Assignment 04

https://github.com/met-ad-688/assignment-04-leoliu36.git

2025-9-30

Feature Engineering and Missing Value Imputation

## DATA LOADING & SETUP  
import pandas as pd  
import plotly.express as px  
import plotly.io as pio  
from pyspark.sql import SparkSession  
import re  
import numpy as np  
import plotly.graph\_objects as go  
from pyspark.sql.functions import col, split, explode, regexp\_replace, transform, when, trim, monotonically\_increasing\_id, pow, length, sum as spark\_sum  
from pyspark.sql import functions as F  
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler  
from pyspark.ml import Pipeline  
from pyspark.ml.regression import LinearRegression  
  
np.random.seed(42)  
  
pio.renderers.default = "notebook"  
  
# Initialize Spark Session  
spark = SparkSession.builder.appName("LightcastData").getOrCreate()  
  
# Load Data  
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine", "true").option("quote", "\"").option("escape", "\"").csv("../data/lightcast\_job\_postings.csv")  
df.createOrReplaceTempView("job\_postings")  
  
# Show Schema and Sample Data  
#print("---This is Diagnostic check, No need to print it in the final doc---")  
#df.printSchema() # comment this line when rendering the submission  
df.show(5)

WARNING: Using incubator modules: jdk.incubator.vector  
Using Spark's default log4j profile: org/apache/spark/log4j2-defaults.properties  
Setting default log level to "WARN".  
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).  
25/10/09 01:18:13 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
25/10/09 01:18:31 WARN SparkStringUtils: Truncated the string representation of a plan since it was too large. This behavior can be adjusted by setting 'spark.sql.debug.maxToStringFields'.  
   
