cols].astype(str).agg(" ".join, axis=1).str.lower()  
 if text\_cols else pd.Series([""] \* len(df), index=df.index)  
)  
  
# FIXED list (no cut-off strings)  
ai\_terms = [  
 "artificial intelligence","ai","machine learning","deep learning",  
 "neural network","nlp","natural language","computer vision",  
 "reinforcement learning","generative ai","llm","gpt","chatgpt",  
 "transformer","bert","prompt engineer","prompt engineering"  
]  
  
# Word-boundary pattern so we don't match 'retail' for 'ai'  
ai\_pattern = re.compile(  
 r"\b(?:"  
 + "|".join(re.escape(t) for t in ai\_terms)  
 + r")\b",  
 flags=re.IGNORECASE,  
)  
  
df["IS\_AI"] = text\_series.str.contains(ai\_pattern, na=False)  
  
# Monthly counts  
monthly = (  
 df.groupby(["month","IS\_AI"])  
 .size()  
 .unstack(fill\_value=0)  
 .rename(columns={True: "AI", False: "Non-AI"})  
 .sort\_index()  
)  
  
monthly\_plt = monthly.copy()  
monthly\_plt.head(12)

| IS\_AI | Non-AI | AI |
| --- | --- | --- |
| month |  |  |
| 2024-05-01 | 12460 | 1503 |
| 2024-06-01 | 11462 | 2263 |
| 2024-07-01 | 8463 | 3720 |
| 2024-08-01 | 10619 | 3977 |
| 2024-09-01 | 10148 | 4582 |

# Job Postings for AI vs Non-AI Jobs

import os  
import matplotlib.pyplot as plt  
from matplotlib.ticker import FuncFormatter  
  
assert "monthly\_plt" in globals(), "Run the setup chunk first."  
  
os.makedirs("output", exist\_ok=True)  
  
plt.figure(figsize=(12, 6))  
plt.plot(monthly\_plt.index, monthly\_plt["AI"], marker="o", linewidth=2, label="AI jobs")  
plt.plot(monthly\_plt.index, monthly\_plt["Non-AI"], marker="o", linewidth=2, label="Non-AI jobs")  
plt.title("AI vs Non-AI Job Postings Over Time (Monthly)")  
plt.xlabel("Month")  
plt.ylabel("Number of Job Postings")  
plt.gca().yaxis.set\_major\_formatter(FuncFormatter(lambda x, \_: f"{int(x):,}"))  
plt.grid(True, alpha=0.25)  
plt.legend()  
plt.tight\_layout()  
plt.savefig("output/ai\_vs\_nonai\_over\_time.png", dpi=200, bbox\_inches="tight")  
plt.show()



The AI VS Non-AI Jobs graph hows two clear trends:

* AI jobs have been increasing uninterruptedly each month from May to September 2024. This suggests growing demand for AI-related roles.
* Non-AI jobs started much higher but declined between May and July before slightly recovering in August. By September, they still remained lower than at the start.

In summary, there are still more Non-AI jobs than AI jobs in total; however, the number of available AI jobs is increasing rapidly. This transition indicates the shifting of the job market towards the AI-based roles.

# Prep for monthly AI counts

import os, re  
import numpy as np  
import pandas as pd  
  
assert "raw\_df" in globals(), "raw\_df must exist."  
  
df = raw\_df.copy()  
  
# ---- Dates -> month ----  
df["POSTED\_DT"] = pd.to\_datetime(df["POSTED"], errors="coerce")  
df = df.dropna(subset=["POSTED\_DT"])  
df["month"] = df["POSTED\_DT"].dt.to\_period("M").dt.to\_timestamp()  
  
# ---- AI detector (reuse if already present) ----  
if "IS\_AI" not in df.columns:  
 ai\_terms = [  
 "artificial intelligence","ai","machine learning","deep learning",  
 "neural network","nlp","natural language","computer vision",  
 "reinforcement learning","generative ai","llm","gpt","chatgpt",  
 "transformer","bert","prompt engineer","prompt engineering"  
 ]  
 text\_cols = [c for c in [  
 "TITLE","TITLE\_CLEAN","BODY",  
 "SKILLS","SKILLS\_NAME",  
 "SPECIALIZED\_SKILLS","SPECIALIZED\_SKILLS\_NAME",  
 "SOFTWARE\_SKILLS","SOFTWARE\_SKILLS\_NAME",  
 "COMMON\_SKILLS","COMMON\_SKILLS\_NAME",  
 "CERTIFICATIONS\_NAME"  
 ] if c in df.columns]  
 combined = (df[text\_cols].astype(str).agg(" ".join, axis=1).str.lower()  
 if text\_cols else pd.Series([""], index=df.index))  
 pattern = re.compile(r"\b(?:%s)\b" % "|".join(re.escape(t) for t in ai\_terms), re.I)  
 df["IS\_AI"] = combined.str.contains(pattern, na=False)  
  
# ---- Pick the best available industry label ----  
ind\_candidates = [  
 "NAICS\_2022\_6\_NAME","NAICS6\_NAME",  
 "NAICS\_2022\_4\_NAME","NAICS4\_NAME",  
 "NAICS\_2022\_2\_NAME","NAICS2\_NAME",  
 "NAICS\_2022\_6","NAICS6"  
]  
IND\_COL = next((c for c in ind\_candidates if c in df.columns), None)  
if IND\_COL is None:  
 raise ValueError("No NAICS/industry name/code columns found.")  
  
