Machine Learning Methods

Evaluating Personal Job Market Prospects in 2024

Anu Sharma, Cindy Guzman, Gavin Boss

2025-10-11

# Overview

The analysis examines trends in Business Analytics, Data Science, and Machine Learning job postings, with a focus on the **skills required** for these roles. The study evaluates how varying skill combinations influence salary levels, remote work availability, and career progression pathways.

This analysis applies three machine learning approaches to job posting data: clustering to group roles by skill requirements, regression to examine how skills and experience influence salary, and classification to distinguish ML/Data Science positions from Business Analytics and other jobs. Using 25 technical skills along with experience and remote work indicators, the analysis shows that Business Analytics dominates the market (35% of roles), while ML and DS remain smaller but specialized segments. Results highlight that experience is the strongest salary driver, jobs fall into six clear clusters with different pay and remote work patterns, and BA, ML, and DS roles each display distinct skill signatures that make them easy to differentiate

# Data Loading and Setup

The analysis starts by loading the Lightcast job postings dataset and identifying relevant skill columns. The dataset contains comprehensive information about job postings including titles, salaries, required skills, and other job characteristics.

import pandas as pd  
import numpy as np  
import plotly.express as px  
import plotly.graph\_objects as go  
from plotly.subplots import make\_subplots  
import plotly.io as pio  
import json  
import re  
from collections import Counter  
  
pio.templates.default = "plotly\_white"  
pio.renderers.default = "notebook"  
  
# Load data from csv  
df = pd.read\_csv("data/lightcast\_job\_postings.csv", low\_memory=False)  
print(f"Dataset loaded: {len(df):,} rows, {len(df.columns)} columns")  
  
# print(df.head())

Dataset loaded: 72,498 rows, 131 columns

## Important Skills columns

The dataset contains multiple skill-related columns. After examining the schema, the columns ‘SKILLS\_NAME’, ‘SOFTWARE\_SKILLS\_NAME’ and ‘SPECIALIZED\_SKILLS\_NAME’ provide the most detailed skill information for this analysis. These columns list the specific technical skills mentioned in each job posting.

# Skills Data Preprocessing

The next step involves filtering the data to include only records with valid salary and title information. Then, binary features are created for 25 key technical skills covering ML, Data Science, and Business Analytics domains to enable machine learning analysis.

# Apply filters  
df\_filtered = df.dropna(subset=['SALARY', 'TITLE'])  
  
# Convert salary to numeric and filter  
df\_filtered['SALARY'] = pd.to\_numeric(df\_filtered['SALARY'], errors='coerce')  
df\_filtered = df\_filtered[df\_filtered['SALARY'] > 0]  
  
print(f"Records after filtering: {len(df\_filtered):,}")  
  
df\_skills = df\_filtered.copy()  
  
# Focus on key Business Analytics/ML/Data Science skills. Key skills for  
# BA/ML/DS roles identified manually.  
key\_skills = [  
 'Python (Programming Language)',  
 'R (Programming Language)',  
 'SQL (Programming Language)',  
 'Machine Learning',  
 'Data Science',  
 'Data Analysis',  
 'Statistics',  
 'Artificial Intelligence',  
 'TensorFlow',  
 'PyTorch (Machine Learning Library)',  
 'Pandas (Python Package)',  
 'NumPy (Python Package)',  
 'Scikit-Learn (Python Package)',  
 'Big Data',  
 'Apache Spark',  
 'Apache Hadoop',  
 'Amazon Web Services',  
 'Microsoft Azure',  
 'Google Cloud Platform (Gcp)',  
 'Data Visualization',  
 'Tableau (Business Intelligence Software)',  
 'Power BI',  
 'Natural Language Processing (NLP)',  
 'Computer Vision',  
 'Deep Learning'  
 ]  
  
print(f"Using focused {len(key\_skills)} BA/ML/DS technical skills for analysis")  
  
# Create binary features for each key skill.  
for skill in key\_skills:  
 # Clean skill name for column naming  
 # Eg: R (Programming Language) --> has\_r\_programming\_language  
 skill\_col\_name = f'has\_{skill.lower().replace(" ", "\_").replace("-", "\_").replace("(", "").replace(")", "")}'  
  
  
 df\_skills[skill\_col\_name] = (  
 df\_skills['SKILLS\_NAME'].str.contains(skill, case=False, na=False, regex=False) |  
 df\_skills['SOFTWARE\_SKILLS\_NAME'].str.contains(skill, case=False, na=False, regex=False) |  
 df\_skills['SPECIALIZED\_SKILLS\_NAME'].str.contains(skill, case=False, na=False, regex=False)  
 ).astype(int)  
  
print("Binary skill features created")  
  
# Create ML/DS role indicator using focused skills  
core\_ml\_skills = [  
 'has\_machine\_learning', 'has\_artificial\_intelligence', 'has\_tensorflow', 'has\_pytorch\_machine\_learning\_library',  
 'has\_deep\_learning', 'has\_natural\_language\_processing\_nlp', 'has\_computer\_vision'  
]  
  
core\_ds\_skills = [  
 'has\_python\_programming\_language', 'has\_r\_programming\_language', 'has\_statistics',  
 'has\_data\_science', 'has\_pandas\_python\_package', 'has\_numpy\_python\_package',  
 'has\_scikit\_learn\_python\_package', 'has\_big\_data'  
]  
  
core\_ba\_skills = [  
 'has\_data\_analysis', 'has\_data\_visualization', 'has\_sql\_programming\_language',  
 'has\_tableau\_business\_intelligence\_software', 'has\_power\_bi'  
]  
  
# Role indicators  
# ML roles are straightforward.  
df\_skills['is\_ml\_role'] = (  
 (df\_skills[core\_ml\_skills].sum(axis=1) > 0)  
).astype(int)  
  