  
+--------------------+-----------------+----------------------+----------+--------+---------+--------+--------------------+--------------------+--------------------+-----------+-------------------+--------------------+--------------------+---------------+----------------+--------+--------------------+-----------+-------------------+----------------+---------------------+-------------+-------------------+-------------+------------------+---------------+--------------------+--------------------+--------------------+-------------+------+-----------+----------------+-------------------+---------+-----------+--------------------+--------------------+-------------+------+--------------+-----+--------------------+-----+----------+---------------+--------------------+---------------+--------------------+------------+--------------------+------------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------------------+-------------------+--------------------+--------------------+--------------------+--------------------+-----------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+----------+---------------+----------+---------------+---------------+--------------------+--------------+--------------------+--------------------------+-------------------------------+--------------------+-------------------------+-----------------------------+----------------------------------+-----------------+----------------------+-----------------------+----------------------------+------------------+-----------------------+-------+--------------------+-------+--------------------+-------+---------------+-------+---------------+-----------------+----------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+  
| ID|LAST\_UPDATED\_DATE|LAST\_UPDATED\_TIMESTAMP|DUPLICATES| POSTED| EXPIRED|DURATION| SOURCE\_TYPES| SOURCES| URL|ACTIVE\_URLS|ACTIVE\_SOURCES\_INFO| TITLE\_RAW| BODY|MODELED\_EXPIRED|MODELED\_DURATION| COMPANY| COMPANY\_NAME|COMPANY\_RAW|COMPANY\_IS\_STAFFING|EDUCATION\_LEVELS|EDUCATION\_LEVELS\_NAME|MIN\_EDULEVELS| MIN\_EDULEVELS\_NAME|MAX\_EDULEVELS|MAX\_EDULEVELS\_NAME|EMPLOYMENT\_TYPE|EMPLOYMENT\_TYPE\_NAME|MIN\_YEARS\_EXPERIENCE|MAX\_YEARS\_EXPERIENCE|IS\_INTERNSHIP|SALARY|REMOTE\_TYPE|REMOTE\_TYPE\_NAME|ORIGINAL\_PAY\_PERIOD|SALARY\_TO|SALARY\_FROM| LOCATION| CITY| CITY\_NAME|COUNTY| COUNTY\_NAME| MSA| MSA\_NAME|STATE|STATE\_NAME|COUNTY\_OUTGOING|COUNTY\_NAME\_OUTGOING|COUNTY\_INCOMING|COUNTY\_NAME\_INCOMING|MSA\_OUTGOING| MSA\_NAME\_OUTGOING|MSA\_INCOMING| MSA\_NAME\_INCOMING|NAICS2| NAICS2\_NAME|NAICS3| NAICS3\_NAME|NAICS4| NAICS4\_NAME|NAICS5| NAICS5\_NAME|NAICS6| NAICS6\_NAME| TITLE| TITLE\_NAME| TITLE\_CLEAN| SKILLS| SKILLS\_NAME| SPECIALIZED\_SKILLS|SPECIALIZED\_SKILLS\_NAME| CERTIFICATIONS| CERTIFICATIONS\_NAME| COMMON\_SKILLS| COMMON\_SKILLS\_NAME| SOFTWARE\_SKILLS|SOFTWARE\_SKILLS\_NAME| ONET| ONET\_NAME| ONET\_2019| ONET\_2019\_NAME| CIP6| CIP6\_NAME| CIP4| CIP4\_NAME| CIP2| CIP2\_NAME|SOC\_2021\_2| SOC\_2021\_2\_NAME|SOC\_2021\_3| SOC\_2021\_3\_NAME|SOC\_2021\_4|SOC\_2021\_4\_NAME|SOC\_2021\_5|SOC\_2021\_5\_NAME|LOT\_CAREER\_AREA|LOT\_CAREER\_AREA\_NAME|LOT\_OCCUPATION| LOT\_OCCUPATION\_NAME|LOT\_SPECIALIZED\_OCCUPATION|LOT\_SPECIALIZED\_OCCUPATION\_NAME|LOT\_OCCUPATION\_GROUP|LOT\_OCCUPATION\_GROUP\_NAME|LOT\_V6\_SPECIALIZED\_OCCUPATION|LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME|LOT\_V6\_OCCUPATION|LOT\_V6\_OCCUPATION\_NAME|LOT\_V6\_OCCUPATION\_GROUP|LOT\_V6\_OCCUPATION\_GROUP\_NAME|LOT\_V6\_CAREER\_AREA|LOT\_V6\_CAREER\_AREA\_NAME| SOC\_2| SOC\_2\_NAME| SOC\_3| SOC\_3\_NAME| SOC\_4| SOC\_4\_NAME| SOC\_5| SOC\_5\_NAME|LIGHTCAST\_SECTORS|LIGHTCAST\_SECTORS\_NAME|NAICS\_2022\_2| NAICS\_2022\_2\_NAME|NAICS\_2022\_3| NAICS\_2022\_3\_NAME|NAICS\_2022\_4| NAICS\_2022\_4\_NAME|NAICS\_2022\_5| NAICS\_2022\_5\_NAME|NAICS\_2022\_6| NAICS\_2022\_6\_NAME|  
+--------------------+-----------------+----------------------+----------+--------+---------+--------+--------------------+--------------------+--------------------+-----------+-------------------+--------------------+--------------------+---------------+----------------+--------+--------------------+-----------+-------------------+----------------+---------------------+-------------+-------------------+-------------+------------------+---------------+--------------------+--------------------+--------------------+-------------+------+-----------+----------------+-------------------+---------+-----------+--------------------+--------------------+-------------+------+--------------+-----+--------------------+-----+----------+---------------+--------------------+---------------+--------------------+------------+--------------------+------------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------------------+-------------------+--------------------+--------------------+--------------------+--------------------+-----------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+----------+---------------+----------+---------------+---------------+--------------------+--------------+--------------------+--------------------------+-------------------------------+--------------------+-------------------------+-----------------------------+----------------------------------+-----------------+----------------------+-----------------------+----------------------------+------------------+-----------------------+-------+--------------------+-------+--------------------+-------+---------------+-------+---------------+-----------------+----------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+  
|1f57d95acf4dc67ed...| 9/6/2024| 2024-09-06 20:32:...| 0|6/2/2024| 6/8/2024| 6| [\n "Company"\n]|[\n "brassring.c...|[\n "https://sjo...| []| NULL|Enterprise Analys...|31-May-2024\n\nEn...| 6/8/2024| 6| 894731| Murphy USA| Murphy USA| false| [\n 2\n]| [\n "Bachelor's ...| 2| Bachelor's degree| NULL| NULL| 1|Full-time (> 32 h...| 2| 2| false| NULL| 0| [None]| NULL| NULL| NULL|{\n "lat": 33.20...|RWwgRG9yYWRvLCBBUg==|El Dorado, AR| 5139| Union, AR|20980| El Dorado, AR| 5| Arkansas| 5139| Union, AR| 5139| Union, AR| 20980| El Dorado, AR| 20980| El Dorado, AR| 44| Retail Trade| 441|Motor Vehicle and...| 4413|Automotive Parts,...| 44133|Automotive Parts ...|441330|Automotive Parts ...|ET29C073C03D1F86B4|Enterprise Analysts|enterprise analys...|[\n "KS126DB6T06...|[\n "Merchandisi...|[\n "KS126DB6T06...| [\n "Merchandisi...| []| []|[\n "KS126706DPF...|[\n "Mathematics...|[\n "KS440W865GC...|[\n "SQL (Progra...|15-2051.01|Business Intellig...|15-2051.01|Business Intellig...|[\n "45.0601",\n...|[\n "Economics, ...|[\n "45.06",\n ...|[\n "Economics",...|[\n "45",\n "27...|[\n "Social Scie...| 15-0000|Computer and Math...| 15-2000|Mathematical Scie...| 15-2050|Data Scientists| 15-2051|Data Scientists| 23|Information Techn...| 231010|Business Intellig...| 23101011| General ERP Analy...| 2310| Business Intellig...| 23101011| General ERP Analy...| 231010| Business Intellig...| 2310| Business Intellig...| 23| Information Techn...|15-0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-2051|Data Scientists| [\n 7\n]| [\n "Artificial ...| 44| Retail Trade| 441|Motor Vehicle and...| 4413|Automotive Parts,...| 44133|Automotive Parts ...| 441330|Automotive Parts ...|  