# ---- Monthly AI counts per industry ----  
ai = df[df["IS\_AI"]].copy()  
ai\_monthly = (  
 ai.groupby([IND\_COL, "month"])  
 .size()  
 .reset\_index(name="count")  
)  
  
# ---- Compute growth: (last 3-mo avg - first 3-mo avg) / first 3-mo avg ----  
def growth\_row(g):  
 g = g.sort\_values("month")  
 k = min(3, len(g))  
 first = g["count"].iloc[:k].mean()  
 last = g["count"].iloc[-k:].mean()  
 total = g["count"].sum()  
 if k < 2 or first == 0:  
 return pd.Series({"growth\_pct": np.nan, "first\_avg": first, "last\_avg": last, "months": len(g), "total": total})  
 return pd.Series({"growth\_pct": (last - first) / first \* 100.0,  
 "first\_avg": first, "last\_avg": last,  
 "months": len(g), "total": total})  
  
growth\_df = ai\_monthly.groupby(IND\_COL).apply(growth\_row).reset\_index()  
  
# filter out tiny-volume industries to avoid wild % changes  
MIN\_TOTAL = 30  
growth\_df = growth\_df[growth\_df["total"] >= MIN\_TOTAL].dropna(subset=["growth\_pct"])  
  
# pick top movers (adjust top\_n)  
top\_n = 12  
growth\_top = growth\_df.sort\_values("growth\_pct", ascending=False).head(top\_n)  
  
# Save a copy if want a table in the doc  
growth\_top\_rounded = growth\_top.copy()  
growth\_top\_rounded["growth\_pct"] = growth\_top\_rounded["growth\_pct"].round(1)  
growth\_top\_rounded.head(top\_n)

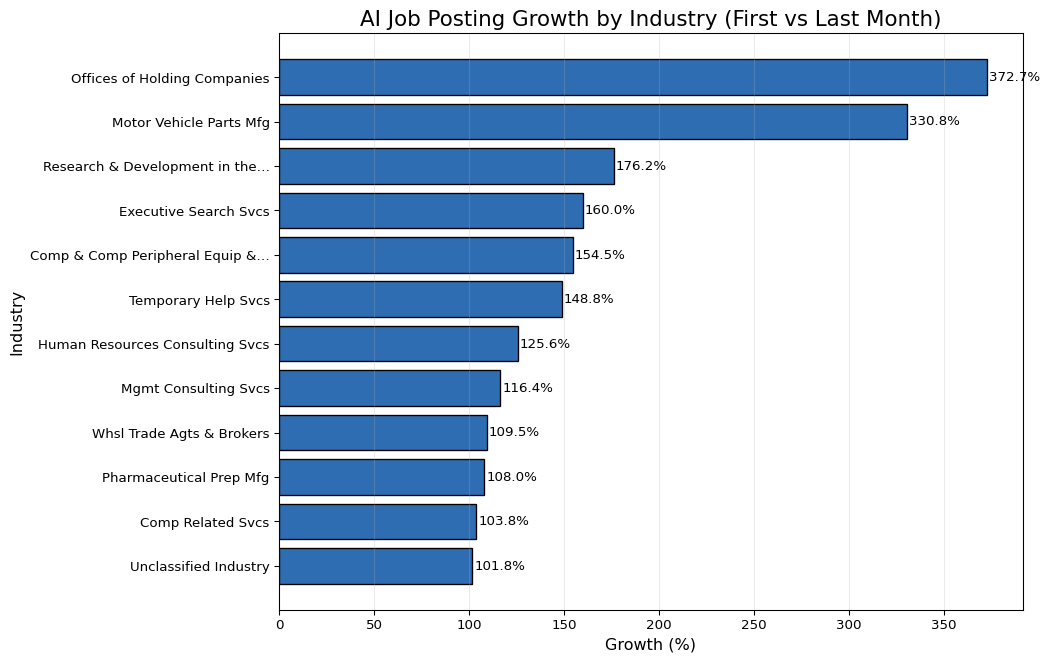
/var/folders/7j/ct705g296ls7nrjh30h9pyg40000gn/T/ipykernel\_71870/672499099.py:67: DeprecationWarning:  
  
DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be excluded from the operation. Either pass `include\_groups=False` to exclude the groupings or explicitly select the grouping columns after groupby to silence this warning.