# R language is primarily associated with Data Science field. So,  
# if job requires R language or if it has more than one data science  
# skills then it is considered DS role.  
df\_skills['is\_ds\_role'] = (  
 df\_skills['has\_r\_programming\_language'] == 1 | (df\_skills[core\_ds\_skills].sum(axis=1) > 1)  
).astype(int)  
  
# Business Analytics roles typically require SQL, visualization tools (Tableau, Power BI)  
# and data analysis capabilities. If job has more than two BA skills, consider it a BA role.  
df\_skills['is\_ba\_role'] = (  
 df\_skills[core\_ba\_skills].sum(axis=1) >= 2  
).astype(int)  
  
# Remote work indicator  
df\_skills['is\_remote'] = df\_skills['REMOTE\_TYPE'].fillna(0).astype(int)  
df\_skills['experience\_years'] = df\_skills['MIN\_YEARS\_EXPERIENCE'].fillna(0)  
  
df\_final = df\_skills  
print(f"Final dataset size: {len(df\_final):,}")  
print(f"ML roles identified: {df\_final['is\_ml\_role'].sum():,}")  
print(f"Data Science roles identified: {df\_final['is\_ds\_role'].sum():,}")  
print(f"Business Analytics roles identified: {df\_final['is\_ba\_role'].sum():,}")

Records after filtering: 30,808  
Using focused 25 BA/ML/DS technical skills for analysis

Binary skill features created  
Final dataset size: 30,808  
ML roles identified: 3,226  
Data Science roles identified: 2,877  
Business Analytics roles identified: 10,831

For each of the 25 key skills, a binary indicator variable is created (1 if the skill is mentioned, 0 otherwise). This transforms the text skill data into numerical features suitable for machine learning models.

## Role Classification Logic

Three role categories are identified based on technical skills:

* **ML roles**: Require advanced ML/AI skills like TensorFlow, PyTorch, Deep Learning, NLP, Computer Vision
* **Data Science roles**: Require R programming, Python with Statistics, or multiple data science tools (Pandas, NumPy, Scikit-learn)
* **Business Analytics roles**: Require SQL, data analysis, visualization tools (Tableau, Power BI), typically 2+ BA skills

The analysis examines how these specialized skills impact salary and career opportunities. Machine learning models are used to find patterns that can guide job seekers in choosing which skills to develop.

# Feature Engineering for ML

Before building models, the dataset is prepared by selecting relevant columns. This includes the salary (target variable), skill indicators, remote work status, and experience years.

# Just prepare the modeling dataset  
modeling\_cols = ['SALARY', 'is\_ml\_role', 'is\_ds\_role', 'is\_ba\_role', 'is\_remote', 'experience\_years'] + \  
 [col for col in df\_final.columns if col.startswith('has\_')]  
  
df\_modeling = df\_final[modeling\_cols].copy()  
  
print("Features for modeling:")  
print(f"Dataset shape: {df\_modeling.shape}")  
print(f"Columns: {list(df\_modeling.columns)}")  
print(f"Missing values: {df\_modeling.isnull().sum().sum()}")

Features for modeling:  
Dataset shape: (30808, 31)  
Columns: ['SALARY', 'is\_ml\_role', 'is\_ds\_role', 'is\_ba\_role', 'is\_remote', 'experience\_years', 'has\_python\_programming\_language', 'has\_r\_programming\_language', 'has\_sql\_programming\_language', 'has\_machine\_learning', 'has\_data\_science', 'has\_data\_analysis', 'has\_statistics', 'has\_artificial\_intelligence', 'has\_tensorflow', 'has\_pytorch\_machine\_learning\_library', 'has\_pandas\_python\_package', 'has\_numpy\_python\_package', 'has\_scikit\_learn\_python\_package', 'has\_big\_data', 'has\_apache\_spark', 'has\_apache\_hadoop', 'has\_amazon\_web\_services', 'has\_microsoft\_azure', 'has\_google\_cloud\_platform\_gcp', 'has\_data\_visualization', 'has\_tableau\_business\_intelligence\_software', 'has\_power\_bi', 'has\_natural\_language\_processing\_nlp', 'has\_computer\_vision', 'has\_deep\_learning']  
Missing values: 0

The modeling dataset now contains binary skill features, experience, remote work indicator, and salary information. This structured format allows application of various machine learning techniques.

# Unsupervised Learning:

## KMeans Clustering Based on Skills

The first machine learning approach uses KMeans clustering to discover natural groupings in the job market. This unsupervised technique groups jobs with similar skill profiles together, without using salary information. The goal is to see if jobs naturally segment into distinct categories based on their requirements.

from sklearn.cluster import KMeans  
from sklearn.preprocessing import StandardScaler, LabelEncoder  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.ensemble import RandomForestRegressor, RandomForestClassifier  
from sklearn.linear\_model import LogisticRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score, accuracy\_score, f1\_score, confusion\_matrix, classification\_report  
  
# Prepare features for clustering using skills and other features  
skill\_feature\_cols = [col for col in df\_modeling.columns if col.startswith('has\_')]  
print(f"Available skill features: {len(skill\_feature\_cols)}")  
  
# Base clustering features  
clustering\_features = skill\_feature\_cols + ['experience\_years', 'is\_remote']  
  
# Encode ONET and NAICS6.  
le\_onet = LabelEncoder()  
df\_modeling['onet\_encoded'] = le\_onet.fit\_transform(df\_final['ONET'].fillna('Unknown'))  
clustering\_features.append('onet\_encoded')  
  
le\_naics = LabelEncoder()  
df\_modeling['naics\_encoded'] = le\_naics.fit\_transform(df\_final['NAICS6'].fillna('Unknown'))  
clustering\_features.append('naics\_encoded')  
  