|0cb072af26757b6c4...| 8/2/2024| 2024-08-02 17:08:...| 0|6/2/2024| 8/1/2024| NULL| [\n "Job Board"\n]| [\n "maine.gov"\n]|[\n "https://job...| []| NULL|Oracle Consultant...|Oracle Consultant...| 8/1/2024| NULL| 133098|Smx Corporation L...| SMX| true| [\n 99\n]| [\n "No Educatio...| 99|No Education Listed| NULL| NULL| 1|Full-time (> 32 h...| 3| 3| false| NULL| 1| Remote| NULL| NULL| NULL|{\n "lat": 44.31...| QXVndXN0YSwgTUU=| Augusta, ME| 23011| Kennebec, ME|12300|Augusta-Watervill...| 23| Maine| 23011| Kennebec, ME| 23011| Kennebec, ME| 12300|Augusta-Watervill...| 12300|Augusta-Watervill...| 56|Administrative an...| 561|Administrative an...| 5613| Employment Services| 56132|Temporary Help Se...|561320|Temporary Help Se...|ET21DDA63780A7DC09| Oracle Consultants|oracle consultant...|[\n "KS122626T55...|[\n "Procurement...|[\n "KS122626T55...| [\n "Procurement...| []| []| []| []|[\n "BGSBF3F508F...|[\n "Oracle Busi...|15-2051.01|Business Intellig...|15-2051.01|Business Intellig...| []| []| []| []| []| []| 15-0000|Computer and Math...| 15-2000|Mathematical Scie...| 15-2050|Data Scientists| 15-2051|Data Scientists| 23|Information Techn...| 231010|Business Intellig...| 23101012| Oracle Consultant...| 2310| Business Intellig...| 23101012| Oracle Consultant...| 231010| Business Intellig...| 2310| Business Intellig...| 23| Information Techn...|15-0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-2051|Data Scientists| NULL| NULL| 56|Administrative an...| 561|Administrative an...| 5613| Employment Services| 56132|Temporary Help Se...| 561320|Temporary Help Se...|  
|85318b12b3331fa49...| 9/6/2024| 2024-09-06 20:32:...| 1|6/2/2024| 7/7/2024| 35| [\n "Job Board"\n]|[\n "dejobs.org"\n]|[\n "https://dej...| []| NULL| Data Analyst|Taking care of pe...| 6/10/2024| 8|39063746| Sedgwick| Sedgwick| false| [\n 2\n]| [\n "Bachelor's ...| 2| Bachelor's degree| NULL| NULL| 1|Full-time (> 32 h...| 5| NULL| false| NULL| 0| [None]| NULL| NULL| NULL|{\n "lat": 32.77...| RGFsbGFzLCBUWA==| Dallas, TX| 48113| Dallas, TX|19100|Dallas-Fort Worth...| 48| Texas| 48113| Dallas, TX| 48113| Dallas, TX| 19100|Dallas-Fort Worth...| 19100|Dallas-Fort Worth...| 52|Finance and Insur...| 524|Insurance Carrier...| 5242|Agencies, Brokera...| 52429|Other Insurance R...|524291| Claims Adjusting|ET3037E0C947A02404| Data Analysts| data analyst|[\n "KS1218W78FG...|[\n "Management"...|[\n "ESF3939CE1F...| [\n "Exception R...|[\n "KS683TN76T7...|[\n "Security Cl...|[\n "KS1218W78FG...|[\n "Management"...|[\n "KS126HY6YLT...|[\n "Microsoft O...|15-2051.01|Business Intellig...|15-2051.01|Business Intellig...| []| []| []| []| []| []| 15-0000|Computer and Math...| 15-2000|Mathematical Scie...| 15-2050|Data Scientists| 15-2051|Data Scientists| 23|Information Techn...| 231113|Data / Data Minin...| 23111310| Data Analyst| 2311| Data Analysis and...| 23111310| Data Analyst| 231113| Data / Data Minin...| 2311| Data Analysis and...| 23| Information Techn...|15-0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-2051|Data Scientists| NULL| NULL| 52|Finance and Insur...| 524|Insurance Carrier...| 5242|Agencies, Brokera...| 52429|Other Insurance R...| 524291| Claims Adjusting|  
|1b5c3941e54a1889e...| 9/6/2024| 2024-09-06 20:32:...| 1|6/2/2024|7/20/2024| 48| [\n "Job Board"\n]|[\n "disabledper...|[\n "https://www...| []| NULL|Sr. Lead Data Mgm...|About this role:\...| 6/12/2024| 10|37615159| Wells Fargo|Wells Fargo| false| [\n 99\n]| [\n "No Educatio...| 99|No Education Listed| NULL| NULL| 1|Full-time (> 32 h...| 3| NULL| false| NULL| 0| [None]| NULL| NULL| NULL|{\n "lat": 33.44...| UGhvZW5peCwgQVo=| Phoenix, AZ| 4013| Maricopa, AZ|38060|Phoenix-Mesa-Chan...| 4| Arizona| 4013| Maricopa, AZ| 4013| Maricopa, AZ| 38060|Phoenix-Mesa-Chan...| 38060|Phoenix-Mesa-Chan...| 52|Finance and Insur...| 522|Credit Intermedia...| 5221|Depository Credit...| 52211| Commercial Banking|522110| Commercial Banking|ET2114E0404BA30075|Management Analysts|sr lead data mgmt...|[\n "KS123QX62QY...|[\n "Exit Strate...|[\n "KS123QX62QY...| [\n "Exit Strate...| []| []|[\n "KS7G6NP6R6L...|[\n "Reliability...|[\n "KS4409D76NW...|[\n "SAS (Softwa...|15-2051.01|Business Intellig...|15-2051.01|Business Intellig...| []| []| []| []| []| []| 15-0000|Computer and Math...| 15-2000|Mathematical Scie...| 15-2050|Data Scientists| 15-2051|Data Scientists| 23|Information Techn...| 231113|Data / Data Minin...| 23111310| Data Analyst| 2311| Data Analysis and...| 23111310| Data Analyst| 231113| Data / Data Minin...| 2311| Data Analysis and...| 23| Information Techn...|15-0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-2051|Data Scientists| [\n 6\n]| [\n "Data Privac...| 52|Finance and Insur...| 522|Credit Intermedia...| 5221|Depository Credit...| 52211| Commercial Banking| 522110| Commercial Banking|  
|cb5ca25f02bdf25c1...| 6/19/2024| 2024-06-19 07:00:00| 0|6/2/2024|6/17/2024| 15|[\n "FreeJobBoar...|[\n "craigslist....|[\n "https://mod...| []| NULL|Comisiones de $10...|Comisiones de $10...| 6/17/2024| 15| 0| Unclassified| LH/GM| false| [\n 99\n]| [\n "No Educatio...| 99|No Education Listed| NULL| NULL| 3|Part-time / full-...| NULL| NULL| false| 92500| 0| [None]| year| 150000| 35000|{\n "lat": 37.63...| TW9kZXN0bywgQ0E=| Modesto, CA| 6099|Stanislaus, CA|33700| Modesto, CA| 6|California| 6099| Stanislaus, CA| 6099| Stanislaus, CA| 33700| Modesto, CA| 33700| Modesto, CA| 99|Unclassified Indu...| 999|Unclassified Indu...| 9999|Unclassified Indu...| 99999|Unclassified Indu...|999999|Unclassified Indu...|ET0000000000000000| Unclassified|comisiones de por...| []| []| []| []| []| []| []| []| []| []|15-2051.01|Business Intellig...|15-2051.01|Business Intellig...| []| []| []| []| []| []| 15-0000|Computer and Math...| 15-2000|Mathematical Scie...| 15-2050|Data Scientists| 15-2051|Data Scientists| 23|Information Techn...| 231010|Business Intellig...| 23101012| Oracle Consultant...| 2310| Business Intellig...| 23101012| Oracle Consultant...| 231010| Business Intellig...| 2310| Business Intellig...| 23| Information Techn...|15-0000|Computer and Math...|15-2000|Mathematical Scie...|15-2050|Data Scientists|15-2051|Data Scientists| NULL| NULL| 99|Unclassified Indu...| 999|Unclassified Indu...| 9999|Unclassified Indu...| 99999|Unclassified Indu...| 999999|Unclassified Indu...|  
+--------------------+-----------------+----------------------+----------+--------+---------+--------+--------------------+--------------------+--------------------+-----------+-------------------+--------------------+--------------------+---------------+----------------+--------+--------------------+-----------+-------------------+----------------+---------------------+-------------+-------------------+-------------+------------------+---------------+--------------------+--------------------+--------------------+-------------+------+-----------+----------------+-------------------+---------+-----------+--------------------+--------------------+-------------+------+--------------+-----+--------------------+-----+----------+---------------+--------------------+---------------+--------------------+------------+--------------------+------------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------------------+-------------------+--------------------+--------------------+--------------------+--------------------+-----------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+----------+---------------+----------+---------------+---------------+--------------------+--------------+--------------------+--------------------------+-------------------------------+--------------------+-------------------------+-----------------------------+----------------------------------+-----------------+----------------------+-----------------------+----------------------------+------------------+-----------------------+-------+--------------------+-------+--------------------+-------+---------------+-------+---------------+-----------------+----------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+  
only showing top 5 rows