|  | NAICS\_2022\_6\_NAME | growth\_pct | first\_avg | last\_avg | months | total |
| --- | --- | --- | --- | --- | --- | --- |
| 260 | Offices of Other Holding Companies | 372.7 | 3.666667 | 17.333333 | 5.0 | 56.0 |
| 300 | Other Motor Vehicle Parts Manufacturing | 330.8 | 4.333333 | 18.666667 | 5.0 | 68.0 |
| 358 | Research and Development in the Physical, Engi... | 176.2 | 14.000000 | 38.666667 | 5.0 | 133.0 |
| 137 | Executive Search Services | 160.0 | 6.666667 | 17.333333 | 5.0 | 59.0 |
| 80 | Computer and Computer Peripheral Equipment and... | 154.5 | 3.666667 | 9.333333 | 5.0 | 33.0 |
| 406 | Temporary Help Services | 148.8 | 42.333333 | 105.333333 | 5.0 | 352.0 |
| 189 | Human Resources Consulting Services | 125.6 | 13.000000 | 29.333333 | 5.0 | 96.0 |
| 296 | Other Management Consulting Services | 116.4 | 46.666667 | 101.000000 | 5.0 | 337.0 |
| 427 | Wholesale Trade Agents and Brokers | 109.5 | 7.000000 | 14.666667 | 5.0 | 52.0 |
| 325 | Pharmaceutical Preparation Manufacturing | 108.0 | 8.333333 | 17.333333 | 5.0 | 62.0 |
| 280 | Other Computer Related Services | 103.8 | 70.666667 | 144.000000 | 5.0 | 509.0 |
| 415 | Unclassified Industry | 101.8 | 280.666667 | 566.333333 | 5.0 | 2025.0 |

In this table, we can see AI job postings growing across many industries. A few sectors jump fast but are small, like Other Holding Companies and Motor Vehicle Parts, and sound signals, but the totals are low, so that they may swing. The strongest, faster growth in real volume is in Temporary Help Services, Other Management Consulting, and Other Computer-Related Services; their monthly averages more than doubled. R&D and Executive Search are up too, with mid totals. There is also a growing group called “Unclassified.” This group includes various categories, so we need to use this data carefully.

# AI-Driven Job Growth by Industry

# --- Short, readable labels + plot & save (Option B) -------------------------  
import os, re  
from textwrap import shorten  
import matplotlib.pyplot as plt  
  
# Use existing industry column if defined; otherwise default:  
IND\_COL = IND\_COL if "IND\_COL" in globals() else "NAICS\_2022\_6\_NAME"  
  
def short\_label(s: str) -> str:  
 s = str(s)  
 # Common abbreviations to keep labels compact but clear  
 s = re.sub(r'(?i)\bservices?\b', 'Svcs', s)  
 s = re.sub(r'(?i)\bmanufacturing\b', 'Mfg', s)  
 s = re.sub(r'(?i)\bpreparation\b', 'Prep', s)  
 s = re.sub(r'(?i)\bmanagement\b', 'Mgmt', s)  
 s = re.sub(r'(?i)\bcomputer\b', 'Comp', s)  
 s = re.sub(r'(?i)\bequipment\b', 'Equip', s)  
 s = re.sub(r'(?i)\bwholesale\b', 'Whsl', s)  
 s = re.sub(r'(?i)\bagents?\b', 'Agts', s)  
 s = re.sub(r'(?i)\bscientific\b', 'Sci', s)  
 s = re.sub(r'(?i)\bengineering\b', 'Eng', s)  
 s = re.sub(r'(?i)\band\b', '&', s)  
 s = re.sub(r'(?i)\bother\b\s\*', '', s) # drop leading "Other"  
 # Final safe trim  
 return shorten(s.strip(), width=32, placeholder='…')  
  
plot\_df = growth\_top.copy()  
plot\_df["label"] = plot\_df[IND\_COL].astype(str).map(short\_label)  
  
fig, ax = plt.subplots(figsize=(11, 7))  
bars = ax.barh(plot\_df["label"], plot\_df["growth\_pct"], color="#2F6DB3", edgecolor="black")  
ax.invert\_yaxis() # biggest at top  
  
ax.set\_title("AI Job Posting Growth by Industry (First vs Last Month)", fontsize=16)  
ax.set\_xlabel("Growth (%)", fontsize=12)  
ax.set\_ylabel("Industry", fontsize=12)  
ax.grid(axis="x", alpha=0.25)  
  
# Value labels  
for b in bars:  
 w = b.get\_width()  
 ax.text(  
 w + (1 if w >= 0 else -1),  
 b.get\_y() + b.get\_height() / 2,  
 f"{w:.1f}%",  
 va="center",  
 ha="left" if w >= 0 else "right",  
 fontsize=10,  
 )  
  
plt.tight\_layout()  
os.makedirs("output", exist\_ok=True)  
plt.savefig("output/ai\_industry\_growth\_shortlabels.png", dpi=200, bbox\_inches="tight")  
plt.show()