# Prepare clustering data  
X\_cluster = df\_modeling[clustering\_features].fillna(0)  
  
# Scale features  
scaler\_cluster = StandardScaler()  
X\_cluster\_scaled = scaler\_cluster.fit\_transform(X\_cluster)  
  
# KMeans clustering  
kmeans = KMeans(n\_clusters=6, random\_state=42, n\_init=10)  
clusters = kmeans.fit\_predict(X\_cluster\_scaled)  
df\_modeling['cluster'] = clusters  
  
# print("Skills based clustering completed")  
# print("Cluster centers:")  
# for i, center in enumerate(kmeans.cluster\_centers\_):  
# print(f"Cluster {i}: {center}")

Available skill features: 25

The clustering model groups similar jobs together using skill patterns, experience requirements, and job characteristics. The algorithm assigns each job to one of 6 clusters. Now the characteristics of each cluster can be examined to understand what makes them distinct.

# Analyze clustering.  
cluster\_summary = df\_modeling.groupby('cluster').agg({  
 'SALARY': ['count', 'mean'],  
 'is\_ml\_role': 'mean',  
 'is\_ds\_role': 'mean',  
 'is\_ba\_role': 'mean',  
 'is\_remote': 'mean',  
 'experience\_years': 'mean'  
}).round(2)  
  
cluster\_summary.columns = ['count', 'avg\_salary', 'ml\_role\_pct', 'ds\_role\_pct', 'ba\_role\_pct',  
 'remote\_percentage', 'avg\_experience']  
cluster\_summary = cluster\_summary.reset\_index()  
  
# Compute combined BA/ML/DS percentage on-the-fly  
# A job has BA/ML/DS if it has any of the three role types  
cluster\_summary['ml\_ds\_ba\_combined\_pct'] = cluster\_summary.apply(  
 lambda row: ((df\_modeling[df\_modeling['cluster'] == row['cluster']][['is\_ml\_role', 'is\_ds\_role', 'is\_ba\_role']].sum(axis=1) > 0).mean()),  
 axis=1  
).round(2)  
  
print("Skills based Cluster Summary:")  
print(cluster\_summary)  
  
# Visualize cluster characteristics.  
fig = make\_subplots(  
 rows=2, cols=3,  
 subplot\_titles=('Cluster Size', 'Average Salary', 'BA/ML/DS Role %',  
 'Remote Work %', 'Avg Experience', 'Salary Distribution'),  
 specs=[[{"type": "bar"}, {"type": "bar"}, {"type": "bar"}],  
 [{"type": "bar"}, {"type": "bar"}, {"type": "scatter"}]]  
)  
  
fig.add\_trace(go.Bar(x=cluster\_summary['cluster'], y=cluster\_summary['count'], name="Count"), row=1, col=1)  
fig.add\_trace(go.Bar(x=cluster\_summary['cluster'], y=cluster\_summary['avg\_salary'], name="Avg Salary"), row=1, col=2)  
fig.add\_trace(go.Bar(x=cluster\_summary['cluster'], y=cluster\_summary['ml\_role\_pct'], name="ML %"), row=1, col=3)  
fig.add\_trace(go.Bar(x=cluster\_summary['cluster'], y=cluster\_summary['ds\_role\_pct'], name="DS %"), row=1, col=3)  
fig.add\_trace(go.Bar(x=cluster\_summary['cluster'], y=cluster\_summary['ba\_role\_pct'], name="BA %"), row=1, col=3)  
fig.add\_trace(go.Bar(x=cluster\_summary['cluster'], y=cluster\_summary['remote\_percentage'], name="Remote %"), row=2, col=1)  
fig.add\_trace(go.Bar(x=cluster\_summary['cluster'], y=cluster\_summary['avg\_experience'], name="Experience"), row=2, col=2)  
  
# Salary distribution by cluster.  
fig.add\_trace(  
 go.Scatter(  
 x=df\_modeling['cluster'],  
 y=df\_modeling['SALARY'],  
 mode='markers',  
 opacity=0.6,  
 name="Jobs"  
 ),  
 row=2, col=3  
)  
  
fig.update\_layout(  
 height=650,  
 showlegend=False,  
 template="plotly\_white",  
 title={  
 'text': "Skills-Based KMeans Clustering Results",  
 'y': 0.98,  
 'x': 0.5,  
 'xanchor': 'center',  
 'yanchor': 'top'  
 },  
 margin=dict(t=80)  
)  
fig.show()

Skills based Cluster Summary:  
 cluster count avg\_salary ml\_role\_pct ds\_role\_pct ba\_role\_pct \  
0 0 583 139707.42 0.60 0.26 0.70   
1 1 10189 144796.54 0.14 0.00 0.04   
2 2 13313 100969.83 0.01 0.01 0.28   
3 3 6573 108557.35 0.17 0.39 0.95   
4 4 77 140001.35 1.00 0.32 0.31   
5 5 73 117793.86 0.55 0.32 0.93   
  
 remote\_percentage avg\_experience ml\_ds\_ba\_combined\_pct   
0 0.44 4.45 0.90   
1 0.25 7.80 0.17   
2 0.39 2.00 0.29   
3 0.48 3.27 0.99   
4 0.34 4.23 1.00   
5 0.56 3.01 0.96

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

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

### Insights from KMeans Clustering

The clustering analysis grouped jobs based on their skill requirements and characteristics. The analysis identified 6 distinct job clusters, each with different salary levels, remote work availability, and skill profiles.