# Data Loading & Cleaning

The dataset contains columns with irrelevant data and missing values, which can negatively impact the performance of our predictive models. To address this, columns containing unnecessary information for our analysis will be removed. We then define the dependent, independent, and categorical variables; columns with over 50% missing values will be dropped. For the remaining columns, only rows with non-null values will be kept for modeling.

## DATA CLEANING  
# Drop columns that are not needed for this analysis   
columns\_to\_drop = [  
 # tracking & other metadata  
 "ID", "LAST\_UPDATED\_DATE", "LAST\_UPDATED\_TIMESTAMP", "DUPLICATES",  
 "SOURCE\_TYPES", "SOURCES", "URL", "ACTIVE\_URLS", "ACTIVE\_SOURCES\_INFO", "MODELED\_EXPIRED", "MODELED\_DURATION", "TITLE\_RAW", "ORIGINAL\_PAY\_PERIOD"  
 # outdated NAICS and SOC codes  
 "NAICS2", "NAICS2\_NAME", "NAICS3", "NAICS3\_NAME",  
 "NAICS4", "NAICS4\_NAME", "NAICS5", "NAICS5\_NAME",  
 "NAICS6", "NAICS6\_NAME",   
 "SOC\_2", "SOC\_2\_NAME", "SOC\_3", "SOC\_3\_NAME",  
 "SOC\_4", "SOC\_4\_NAME", "SOC\_5", "SOC\_5\_NAME",  
 "SOC\_2021\_2", "SOC\_2021\_2\_NAME", "SOC\_2021\_3", "SOC\_2021\_3\_NAME",  
 "SOC\_2021\_5", "SOC\_2021\_5\_NAME",  
 "NAICS\_2022\_2", "NAICS\_2022\_2\_NAME", "NAICS\_2022\_3", "NAICS\_2022\_3\_NAME",  
 "NAICS\_2022\_4", "NAICS\_2022\_4\_NAME", "NAICS\_2022\_5", "NAICS\_2022\_5\_NAME"  
 # Location encodings  
 "COUNTY\_OUTGOING", "COUNTY\_NAME\_OUTGOING",  
 "COUNTY\_INCOMING", "COUNTY\_NAME\_INCOMING",  
 "MSA\_OUTGOING", "MSA\_NAME\_OUTGOING",  
 "MSA\_INCOMING", "MSA\_NAME\_INCOMING"  
]  
  