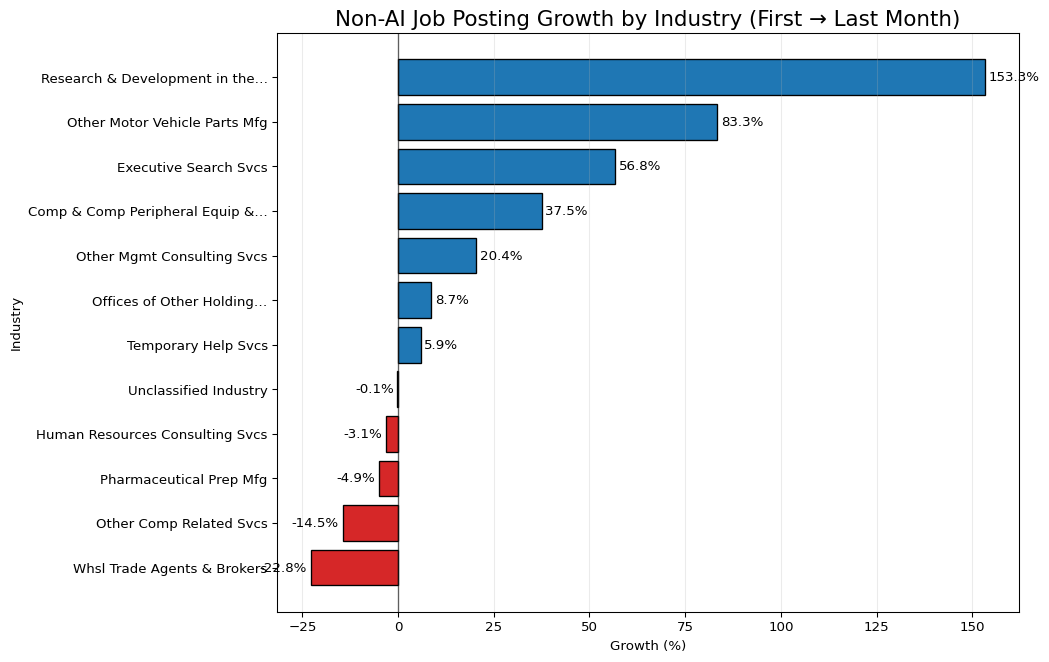
The chart shows that AI job postings are increasing quickly in many industries. The largest increases are in Offices of Holding Companies (about 373%) and Motor Vehicle Parts Manufacturing (about 331%). We also see significant growth in R&D (physical/engineering) (about 176%), Executive Search (about 160%), Computer & Peripheral Equipment (about 155%), and Temporary Help Services (about 149%). Other sectors like HR Consulting, Management Consulting, Wholesale Trade Agents, Pharmaceutical Preparation, Other Computer Related, and Unclassified categories enlarged by slightly more than 100% all together. The growth is mainly distributed among the different sectors, while only some industries exhibit the highest increases.

# Non-AI Job Posting Growth by Industry

import os, re  
import pandas as pd  
from textwrap import shorten  
  
assert "raw\_df" in globals(), "raw\_df must exist (cleaned dataframe)."  
df = raw\_df.copy()  
  
# Pick an industry-name column that exists  
IND\_COL = next((c for c in [  
 "NAICS\_2022\_6\_NAME","NAICS6\_NAME","NAICS\_2022\_4\_NAME","NAICS4\_NAME",  
 "NAICS\_2022\_2\_NAME","NAICS2\_NAME"  
] if c in df.columns), None)  
assert IND\_COL, "No industry/NAICS name column found."  
  
TITLE\_COL = "TITLE\_CLEAN" if "TITLE\_CLEAN" in df.columns else "TITLE"  
assert TITLE\_COL in df.columns, "Need a TITLE or TITLE\_CLEAN column."  
  
# Dates → month  
df["POSTED\_DT"] = pd.to\_datetime(df["POSTED"], errors="coerce")  
df = df.dropna(subset=["POSTED\_DT", IND\_COL])  
df["month"] = df["POSTED\_DT"].dt.to\_period("M").dt.to\_timestamp()  
  
# Detect AI vs non-AI via title keywords  
AI\_TERMS = [  
 "artificial intelligence", r"\bAI\b", "machine learning", r"\bML\b",  
 "deep learning", r"\bLLM\b", r"\bNLP\b", "computer vision",  
 "generative", "chatgpt", r"gpt-\d+", "transformer", "bert",  
 "prompt engineer", "reinforcement learning"  
]  
ai\_pat = re.compile("|".join(AI\_TERMS), flags=re.IGNORECASE)  
if "is\_ai" not in df.columns:  
 df["is\_ai"] = df[TITLE\_COL].astype(str).str.contains(ai\_pat, na=False)  
  
# Use industries from AI-growth visual if available; otherwise pick top 10 by AI postings last month  
if "growth\_top" in globals():  
 target\_inds = [str(x) for x in growth\_top[IND\_COL].dropna().tolist()]  
else:  
 last\_m = df["month"].max()  
 ai\_last = (df[df["is\_ai"] & (df["month"] == last\_m)]  
 .groupby(IND\_COL).size().sort\_values(ascending=False).head(10))  
 target\_inds = ai\_last.index.astype(str).tolist()  
  