**Key Findings:**

* **Business Analytics dominates**: 10,831 BA roles vs. 3,226 ML and 2,877 DS
* **Cluster 0 (583 jobs, $140K):** High-skill hybrid (60% ML, 26% DS, 70% BA)
* **Cluster 1 (10,189 jobs, $145K):** Mostly general tech, only 17% BA/DS/ML, highest pay
* **Cluster 2 (13,313 jobs, $101K):** Entry-level, lowest experience (2 yrs), BA-focused (28%)
* **Cluster 3 (6,573 jobs, $109K):** BA-heavy (95%) with DS overlap (39%)
* **Cluster 4 (77 jobs, $140K):** Pure ML specialists (100% ML), niche but high-paying
* **Cluster 5 (73 jobs, $118K):** Hybrid roles (96% BA/DS/ML), most remote-friendly (56%)
* **Remote work:** 25%–56% across clusters
* **Experience:** 2.0–7.8 years, showing clear career progression

**Career Implications:**

* **Most opportunities:** Business Analytics (SQL, Tableau, Power BI, visualization)
* **Highest pay + volume:** Cluster 1 ($145K, 10K+ jobs) — general tech roles
* **Entry path:** Cluster 2 ($101K, 13K jobs) — BA-focused, lowest experience needed
* **BA-focused growth:** Cluster 3 ($109K) — strong BA demand with DS hybrid edge
* **Specialist track:** Cluster 4 ($140K) — pure ML, fewer jobs but high pay
* **Hybrid advantage:** Cluster 0 ($140K) and Cluster 5 ($118K, 56% remote) — multi-skill roles with flexibility

# Supervised Learning:

## Multiple Regression

The second approach uses supervised learning to predict salary based on skills and experience. Two regression models are trained: Multiple Linear Regression and Random Forest. This analysis identifies which skills and factors most strongly influence compensation.

# Regression features.  
# Focus on skills (not role labels) to understand how skills directly affect salary  
regression\_features = skill\_feature\_cols + ['experience\_years', 'is\_remote']  
  
# Preparing regression data using salary as the target variable  
X\_reg = df\_modeling[regression\_features].fillna(0)  
y\_reg = df\_modeling['SALARY']  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_reg, y\_reg, test\_size=0.2, random\_state=42)  
  
print(f"Training set size: {len(X\_train):,}")  
print(f"Test set size: {len(X\_test):,}")  
  
# Scale features  
scaler\_reg = StandardScaler()  
X\_train\_scaled = scaler\_reg.fit\_transform(X\_train)  
X\_test\_scaled = scaler\_reg.transform(X\_test)  
  
# Multiple Linear Regression  
lr = LinearRegression()  
lr.fit(X\_train\_scaled, y\_train)  
  
# Random Forest Regression  
rf\_reg = RandomForestRegressor(n\_estimators=100, random\_state=42)  
rf\_reg.fit(X\_train\_scaled, y\_train)  
  
print("Skills based regression models training completed")  
  
# Regression statistics for Multiple linear regression  
y\_train\_pred = lr.predict(X\_train\_scaled)  
  
# Residuals and RSS  
residuals = y\_train - y\_train\_pred  
rss = np.sum(residuals\*\*2)  
n = len(y\_train)  
k = len(regression\_features)  
  
# Model statistics for Multiple linear regression  
mse\_lr = rss / n  
rmse\_train\_lr = np.sqrt(mse\_lr)  
r2\_train\_lr = r2\_score(y\_train, y\_train\_pred)  
adj\_r2\_lr = 1 - (1 - r2\_train\_lr) \* (n - 1) / (n - k - 1)  
  
# AIC and BIC for Multiple linear regression  
aic\_lr = n \* np.log(rss / n) + 2 \* k  
bic\_lr = n \* np.log(rss / n) + k \* np.log(n)  
log\_likelihood\_lr = -aic\_lr/2 + k  
  
# Random Forest statistics  
y\_train\_pred\_rf = rf\_reg.predict(X\_train\_scaled)  
r2\_train\_rf = r2\_score(y\_train, y\_train\_pred\_rf)  
rmse\_train\_rf = np.sqrt(mean\_squared\_error(y\_train, y\_train\_pred\_rf))  
residuals\_rf = y\_train - y\_train\_pred\_rf  
rss\_rf = np.sum(residuals\_rf\*\*2)  
  
# Test set performance  
y\_test\_pred\_lr = lr.predict(X\_test\_scaled)  
y\_test\_pred\_rf = rf\_reg.predict(X\_test\_scaled)  
r2\_test\_lr = r2\_score(y\_test, y\_test\_pred\_lr)  
r2\_test\_rf = r2\_score(y\_test, y\_test\_pred\_rf)  
rmse\_test\_lr = np.sqrt(mean\_squared\_error(y\_test, y\_test\_pred\_lr))  
rmse\_test\_rf = np.sqrt(mean\_squared\_error(y\_test, y\_test\_pred\_rf))  
  
# Regression statistics table  
print("\n=== REGRESSION MODEL STATISTICS ===\n")  
  
regression\_stats = pd.DataFrame({  
 'Statistic': [  
 'Intercept',  
 'Number of Features',  
 'Number of Observations (Train)',  
 'R-squared (Training)',  
 'R-squared (Test)',  
 'Adjusted R-squared',  
 'RMSE (Training)',  
 'RMSE (Test)',  
 'RSS (Residual Sum of Squares)',  
 'MSE (Mean Squared Error)',  
 'AIC',  
 'BIC',  
 'Log-Likelihood'  
 ],  
 'Multiple linear regression': [  
 f"{lr.intercept\_:.4f}",  
 f"{k}",  
 f"{n:,}",  
 f"{r2\_train\_lr:.4f}",  
 f"{r2\_test\_lr:.4f}",  
 f"{adj\_r2\_lr:.4f}",  
 f"${rmse\_train\_lr:,.2f}",  
 f"${rmse\_test\_lr:,.2f}",  
 f"{rss:,.2f}",  
 f"{mse\_lr:,.2f}",  
 f"{aic\_lr:.2f}",  
 f"{bic\_lr:.2f}",  
 f"{log\_likelihood\_lr:.2f}"  
 ],  
 'Random Forest': [  
 'N/A',  
 f"{k}",  
 f"{n:,}",  
 f"{r2\_train\_rf:.4f}",  
 f"{r2\_test\_rf:.4f}",  
 'N/A',  
 f"${rmse\_train\_rf:,.2f}",  
 f"${rmse\_test\_rf:,.2f}",  
 f"{rss\_rf:,.2f}",  
 f"{rmse\_train\_rf\*\*2:,.2f}",  
 'N/A\*',  
 'N/A\*',  
 'N/A\*'  
 ]  
})  
  