# Drop columns   
df = df.drop(\*columns\_to\_drop)  
  
# Show resulting schema  
df.printSchema()

root  
 |-- POSTED: string (nullable = true)  
 |-- EXPIRED: string (nullable = true)  
 |-- DURATION: integer (nullable = true)  
 |-- BODY: string (nullable = true)  
 |-- COMPANY: integer (nullable = true)  
 |-- COMPANY\_NAME: string (nullable = true)  
 |-- COMPANY\_RAW: string (nullable = true)  
 |-- COMPANY\_IS\_STAFFING: boolean (nullable = true)  
 |-- EDUCATION\_LEVELS: string (nullable = true)  
 |-- EDUCATION\_LEVELS\_NAME: string (nullable = true)  
 |-- MIN\_EDULEVELS: integer (nullable = true)  
 |-- MIN\_EDULEVELS\_NAME: string (nullable = true)  
 |-- MAX\_EDULEVELS: integer (nullable = true)  
 |-- MAX\_EDULEVELS\_NAME: string (nullable = true)  
 |-- EMPLOYMENT\_TYPE: integer (nullable = true)  
 |-- EMPLOYMENT\_TYPE\_NAME: string (nullable = true)  
 |-- MIN\_YEARS\_EXPERIENCE: integer (nullable = true)  
 |-- MAX\_YEARS\_EXPERIENCE: integer (nullable = true)  
 |-- IS\_INTERNSHIP: boolean (nullable = true)  
 |-- SALARY: integer (nullable = true)  
 |-- REMOTE\_TYPE: integer (nullable = true)  
 |-- REMOTE\_TYPE\_NAME: string (nullable = true)  
 |-- ORIGINAL\_PAY\_PERIOD: string (nullable = true)  
 |-- SALARY\_TO: integer (nullable = true)  
 |-- SALARY\_FROM: integer (nullable = true)  
 |-- LOCATION: string (nullable = true)  
 |-- CITY: string (nullable = true)  
 |-- CITY\_NAME: string (nullable = true)  
 |-- COUNTY: integer (nullable = true)  
 |-- COUNTY\_NAME: string (nullable = true)  
 |-- MSA: integer (nullable = true)  
 |-- MSA\_NAME: string (nullable = true)  
 |-- STATE: integer (nullable = true)  
 |-- STATE\_NAME: string (nullable = true)  
 |-- COUNTY\_OUTGOING: integer (nullable = true)  
 |-- NAICS2: integer (nullable = true)  
 |-- TITLE: string (nullable = true)  
 |-- TITLE\_NAME: string (nullable = true)  
 |-- TITLE\_CLEAN: string (nullable = true)  
 |-- SKILLS: string (nullable = true)  
 |-- SKILLS\_NAME: string (nullable = true)  
 |-- SPECIALIZED\_SKILLS: string (nullable = true)  
 |-- SPECIALIZED\_SKILLS\_NAME: string (nullable = true)  
 |-- CERTIFICATIONS: string (nullable = true)  
 |-- CERTIFICATIONS\_NAME: string (nullable = true)  
 |-- COMMON\_SKILLS: string (nullable = true)  
 |-- COMMON\_SKILLS\_NAME: string (nullable = true)  
 |-- SOFTWARE\_SKILLS: string (nullable = true)  
 |-- SOFTWARE\_SKILLS\_NAME: string (nullable = true)  
 |-- ONET: string (nullable = true)  
 |-- ONET\_NAME: string (nullable = true)  
 |-- ONET\_2019: string (nullable = true)  
 |-- ONET\_2019\_NAME: string (nullable = true)  
 |-- CIP6: string (nullable = true)  
 |-- CIP6\_NAME: string (nullable = true)  
 |-- CIP4: string (nullable = true)  
 |-- CIP4\_NAME: string (nullable = true)  
 |-- CIP2: string (nullable = true)  
 |-- CIP2\_NAME: string (nullable = true)  
 |-- SOC\_2021\_4: string (nullable = true)  
 |-- SOC\_2021\_4\_NAME: string (nullable = true)  
 |-- LOT\_CAREER\_AREA: integer (nullable = true)  
 |-- LOT\_CAREER\_AREA\_NAME: string (nullable = true)  
 |-- LOT\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_SPECIALIZED\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_SPECIALIZED\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_OCCUPATION\_GROUP: integer (nullable = true)  
 |-- LOT\_OCCUPATION\_GROUP\_NAME: string (nullable = true)  
 |-- LOT\_V6\_SPECIALIZED\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_V6\_SPECIALIZED\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_V6\_OCCUPATION: integer (nullable = true)  
 |-- LOT\_V6\_OCCUPATION\_NAME: string (nullable = true)  
 |-- LOT\_V6\_OCCUPATION\_GROUP: integer (nullable = true)  
 |-- LOT\_V6\_OCCUPATION\_GROUP\_NAME: string (nullable = true)  
 |-- LOT\_V6\_CAREER\_AREA: integer (nullable = true)  
 |-- LOT\_V6\_CAREER\_AREA\_NAME: string (nullable = true)  
 |-- LIGHTCAST\_SECTORS: string (nullable = true)  
 |-- LIGHTCAST\_SECTORS\_NAME: string (nullable = true)  
 |-- NAICS\_2022\_5\_NAME: string (nullable = true)  
 |-- NAICS\_2022\_6: integer (nullable = true)  
 |-- NAICS\_2022\_6\_NAME: string (nullable = true)