# Build non-AI growth (% first→last month) for those industries  
non\_ai = df[(~df["is\_ai"]) & (df[IND\_COL].astype(str).isin(target\_inds))].copy()  
monthly = (non\_ai.groupby([IND\_COL, "month"]).size().reset\_index(name="count"))  
  
first\_last = (monthly.sort\_values("month")  
 .groupby(IND\_COL, as\_index=False)  
 .agg(first=("count","first"), last=("count","last")))  
first\_last = first\_last[first\_last["first"] > 0].copy()  
first\_last["growth\_pct"] = ((first\_last["last"] - first\_last["first"])  
 / first\_last["first"]) \* 100.0  
first\_last = first\_last[first\_last[IND\_COL].astype(str).isin(target\_inds)]  
  
def short\_label(s: str) -> str:  
 s = str(s)  
 s = re.sub(r'(?i)\bservices?\b', 'Svcs', s)  
 s = re.sub(r'(?i)\bmanufacturing\b', 'Mfg', s)  
 s = re.sub(r'(?i)\bmanagement\b', 'Mgmt', s)  
 s = re.sub(r'(?i)\bpreparation\b', 'Prep', s)  
 s = re.sub(r'(?i)\bcomputer\b', 'Comp', s)  
 s = re.sub(r'(?i)\bequipment\b', 'Equip', s)  
 s = re.sub(r'(?i)\bwholesale\b', 'Whsl', s)  
 s = re.sub(r'(?i)\band\b', '&', s)  
 s = re.sub(r'\s+', ' ', s).strip()  
 return shorten(s, width=32, placeholder='…')  
  
non\_ai\_growth\_plot\_df = (first\_last  
 .sort\_values("growth\_pct", ascending=False)  
 .assign(label=lambda d: d[IND\_COL].map(short\_label)))  
  
os.makedirs("output", exist\_ok=True)  
non\_ai\_growth\_plot\_df.head(10) # preview table

|  | NAICS\_2022\_6\_NAME | first | last | growth\_pct | label |
| --- | --- | --- | --- | --- | --- |
| 8 | Research and Development in the Physical, Engi... | 45 | 114 | 153.333333 | Research & Development in the… |
| 6 | Other Motor Vehicle Parts Manufacturing | 12 | 22 | 83.333333 | Other Motor Vehicle Parts Mfg |
| 1 | Executive Search Services | 37 | 58 | 56.756757 | Executive Search Svcs |
| 0 | Computer and Computer Peripheral Equipment and... | 16 | 22 | 37.500000 | Comp & Comp Peripheral Equip &… |
| 5 | Other Management Consulting Services | 162 | 195 | 20.370370 | Other Mgmt Consulting Svcs |
| 3 | Offices of Other Holding Companies | 23 | 25 | 8.695652 | Offices of Other Holding… |
| 9 | Temporary Help Services | 272 | 288 | 5.882353 | Temporary Help Svcs |
| 10 | Unclassified Industry | 2028 | 2025 | -0.147929 | Unclassified Industry |
| 2 | Human Resources Consulting Services | 32 | 31 | -3.125000 | Human Resources Consulting Svcs |
| 7 | Pharmaceutical Preparation Manufacturing | 61 | 58 | -4.918033 | Pharmaceutical Prep Mfg |

import os  
import matplotlib.pyplot as plt  
  
# If prep didn't run in this kernel, rebuild quickly  
if "non\_ai\_growth\_plot\_df" not in globals():  
 # Re-run a minimal rebuild using the same logic  
 assert "raw\_df" in globals(), "raw\_df is required to rebuild plot data."  
 # execute the prep cell above; keeping this here for resilience:  
 # (We simply import the name if it exists; otherwise advise to run prep.)  
 raise AssertionError("Run the prep chunk first to create non\_ai\_growth\_plot\_df.")  
  
dfp = non\_ai\_growth\_plot\_df.copy()  
  
if dfp.empty:  
 print("non\_ai\_growth\_plot\_df is empty—nothing to plot. "  
 "Check that selected industries have non-AI rows in the first/last month.")  
else:  
 os.makedirs("output", exist\_ok=True)  
 colors = dfp["growth\_pct"].ge(0).map({True: "#1f77b4", False: "#d62728"}).to\_numpy()  
  
 fig, ax = plt.subplots(figsize=(11, 7))  
 bars = ax.barh(dfp["label"], dfp["growth\_pct"], color=colors, edgecolor="black")  
  
 ax.axvline(0, color="black", linewidth=1, alpha=0.6)  
 ax.invert\_yaxis()  
 ax.set\_title("Non-AI Job Posting Growth by Industry (First → Last Month)", fontsize=16)  
 ax.set\_xlabel("Growth (%)")  
 ax.set\_ylabel("Industry")  
 ax.grid(axis="x", alpha=0.25)  
  
 for b, v in zip(bars, dfp["growth\_pct"]):  
 ax.text(  
 v + (1 if v >= 0 else -1),  
 b.get\_y() + b.get\_height() / 2,  
 f"{v:.1f}%",  
 va="center",  
 ha="left" if v >= 0 else "right",  
 fontsize=10,  
 )  
  
 plt.tight\_layout()  
 plt.savefig("output/non\_ai\_growth\_pct\_in\_ai\_growth\_industries\_colored.png",  
 dpi=200, bbox\_inches="tight")  
 plt.show()