print(regression\_stats.to\_string(index=False))  
print("\n\* AIC/BIC/Log-Likelihood are only applicable to parametric linear models")  
print("\nNote: R-squared (Test) shows model performance on unseen data")  
  
print("\n=== FEATURE COEFFICIENTS / IMPORTANCE COMPARISON ===\n")  
  
# Sanity check for alignment  
assert len(regression\_features) == len(lr.coef\_) == len(rf\_reg.feature\_importances\_), \  
 "Mismatch between features, LR coefficients, and RF importances!"  
  
# Combine features, coefficients, and importances  
coef\_comparison = pd.DataFrame({  
 'Feature': regression\_features,  
 'LR\_Coefficient': lr.coef\_,  
 'RF\_Importance': rf\_reg.feature\_importances\_  
})  
  
# Filter out zero coefficients  
coef\_comparison = coef\_comparison[coef\_comparison['LR\_Coefficient'] != 0.0]  
  
# Adding impact type (positive or negative only)  
coef\_comparison['Impact'] = coef\_comparison['LR\_Coefficient'].apply(  
 lambda x: 'Positive' if x > 0 else 'Negative'  
)  
  
coef\_comparison['LR\_Coefficient'] = coef\_comparison['LR\_Coefficient'].round(4)  
coef\_comparison['RF\_Importance'] = coef\_comparison['RF\_Importance'].round(4)  
  
# Positive coefficients (highest impact)  
top\_positive = (  
 coef\_comparison[coef\_comparison['Impact'] == 'Positive']  
 .sort\_values(by='LR\_Coefficient', ascending=False)  
 .head(15)  
)  
  
print("Top 15 Features by Multiple linear regression Coefficient (Positive Impact):")  
print(top\_positive[['Feature', 'LR\_Coefficient', 'RF\_Importance']].to\_string(index=False))  
  
# Negative coefficients (lowest impact)  
top\_negative = (  
 coef\_comparison[coef\_comparison['Impact'] == 'Negative']  
 .sort\_values(by='LR\_Coefficient', ascending=True)  
 .head(15)  
)  
  
print("\nTop 15 Features by Multiple linear regression Coefficient (Negative Impact):")  
print(top\_negative[['Feature', 'LR\_Coefficient', 'RF\_Importance']].to\_string(index=False))  
  
print(f"\nMultiple linear regression Intercept: {lr.intercept\_:.4f}")  
  
print("\nNote:")  
print("- LR Coefficients show the direction and strength of linear relationships with the target.")  
print("- Positive coefficients increase predicted salary; negative coefficients decrease it.")  
print("- RF Importance reflects how much each feature contributes to model accuracy (non-linear).")

Training set size: 24,646  
Test set size: 6,162  
Skills based regression models training completed  
  
=== REGRESSION MODEL STATISTICS ===  
  
 Statistic Multiple linear regression Random Forest  
 Intercept 117744.2020 N/A  
 Number of Features 27 27  
Number of Observations (Train) 24,646 24,646  
 R-squared (Training) 0.2678 0.5237  
 R-squared (Test) 0.2780 0.4672  
 Adjusted R-squared 0.2670 N/A  
 RMSE (Training) $38,730.71 $31,238.34  
 RMSE (Test) $37,899.01 $32,558.54  
 RSS (Residual Sum of Squares) 36,970,667,997,179.88 24,050,395,305,765.03  
 MSE (Mean Squared Error) 1,500,067,678.21 975,833,616.24  
 AIC 520793.81 N/A\*  
 BIC 521012.85 N/A\*  
 Log-Likelihood -260369.91 N/A\*  
  
\* AIC/BIC/Log-Likelihood are only applicable to parametric linear models  
  
Note: R-squared (Test) shows model performance on unseen data  
  
=== FEATURE COEFFICIENTS / IMPORTANCE COMPARISON ===  
  
Top 15 Features by Multiple linear regression Coefficient (Positive Impact):  
 Feature LR\_Coefficient RF\_Importance  
 experience\_years 17850.7023 0.4932  
has\_python\_programming\_language 5665.9793 0.0300  
 has\_amazon\_web\_services 3354.0141 0.0361  
 has\_big\_data 3255.5115 0.0257  
 has\_artificial\_intelligence 1905.1339 0.0239  
 has\_microsoft\_azure 1587.5365 0.0192  
 has\_machine\_learning 1501.0951 0.0265  
 has\_data\_science 1297.6624 0.0261  
 has\_pandas\_python\_package 934.5885 0.0042  
has\_scikit\_learn\_python\_package 733.9453 0.0003  
 has\_deep\_learning 404.1220 0.0023  
 has\_google\_cloud\_platform\_gcp 213.1297 0.0084  
 is\_remote 201.2772 0.0728  
 has\_computer\_vision 130.9562 0.0006  
 has\_apache\_hadoop 70.5746 0.0075  
  