# Define columns for EDA:   
# dependent variable: SALARY  
# indepdendent variable: MIN\_YEARS\_EXPERIENCE, SALARY\_FROM, SALARY\_TO, DURATION   
# categorical variables: COMPANY\_IS\_STAFFING, IS\_INTERNSHIP, REMOTE\_TYPE\_NAME, EMPLOYMENT\_TYPE\_NAME, MIN\_EDULEVELS\_NAME, MAX\_EDULEVELS\_NAME, STATE\_NAME  
  
from pyspark.sql.functions import col, pow  
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler  
from pyspark.ml import Pipeline  
  
eda\_columns = [  
 "SALARY",  
 "MIN\_YEARS\_EXPERIENCE", "DURATION",  
 "COMPANY\_IS\_STAFFING", "IS\_INTERNSHIP", "REMOTE\_TYPE\_NAME", "EMPLOYMENT\_TYPE\_NAME",  
 "MIN\_EDULEVELS\_NAME", "STATE\_NAME"  
]  
df\_eda = df.select(eda\_columns)  
df\_eda.show(5, truncate=False)

+------+--------------------+--------+-------------------+-------------+----------------+----------------------+-------------------+----------+  
|SALARY|MIN\_YEARS\_EXPERIENCE|DURATION|COMPANY\_IS\_STAFFING|IS\_INTERNSHIP|REMOTE\_TYPE\_NAME|EMPLOYMENT\_TYPE\_NAME |MIN\_EDULEVELS\_NAME |STATE\_NAME|  
+------+--------------------+--------+-------------------+-------------+----------------+----------------------+-------------------+----------+  
|NULL |2 |6 |false |false |[None] |Full-time (> 32 hours)|Bachelor's degree |Arkansas |  
|NULL |3 |NULL |true |false |Remote |Full-time (> 32 hours)|No Education Listed|Maine |  
|NULL |5 |35 |false |false |[None] |Full-time (> 32 hours)|Bachelor's degree |Texas |  
|NULL |3 |48 |false |false |[None] |Full-time (> 32 hours)|No Education Listed|Arizona |  
|92500 |NULL |15 |false |false |[None] |Part-time / full-time |No Education Listed|California|  
+------+--------------------+--------+-------------------+-------------+----------------+----------------------+-------------------+----------+  
only showing top 5 rows

from pyspark.sql.functions import col, split, explode, regexp\_replace, transform, when, trim, monotonically\_increasing\_id, pow, length, sum as spark\_sum  
import hvplot.pandas  
  
# Visualize the percentage of missing values for each column  
df\_na = df\_eda.select([  
 spark\_sum(  
 when(col(c).isNull() | (length(trim(col(c))) == 0), 1)  
 ).alias(c)  
 for c in df\_eda.columns  
])  
  
df\_na\_pd = df\_na.toPandas().T.reset\_index()  
df\_na\_pd.columns = ["column", "missing\_count"]  
  
total\_rows = df.count()  
df\_na\_pd["missing\_pct"] = df\_na\_pd["missing\_count"] / total\_rows \* 100  
  
  
df\_na\_pd.sort\_values("missing\_pct", ascending=False).hvplot.bar(  
 x="column",  
 y="missing\_pct",  
 title="Percentage of Missing Values by Column",  
 xlabel="Column Name",  
 ylabel="Percentage of Missing Values",  
 rot=45,  
 height=600,  
 width=1000  
)

import pandas as pd  
import hvplot.pandas # make sure this is imported for hvplot support  
  
# Sample a small fraction of the data and convert to Pandas  
df\_sample = df\_eda.sample(fraction=0.05, seed=42).toPandas()  
  
# Create a boolean mask of missing values  
missing\_mask = df\_sample.isnull()  
  
# Melt the mask into long-form format  
missing\_long = (  
 missing\_mask.reset\_index()  
 .melt(id\_vars="index", var\_name="column", value\_name="is\_missing")  
)  
  
# Convert boolean to int (True → 1, False → 0)  
missing\_long["is\_missing"] = missing\_long["is\_missing"].astype(int)  
  
# Plot heatmap  
missing\_long.hvplot.heatmap(  
 x="column", y="index", C="is\_missing",  
 cmap="Reds", colorbar=False,  
 width=900, height=700,  
 title="Heatmap of Missing Values (Sample)"  
).opts(xrotation=45)

from pyspark.sql.functions import countDistinct  
  
# Count number of unique values per column  
df\_eda.select([  
 countDistinct(c).alias(c + "\_nunique")  
 for c in df\_eda.columns  
]).show(truncate=False)  
  