The Non-AI job posting growth chart shows the following:

* The biggest gains are in Research & Development (~153%) and Motor Vehicle Parts (~83%). Executive Search (~57%) and Computer/Peripheral Equipment (~38%) also grew well. Management Consulting (~20%) is up, while Offices of Holding Companies (~9%) and Temporary Help (~6%) show small growth.
* A few areas are slipping: Wholesale Trade Agents & Brokers (~-23%), Other Computer-Related Services (~-15%), Pharmaceutical Prep (~-5%), and HR Consulting (~-3%) declined; Unclassified is flat.

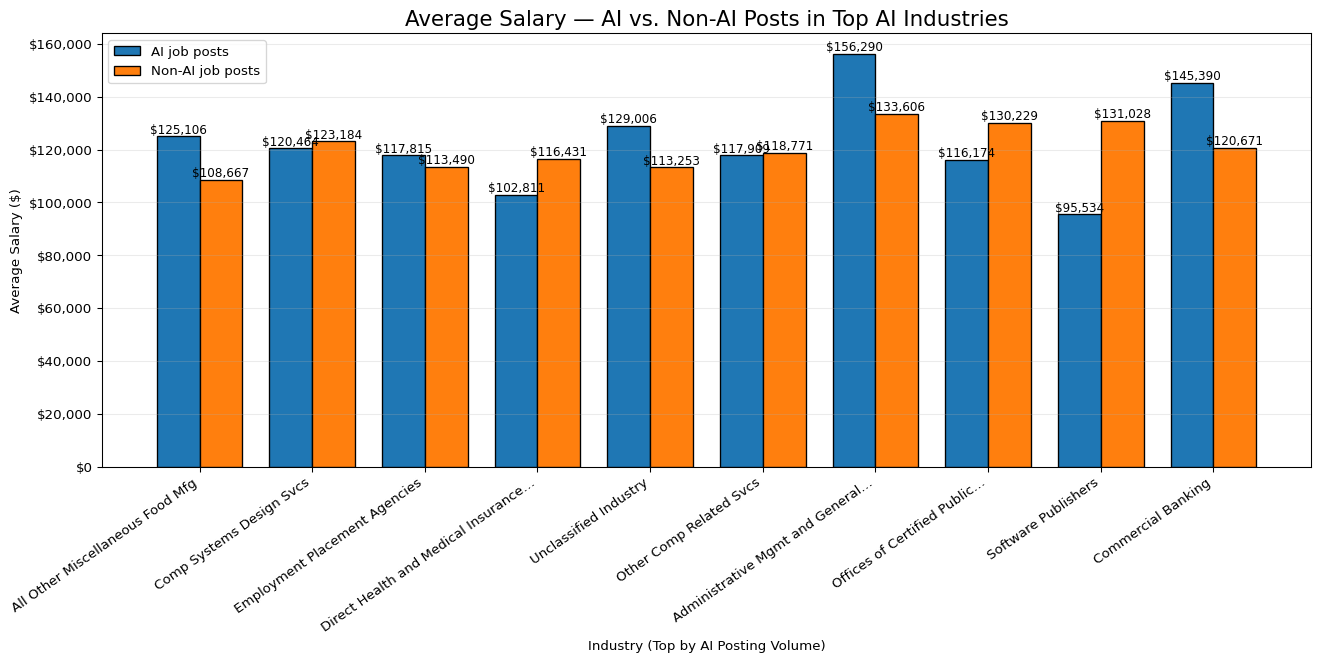
This means for us that non-AI roles are still growing, just not as fast as AI roles. The non-AI opportunities we want more of should be targeted to R&D labs, automotive suppliers, executive search firms, and hardware/peripherals. Outreach to these fields will guarantee more non-AI opportunities. The areas of wholesale, general computer, and pharma-related services, HR consulting would be where one could select with more discrimination since the demand for these practices is on the decline.

# AI-powered roles vs non-ai roles salary prep

import os, re  
import numpy as np  
import pandas as pd  
  
assert "raw\_df" in globals(), "raw\_df (cleaned dataframe) must exist."  
  
df = raw\_df.copy()  
  
# --- Pick columns robustly ---  
TITLE\_COL = "TITLE\_CLEAN" if "TITLE\_CLEAN" in df.columns else "TITLE"  
assert TITLE\_COL in df.columns, "Need a TITLE or TITLE\_CLEAN column."  
  