Top 15 Features by Multiple linear regression Coefficient (Negative Impact):  
 Feature LR\_Coefficient RF\_Importance  
 has\_data\_analysis -7222.0464 0.0426  
 has\_r\_programming\_language -2811.3382 0.0193  
 has\_power\_bi -1737.9884 0.0232  
 has\_statistics -1298.4926 0.0302  
 has\_sql\_programming\_language -1087.0998 0.0350  
 has\_pytorch\_machine\_learning\_library -801.3560 0.0005  
has\_tableau\_business\_intelligence\_software -789.6728 0.0372  
 has\_numpy\_python\_package -635.2314 0.0004  
 has\_data\_visualization -524.2779 0.0249  
 has\_natural\_language\_processing\_nlp -260.2604 0.0019  
 has\_tensorflow -256.5688 0.0005  
 has\_apache\_spark -248.2513 0.0073  
  
Multiple linear regression Intercept: 117744.2020  
  
Note:  
- LR Coefficients show the direction and strength of linear relationships with the target.  
- Positive coefficients increase predicted salary; negative coefficients decrease it.  
- RF Importance reflects how much each feature contributes to model accuracy (non-linear).

Both models are trained on 80% of the data and will be evaluated on the remaining 20% test set. The Random Forest model can capture non-linear relationships and interactions between skills, while Multiple Linear Regression provides a baseline for comparison.

# Test metrics already calculated above  
r2\_lr = r2\_test\_lr  
r2\_rf = r2\_test\_rf  
rmse\_lr = rmse\_test\_lr  
rmse\_rf = rmse\_test\_rf  
y\_pred\_rf = y\_test\_pred\_rf  
  
print("Skills-based Regression Model Performance (Test Set):")  
print(f"Multiple Linear Regression - RMSE: ${rmse\_lr:,.2f}, R²: {r2\_lr:.4f}")  
print(f"Random Forest - RMSE: ${rmse\_rf:,.2f}, R²: {r2\_rf:.4f}")  
  
# Feature importance for Random Forest  
#Features that actually exist in the model  
actual\_feature\_names = [col for col in regression\_features if col in X\_train.columns]  
importances = rf\_reg.feature\_importances\_  
  
# Visualize feature importance  
fig = px.bar(x=actual\_feature\_names, y=importances,  
 title="Skills Impact on Salary (Random Forest Feature Importance)",  
 labels={'x': 'Features', 'y': 'Importance'})  
fig.update\_layout(template="plotly\_white", xaxis\_tickangle=-45)  
fig.show()  
  
# Top skills by salary impact  
skill\_importance = list(zip(actual\_feature\_names, importances))  
skill\_importance.sort(key=lambda x: x[1], reverse=True)  
print("\nTop skills by salary impact:")  
for skill, importance in skill\_importance[:10]:  
 print(f"{skill}: {importance:.4f}")

Skills-based Regression Model Performance (Test Set):  
Multiple Linear Regression - RMSE: $37,899.01, R²: 0.2780  
Random Forest - RMSE: $32,558.54, R²: 0.4672

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

Top skills by salary impact:  
experience\_years: 0.4932  
is\_remote: 0.0728  
has\_data\_analysis: 0.0426  
has\_tableau\_business\_intelligence\_software: 0.0372  
has\_amazon\_web\_services: 0.0361  
has\_sql\_programming\_language: 0.0350  
has\_statistics: 0.0302  
has\_python\_programming\_language: 0.0300  
has\_machine\_learning: 0.0265  
has\_data\_science: 0.0261

### Regression Analysis: What drives salary?

Prediction models were built to understand how skills influence salary. The Random Forest model achieved R2 of 0.47 compared to 0.28 for Multiple Linear Regression, showing that skill-salary relationships are complex.

**Model Performance:**

* **Random Forest:** R² = 0.47 (explains 47% of salary variation), RMSE = $32,559
* **Multiple Linear Regression:** R² = 0.28
* **Insight:** Skills alone do not fully explain salary — other factors also matter.

**Key Salary Drivers (Feature Importance):**

1. **Experience (0.49):** Largest factor, nearly half of salary variation
2. **Remote work (0.07):** Flexibility influences pay differences
3. **Data Analysis (0.04):** Core analytical capability
4. **Tableau (0.04):** Visualization and BI tool
5. **AWS (0.04):** Cloud computing platform
6. **SQL (0.04):** Database querying and manipulation
7. **Statistics (0.03):** Analytical foundation
8. **Python (0.03):** Programming language

**Career Implications:**

* **Experience is critical** — the strongest driver of salary.
* **Remote work adds value** — flexibility can boost compensation.
* **Skill combinations matter** — technical, analytical, and cloud skills together shape salary outcomes.

**Summary:** Salary is not determined by skills alone. Experience and work flexibility are key, while technical skills provide additional differentiation.

## Classification to Identify BA/ML/DS Roles

Although the project required only one of the supervised learning models. This analysis also explores the classification to distinguish ML/Data Science roles from Business Analytics and other positions. A Random Forest Classifier is trained to predict whether a job is an ML/DS role based on its skill requirements. This analysis reveals which skills are the strongest “signature” indicators that distinguish ML/DS positions from BA roles.

# Prepare features for classification.  
classification\_features = skill\_feature\_cols + ['experience\_years', 'is\_remote']  
  
# Classification data  
X\_clf = df\_modeling[classification\_features].fillna(0)  
# Target: ML/DS roles (computed from is\_ml\_role OR is\_ds\_role)  
y\_clf = ((df\_modeling['is\_ml\_role'] == 1) | (df\_modeling['is\_ds\_role'] == 1)).astype(int)  
  
# Train/test split for classification  
X\_train\_clf, X\_test\_clf, y\_train\_clf, y\_test\_clf = train\_test\_split(X\_clf, y\_clf, test\_size=0.2, random\_state=42)  
  
# Scale features  
scaler\_clf = StandardScaler()  
X\_train\_clf\_scaled = scaler\_clf.fit\_transform(X\_train\_clf)  
X\_test\_clf\_scaled = scaler\_clf.transform(X\_test\_clf)  
  
# Random Forest Classification  
rf\_clf = RandomForestClassifier(n\_estimators=100, random\_state=42)  
rf\_clf.fit(X\_train\_clf\_scaled, y\_train\_clf)  
  
print("Skills-based classification model trained successfully!")