# Select REMOTE\_TYPE\_NAME and MIN\_EDULEVELS\_NAME as the two categorical columns for further inspection  
categorical\_cols = [  
 #"STATE\_NAME",   
 "REMOTE\_TYPE\_NAME",   
 #"EMPLOYMENT\_TYPE\_NAME",  
 "MIN\_EDULEVELS\_NAME",  
 #"COMPANY\_IS\_STAFFING", "IS\_INTERNSHIP"  
]  
  
for colname in categorical\_cols:  
 print(f"\n---- {colname} ----")  
 df\_eda.select(colname).distinct().show(50, truncate=False)

+--------------+----------------------------+----------------+---------------------------+---------------------+------------------------+----------------------------+--------------------------+------------------+  
|SALARY\_nunique|MIN\_YEARS\_EXPERIENCE\_nunique|DURATION\_nunique|COMPANY\_IS\_STAFFING\_nunique|IS\_INTERNSHIP\_nunique|REMOTE\_TYPE\_NAME\_nunique|EMPLOYMENT\_TYPE\_NAME\_nunique|MIN\_EDULEVELS\_NAME\_nunique|STATE\_NAME\_nunique|  
+--------------+----------------------------+----------------+---------------------------+---------------------+------------------------+----------------------------+--------------------------+------------------+  
|6052 |16 |60 |2 |2 |4 |3 |6 |51 |  
+--------------+----------------------------+----------------+---------------------------+---------------------+------------------------+----------------------------+--------------------------+------------------+  
  
  
---- REMOTE\_TYPE\_NAME ----  
  
  
  
  
+----------------+  
|REMOTE\_TYPE\_NAME|  
+----------------+  
|Remote |  
|[None] |  
|Not Remote |  
|Hybrid Remote |  
|NULL |  
+----------------+  
  
  
---- MIN\_EDULEVELS\_NAME ----  
  
  
[Stage 20:> (0 + 1) / 1]  
  
+----------------------------+  
|MIN\_EDULEVELS\_NAME |  
+----------------------------+  
|Bachelor's degree |  
|Ph.D. or professional degree|  
|High school or GED |  
|Master's degree |  
|No Education Listed |  
|Associate degree |  
|NULL |  
+----------------------------+

# For REMOTE\_TYPE\_NAME replace Remote with Remote, [None] with undefined, Not Remote with On Premise, Hybrid Remote with Hybrid, and Null with On Premise   
  
from pyspark.sql.functions import col, when  
  
df\_eda = df\_eda.withColumn(  
 "REMOTE\_TYPE\_NAME",  
 when(col("REMOTE\_TYPE\_NAME") == "Remote", "Remote")  
 .when(col("REMOTE\_TYPE\_NAME") == "[None]", "On Premise")  
 .when(col("REMOTE\_TYPE\_NAME") == "Not Remote", "On Premise")  
 .when(col("REMOTE\_TYPE\_NAME") == "Hybrid Remote", "Hybrid")  
 .when(col("REMOTE\_TYPE\_NAME").isNull(), "On Premise")  
 .otherwise(col("REMOTE\_TYPE\_NAME"))  
)  
  
# create a temporary SQL view if using Spark SQL queries later  
df\_eda.createOrReplaceTempView("df\_eda")  
print(f"\n---- Distinct Values in REMOTE\_TYPE\_NAME ----")  
df\_eda.select("REMOTE\_TYPE\_NAME").distinct().show(10, truncate=False)

---- Distinct Values in REMOTE\_TYPE\_NAME ----  
  
  
[Stage 23:> (0 + 1) / 1]  
  
+----------------+  
|REMOTE\_TYPE\_NAME|  
+----------------+  
|Remote |  
|On Premise |  
|Hybrid |  
+----------------+

# For MIN\_EDULEVELS\_NAME, replace null with No Education Listed  
  
from pyspark.sql.functions import col, when  
  
df\_eda = df\_eda.withColumn(  
 "MIN\_EDULEVELS\_NAME",  
 when(col("MIN\_EDULEVELS\_NAME").isNull(), "No Education Listed")  
 .otherwise(col("MIN\_EDULEVELS\_NAME"))  
)  
  
# create a temporary SQL view if using Spark SQL queries later  
df\_eda.createOrReplaceTempView("df\_eda")  
print(f"\n---- Distinct Values in MIN\_EDULEVELS\_NAME ----")  
df\_eda.select("MIN\_EDULEVELS\_NAME").distinct().show(10, truncate=False)

---- Distinct Values in MIN\_EDULEVELS\_NAME ----  
  
  
[Stage 26:> (0 + 1) / 1]  
  
+----------------------------+  
|MIN\_EDULEVELS\_NAME |  
+----------------------------+  
|Bachelor's degree |  
|Ph.D. or professional degree|  
|High school or GED |  
|Master's degree |  
|No Education Listed |  
|Associate degree |  
+----------------------------+

# Calculate median for DURATION  
median\_DURATION = df\_eda.approxQuantile("DURATION", [0.5], 0.01)[0]  
  
# Check for nulls in DURATION and impute with median  
  
df\_eda = df\_eda.withColumn(  
 "DURATION",  
 when(col("DURATION").isNull(), median\_DURATION)  
 .otherwise(col("DURATION"))  
)

# Feature Engineering

For this section, we will define key features for modeling based on relevance to the target variable (SALARY). Categorical variables related to experience, duration, remote status, and employment type were chosen as features to be encoded using StringIndexer and OneHotEncoder and assembled into a vector in preparation for modeling.