IND\_COL = next((c for c in [  
 "NAICS\_2022\_6\_NAME","NAICS6\_NAME","NAICS\_2022\_4\_NAME","NAICS4\_NAME",  
 "NAICS\_2022\_2\_NAME","NAICS2\_NAME","LIGHTCAST\_SECTORS\_NAME"  
] if c in df.columns), None)  
assert IND\_COL, "No industry/NAICS name column found."  
  
# Salary: prefer SALARY; otherwise average of FROM/TO if present  
sal = pd.to\_numeric(df.get("SALARY", np.nan), errors="coerce")  
if sal.isna().all() and {"SALARY\_FROM","SALARY\_TO"} <= set(df.columns):  
 s\_from = pd.to\_numeric(df["SALARY\_FROM"], errors="coerce")  
 s\_to = pd.to\_numeric(df["SALARY\_TO"], errors="coerce")  
 sal = (s\_from + s\_to) / 2  
df["salary\_num"] = pd.to\_numeric(sal, errors="coerce")  
  
# keep positive salaries only  
df = df[df["salary\_num"] > 0].copy()  
  
# Optional: cap extreme outliers so a few posts don't dominate the mean  
q\_low, q\_hi = df["salary\_num"].quantile([0.01, 0.99])  
df["salary\_num"] = df["salary\_num"].clip(q\_low, q\_hi)  
  
# --- Tag AI vs non-AI using title keywords ---  
AI\_TERMS = [  
 "artificial intelligence", r"\bAI\b", "machine learning", r"\bML\b",  
 "deep learning", r"\bLLM\b", r"\bNLP\b", "computer vision",  
 "generative", "chatgpt", r"gpt-\d+", "transformer", r"\bbert\b",  
 "prompt engineer", "reinforcement learning", "data scientist", "ai engineer"  
]  
ai\_pat = re.compile("|".join(AI\_TERMS), flags=re.IGNORECASE)  
df["is\_ai"] = df[TITLE\_COL].astype(str).str.contains(ai\_pat, na=False)  
  
# --- Find top industries by AI posting count (choose N)  
N = 10  
ai\_counts = (df[df["is\_ai"]]  
 .groupby(IND\_COL, dropna=True)  
 .size()  
 .sort\_values(ascending=False)  
 .head(N)  
 .rename("ai\_posts")  
 .reset\_index())  
top\_inds = ai\_counts[IND\_COL].astype(str).tolist()  
  
# --- Compute average salary (AI vs non-AI) for those industries  
subset = df[df[IND\_COL].astype(str).isin(top\_inds)].copy()  
grp = subset.groupby([IND\_COL, "is\_ai"])["salary\_num"].agg(["mean", "count"]).reset\_index()  
  
ai\_part = grp[grp["is\_ai"]].rename(columns={"mean":"avg\_ai\_salary", "count":"ai\_postings"})  
non\_part = grp[~grp["is\_ai"]].rename(columns={"mean":"avg\_nonai\_salary", "count":"non\_ai\_postings"})  
  
ai\_vs\_trad\_industry\_salary = (  
 ai\_counts[[IND\_COL, "ai\_posts"]] # preserves AI-based ordering  
 .merge(ai\_part[[IND\_COL, "avg\_ai\_salary", "ai\_postings"]], on=IND\_COL, how="left")  
 .merge(non\_part[[IND\_COL, "avg\_nonai\_salary", "non\_ai\_postings"]], on=IND\_COL, how="left")  
)  
  
# Clean up & preview  
ai\_vs\_trad\_industry\_salary = ai\_vs\_trad\_industry\_salary.fillna(0)  
os.makedirs("output", exist\_ok=True)  
ai\_vs\_trad\_industry\_salary.head(N)

|  | NAICS\_2022\_6\_NAME | ai\_posts | avg\_ai\_salary | ai\_postings | avg\_nonai\_salary | non\_ai\_postings |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | All Other Miscellaneous Food Manufacturing | 124 | 125106.451613 | 124 | 108667.836735 | 49 |
| 1 | Computer Systems Design Services | 81 | 120464.666667 | 81 | 123184.301252 | 4073 |
| 2 | Employment Placement Agencies | 72 | 117815.555556 | 72 | 113490.867107 | 4244 |
| 3 | Direct Health and Medical Insurance Carriers | 67 | 102811.940299 | 67 | 116431.746753 | 1386 |
| 4 | Unclassified Industry | 57 | 129006.070175 | 57 | 113253.334609 | 9148 |
| 5 | Other Computer Related Services | 53 | 117909.245283 | 53 | 118771.586707 | 1309 |
| 6 | Administrative Management and General Manageme... | 52 | 156290.384615 | 52 | 133606.945132 | 4447 |
| 7 | Offices of Certified Public Accountants | 48 | 116174.354167 | 48 | 130229.120365 | 1645 |
| 8 | Software Publishers | 35 | 95534.285714 | 35 | 131028.541624 | 973 |
| 9 | Commercial Banking | 34 | 145390.588235 | 34 | 120671.148515 | 1919 |

import os, re  
import matplotlib.pyplot as plt  
from textwrap import shorten  
from matplotlib.ticker import FuncFormatter  
  
assert "ai\_vs\_trad\_industry\_salary" in globals(), "Run the prep chunk first."  
  