Skills-based classification model trained successfully!

The classifier learns patterns that distinguish ML/DS roles from BA and other positions based on their skill profiles. The model is now evaluated to see how accurately it can identify these specialized ML/DS roles versus the more common BA positions.

# Random Forest predictions  
y\_pred\_rf\_clf = rf\_clf.predict(X\_test\_clf\_scaled)  
accuracy\_rf = accuracy\_score(y\_test\_clf, y\_pred\_rf\_clf)  
f1\_rf = f1\_score(y\_test\_clf, y\_pred\_rf\_clf)  
  
print("Skills based Classification Model Performance:")  
print(f"Random Forest - Accuracy: {accuracy\_rf:.4f}, F1 Score: {f1\_rf:.4f}")  
  
# Confusion Matrix for Random Forest  
cm = confusion\_matrix(y\_test\_clf, y\_pred\_rf\_clf)  
  
# Visualize confusion matrix  
fig = px.imshow(cm, text\_auto=True, aspect="auto",  
 title="Confusion Matrix - ML/DS Role Classification",  
 labels=dict(x="Predicted", y="Actual"),  
 color\_continuous\_scale="Blues")  
  
fig.update\_layout(template="plotly\_white")  
fig.update\_xaxes(tickvals=[0,1], ticktext=['Not ML/DS', 'ML/DS'])  
fig.update\_yaxes(tickvals=[0,1], ticktext=['Not ML/DS', 'ML/DS'])  
fig.show()  
  
print("Classification Report:")  
print(classification\_report(y\_test\_clf, y\_pred\_rf\_clf))  
  
# Features that actually exist in the classification model  
clf\_actual\_feature\_names = [col for col in classification\_features if col in X\_train\_clf.columns]  
clf\_importances = rf\_clf.feature\_importances\_  
  
# Visualize classification feature importance  
fig = px.bar(x=clf\_actual\_feature\_names, y=clf\_importances,  
 title="Skills Impact on ML/Data Science Role Classification",  
 labels={'x': 'Features', 'y': 'Importance'})  
fig.update\_layout(template="plotly\_white", xaxis\_tickangle=-45)  
fig.show()

Skills based Classification Model Performance:  
Random Forest - Accuracy: 0.9995, F1 Score: 0.9986

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

Classification Report:  
 precision recall f1-score support  
  
 0 1.00 1.00 1.00 5082  
 1 1.00 1.00 1.00 1080  
  
 accuracy 1.00 6162  
 macro avg 1.00 1.00 1.00 6162  
weighted avg 1.00 1.00 1.00 6162

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

### Classification Results: Identifying ML/Data Science Roles

A Random Forest classifier was used to predict whether a job is an ML/Data Science role based on its skill requirements. The model achieved very strong performance in separating ML/DS roles from Business Analytics and other positions.

**Model Performance:**

* **Accuracy:** 99.95% — nearly all ML/DS roles correctly identified
* **Insight:** ML/DS roles have distinct skill patterns compared to BA and general analyst jobs
* **Conclusion:** Skill-based criteria effectively distinguish ML/DS roles from BA positions

**Key Predictive Skills (Feature Importance)**

* **Programming:** Python, R
* **ML Frameworks:** TensorFlow, PyTorch
* **Statistical Modeling:** Core differentiator for ML/DS
* **BA-Oriented Skills:** SQL, Tableau, Power BI, Data Analysis (more common in BA roles)

**Career Implications**

* **Distinct skill sets:** ML/DS roles require clearly different capabilities than BA roles
* **ML/DS focus:** Programming, modeling, and ML frameworks are the strongest signals
* **BA focus:** SQL, visualization, and reporting tools dominate BA roles
* **Career development:** Building expertise in high-importance ML/DS features directly improves readiness for ML/DS positions

**Summary:**The Random Forest classifier confirms that ML/DS roles are defined by specialized technical skills, while BA roles emphasize analysis and visualization tools. This distinction provides a clear roadmap for professionals aiming to transition into ML/DS careers.

# Model Results Visualization

This section provides a consolidated view of the regression modeling approaches. The comparison shows how different models perform on salary prediction and highlights the most impactful skills across different analyses.

# Model performance  
model\_summary = pd.DataFrame({  
 'Model': ['Multiple Linear Regression', 'Random Forest (Regression)'],  
 'R² (Test)': [r2\_lr, r2\_rf],  
 'RMSE (Test)': [rmse\_lr, rmse\_rf]  
})  
print(model\_summary)  
  
# Visualization of model results  
fig = make\_subplots(  
 rows=2, cols=2,  
 subplot\_titles=('R² Comparison (Test Set)', 'RMSE Comparison (Test Set)',  
 'Skills vs Salary Impact', 'Predicted vs Actual Salary'),  
 specs=[[{"type": "bar"}, {"type": "bar"}],  
 [{"type": "bar"}, {"type": "scatter"}]]  
)  
  
# Row 1, Col 1: R² comparison  
models = ['Multiple linear regression', 'Random Forest']  
r2\_values = [r2\_lr, r2\_rf]  
fig.add\_trace(go.Bar(x=models, y=r2\_values, name="R² Score",  
 marker\_color=['steelblue', 'darkgreen']), row=1, col=1)  
  
# Row 1, Col 2: RMSE comparison  
rmse\_values = [rmse\_lr, rmse\_rf]  
fig.add\_trace(go.Bar(x=models, y=rmse\_values, name="RMSE",  
 marker\_color=['coral', 'orange']), row=1, col=2)  
  
# Row 2, Col 1: Skills vs salary impact (top 10)  
top\_skills\_salary = skill\_importance[:10]  
fig.add\_trace(go.Bar(  
 x=[s[1] for s in top\_skills\_salary],  
 y=[s[0] for s in top\_skills\_salary],  
 orientation='h',  
 name="Feature Importance",  
 marker\_color='purple'), row=2, col=1)  
  
# Row 2, Col 2: Predicted vs Actual for Random Forest  
sample\_size = min(500, len(y\_test))  
sample\_indices = np.random.choice(len(y\_test), sample\_size, replace=False)  
fig.add\_trace(go.Scatter(  
 x=y\_test.iloc[sample\_indices],  
 y=y\_pred\_rf[sample\_indices],  
 mode='markers',  
 name='RF Predictions',  
 marker=dict(color='darkgreen', size=5, opacity=0.6)), row=2, col=2)  
  
# Prediction line  
min\_val = min(y\_test.min(), y\_pred\_rf.min())  
max\_val = max(y\_test.max(), y\_pred\_rf.max())  
fig.add\_trace(go.Scatter(  
 x=[min\_val, max\_val],  
 y=[min\_val, max\_val],  
 mode='lines',  
 name='Perfect Prediction',  
 line=dict(color='red', dash='dash')), row=2, col=2)  
  
# Axis labels  
fig.update\_xaxes(title\_text="Model", row=1, col=1)  
fig.update\_yaxes(title\_text="R² Score", row=1, col=1)  
fig.update\_xaxes(title\_text="Model", row=1, col=2)  
fig.update\_yaxes(title\_text="RMSE ($)", row=1, col=2)  
fig.update\_xaxes(title\_text="Importance", row=2, col=1)  
fig.update\_yaxes(title\_text="Feature", row=2, col=1)  
fig.update\_xaxes(title\_text="Actual Salary ($)", row=2, col=2)  
fig.update\_yaxes(title\_text="Predicted Salary ($)", row=2, col=2)  
  
fig.update\_layout(  
 height=800,  
 showlegend=False,  
 template="plotly\_white",  
 title={  
 'text': "Regression Model Comparison - BA/ML/DS Salary Prediction",  
 'y': 0.98,  
 'x': 0.5,  
 'xanchor': 'center',  
 'yanchor': 'top',  
 },  
 margin=dict(t=80)  
)  
fig.show()

Model R² (Test) RMSE (Test)  
0 Multiple Linear Regression 0.278032 37899.005358  
1 Random Forest (Regression) 0.467166 32558.537199

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

# Key Takeaways and Recommendations

## Summary of Findings

Our analysis of business analytics, data science and machine learning job postings reveals several important patterns:

* **Role Distribution:** Business Analytics dominates (35% of jobs), while ML and DS remain smaller but specialized segments.
* **Job Segmentation:** Six distinct clusters reveal clear differences in pay, experience, and hybrid skill mixes.
* **Salary Drivers:** Experience is the strongest factor (49%), with remote work and technical skills adding incremental impact.
* **Role Differentiation:** ML/DS roles are highly distinct, with classification accuracy of 99.95% separating them from BA roles.

## Recommendations for Job Seekers

**For Career Advancement:**

* **Gain experience** - it’s the single biggest salary driver (49% importance)
* **Remote work flexibility** - BA/ML/DS roles pay well even when remote, showing that onsite presence is not necessary for competitive salaries.
* **Learn practical tools**: Data analysis (4.3%), Tableau (3.7%), AWS (3.6%), SQL (3.5%), Statistics (3.0%), Python (3.0%)
* General technical roles (Cluster 1) pay highest ($145K) with most opportunities (10,189 jobs)

**For Business Analytics Path:**

* **Highest volume opportunity**: 10,831 BA roles identified (35% of job market)
* **Core BA skills**: SQL, Tableau/Power BI, data visualization, data analysis
* **Best BA cluster**: Cluster 3 (6,573 jobs at $109K) with 95% BA roles
* **Hybrid advantage**: Many BA roles overlap with DS (39% in Cluster 3), so learning Python/statistics opens DS opportunities

**For Transitioning to ML/Data Science:**

* **ML path** (3,226 roles): Most specialized and competitive - requires TensorFlow, PyTorch, Deep Learning, NLP
* **DS path** (2,877 roles): Requires R or Python + Statistics + multiple DS tools (Pandas, NumPy, Scikit-learn)
* **Pure ML roles** (Cluster 4): Only 77 jobs at $140K - highly specialized
* The 99.95% classification accuracy shows these roles need very specific skill combinations

**For Maximizing Opportunities:**

* **Most jobs + highest pay**: Cluster 1 (10,189 jobs at $145K) - general technical roles, only 17% need BA/ML/DS
* **Entry-level**: Cluster 2 (13,313 jobs at $101K) - 29% BA/ML/DS, lowest experience requirement (2.0 years)
* **BA opportunities**: Cluster 3 (6,573 jobs at $109K) - 99% need BA/ML/DS (95% BA, 39% DS overlap)
* **Remote work**: Cluster 5 (73 jobs at $118K, 56% remote) - 96% hybrid BA/ML/DS roles
* **High-skill hybrid**: Cluster 0 (583 jobs at $140K) - 90% BA/ML/DS (60% ML + 70% BA combination)

## Limitations and Considerations

* The analysis is based on job posting data which may not reflect actual hiring outcomes
* Skill requirements in job posts may differ from day-to-day job responsibilities
* Market conditions and geographic factors also influence salaries beyond just skills