dfp = ai\_vs\_trad\_industry\_salary.copy()  
if dfp.empty:  
 print("No data to plot after filtering. Check AI detection or industry column.")  
else:  
 # shorten very long industry labels  
 def short\_label(s: str) -> str:  
 s = str(s)  
 s = re.sub(r'(?i)\bservices?\b', 'Svcs', s)  
 s = re.sub(r'(?i)\bmanufacturing\b', 'Mfg', s)  
 s = re.sub(r'(?i)\bmanagement\b', 'Mgmt', s)  
 s = re.sub(r'(?i)\bcomputer\b', 'Comp', s)  
 s = re.sub(r'(?i)\bequipment\b', 'Equip', s)  
 s = re.sub(r'\s+', ' ', s).strip()  
 return shorten(s, width=36, placeholder='…')  
  
 dfp["label"] = dfp[IND\_COL].map(short\_label)  
  
 x = range(len(dfp))  
 w = 0.38  
  
 fig, ax = plt.subplots(figsize=(14, 7))  
 b1 = ax.bar([i - w/2 for i in x], dfp["avg\_ai\_salary"], width=w, label="AI job posts", edgecolor="black")  
 b2 = ax.bar([i + w/2 for i in x], dfp["avg\_nonai\_salary"], width=w, label="Non-AI job posts", edgecolor="black")  
  
 ax.set\_title("Average Salary — AI vs. Non-AI Posts in Top AI Industries", fontsize=16)  
 ax.set\_xlabel("Industry (Top by AI Posting Volume)")  
 ax.set\_ylabel("Average Salary ($)")  
 ax.set\_xticks(list(x))  
 ax.set\_xticklabels(dfp["label"], rotation=35, ha="right")  
 ax.yaxis.set\_major\_formatter(FuncFormatter(lambda v, pos: f"${int(v):,}"))  
 ax.grid(axis="y", alpha=0.25)  
 ax.legend()  
  
 # annotate bars  
 for bars in (b1, b2):  
 for p in bars:  
 h = p.get\_height()  
 if h > 0:  
 ax.text(p.get\_x()+p.get\_width()/2, h, f"${int(h):,}",  
 ha="center", va="bottom", fontsize=9)  
  
 plt.tight\_layout()  
 os.makedirs("output", exist\_ok=True)  
 plt.savefig("output/ai\_vs\_nonai\_avg\_salary\_by\_industry.png", dpi=200, bbox\_inches="tight")  
 plt.show()



In this graph, we compared pay for AI and non-AI roles across industries. In most of these sectors, AI jobs pay more; significant gaps show up in Administrative Management & General Services, Commercial Banking, Unclassified, and Food Manufacturing. A few places flip the pattern: Software Publishers, Offices of Certified Public Accountants, Health/Medical Insurance, and Computer Systems Design show higher pay for non-AI roles (or pay that is almost the same in Other Computer-Related Services). For our search, we should target AI roles in industries with clear premium pay, and be careful in the few sectors where AI pay trails non-AI; those roles may be more junior or support.

# Market Takeaways from our EDA

AI does not affect every field the same. From our EDA, the fields that use AI consulting, computer services, R&D, and parts of banking show rising postings and often higher pay, so jobs look safer there. In contrast, areas with flat or falling non-AI postings, wholesale trade agents, some general computer services, pharma prep, and HR consulting look less secure unless they adopt AI tools. Routine, repeat tasks are most at risk, while roles that mix tech and people skills (analysis, client work, project work) look safer.

Looking across industries, we see growth where AI is being put to work: other computer-related services, management consulting, temporary help services, R&D, and commercial banking. Meanwhile, we see possible displacement or slowdown on the non-AI side in wholesale trade agents, parts of computer services, pharma prep, and HR consulting. In short, sectors leaning into AI grow; those that don’t may shrink or re-scope roles.

Compared with traditional paths like mechanical engineering, farming, and retail, AI-powered roles are growing faster and usually pay more, according to our charts. Traditional fields still matter, but many tasks there now use AI (e.g., predictive maintenance in mechanical, precision ag in farming, demand forecasting in retail). People who add data and AI skills to their core trade tend to get better options and higher pay.

Finally, we see new job titles emerging: AI/ML Engineer, MLOps/AI Ops, AI Product Manager, Prompt Engineer, Data/ML Analyst, AI Solutions Consultant, and AI Ethics/Governance, along with more analytics-heavy PM and consulting roles. Overall, where AI is adopted, jobs shift toward analysis, tools, and delivery, and hiring grows. Where AI is ignored, roles risk shrinking. Our best move is to build AI and delivery skills (CS basics, operations, project management) and target the growing sectors.