## FEATURE ENGINEERING  
  
from pyspark.sql.functions import col, pow  
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler  
from pyspark.ml import Pipeline  
  
# Drop rows with NA Values   
df\_eda = df\_eda.dropna(subset=[  
 "SALARY",  
 "MIN\_YEARS\_EXPERIENCE", "DURATION",  
 "COMPANY\_IS\_STAFFING", "IS\_INTERNSHIP", "REMOTE\_TYPE\_NAME", "EMPLOYMENT\_TYPE\_NAME",  
 "MIN\_EDULEVELS\_NAME", "STATE\_NAME"  
])  
  
# Define categorical columns to encode  
categorical\_cols = [  
 "MIN\_EDULEVELS\_NAME",   
 "REMOTE\_TYPE\_NAME"  
]  
  
# Index and One-Hot Encode  
indexers = [  
 StringIndexer(  
 inputCol=col,   
 outputCol=f"{col}\_idx",   
 handleInvalid="skip"  
 ) for col in categorical\_cols  
]  
  
encoders = [  
 OneHotEncoder(  
 inputCol=f"{col}\_idx",   
 outputCol=f"{col}\_vec"  
 ) for col in categorical\_cols  
]

# Assemble base features (for GLR and Random Forest)  
assembler = VectorAssembler(  
 inputCols=[  
 "MIN\_YEARS\_EXPERIENCE", "DURATION",  
 "IS\_INTERNSHIP", "COMPANY\_IS\_STAFFING"  
 ] + [f"{col}\_vec" for col in categorical\_cols],  
 outputCol="features"  
)  
  
# Build pipeline and transform df\_eda  
pipeline = Pipeline(stages=indexers + encoders + [assembler])  
data = pipeline.fit(df\_eda).transform(df\_eda)  
  
# Create squared term for Polynomial Regression  
data = data.withColumn("MIN\_YEARS\_EXPERIENCE\_SQ", pow(col("MIN\_YEARS\_EXPERIENCE"), 2))  
  
# Assemble polynomial features  
assembler\_poly = VectorAssembler(  
 inputCols=[  
 "MIN\_YEARS\_EXPERIENCE", "MIN\_YEARS\_EXPERIENCE\_SQ", "DURATION",  
 "IS\_INTERNSHIP", "COMPANY\_IS\_STAFFING"  
 ] + [f"{col}\_vec" for col in categorical\_cols],  
 outputCol="features\_poly"  
)  
  
# Transform with polynomial features  
data = assembler\_poly.transform(data)  
  
# Show sample of features and label  
data.select("SALARY", "features", "features\_poly").show(5, truncate=False)

+------+----------------------------------+-------------------------------------------+  
|SALARY|features |features\_poly |  
+------+----------------------------------+-------------------------------------------+  
|92962 |(11,[0,1,4,9],[2.0,18.0,1.0,1.0]) |(12,[0,1,2,5,10],[2.0,4.0,18.0,1.0,1.0]) |  
|107645|(11,[0,1,7,9],[10.0,18.0,1.0,1.0])|(12,[0,1,2,8,10],[10.0,100.0,18.0,1.0,1.0])|  
|192800|(11,[0,1,4,9],[6.0,55.0,1.0,1.0]) |(12,[0,1,2,5,10],[6.0,36.0,55.0,1.0,1.0]) |  
|125900|(11,[0,1,6,9],[12.0,18.0,1.0,1.0])|(12,[0,1,2,7,10],[12.0,144.0,18.0,1.0,1.0])|  
|170000|(11,[0,1,5,9],[6.0,18.0,1.0,1.0]) |(12,[0,1,2,6,10],[6.0,36.0,18.0,1.0,1.0]) |  
+------+----------------------------------+-------------------------------------------+  
only showing top 5 rows

# Train/Test Split

The dataset containing vectorized features will be split into 80% for Training and 20% for Testing to ensure that the model evaluation is unbiased. This split will be applied throughout the various models (GLR, Polynomial GLR, and Random Forest) to allow us to compare the results.

# Split the data into training and testing sets  
train\_data, test\_data = data.randomSplit([0.8, 0.2], seed=42)  
  
# Confirm the split sizes  
print(f"Training rows: {train\_data.count()}")  
print(f"Testing rows: {test\_data.count()}")

Training rows: 18966  
  
  
[Stage 46:> (0 + 1) / 1]  
  
Testing rows: 4731

# GLR Model Summary

The GLR model yielded an intercept of $118,078 with several of the features demonstrating statistical significance. Among those features, the most influential and positive coefficient is MIN\_YEARS\_EXPERIENCE (+$7441.28, p=0.0008), indicating that higher salaries roles often require more years of experience. On the otherhand, the most influential and statistically significant negative coefficient is lower education levels, specifically ‘Associated Degree’ (-$65370, p>0001). It’s also worth noting that IS\_INTERNSHIP and COMPANY\_IS\_STAFFING are both statistically significant negative coefficients, suggesting that internships and staffing firms tend to offer lower salaries.

## GLR  
from pyspark.ml.regression import GeneralizedLinearRegression  
from pyspark.ml.evaluation import RegressionEvaluator  
from IPython.display import HTML  
import pandas as pd  
  
feature\_names = assembler.getInputCols()  
  
# Define Generalized Linear Regression model  
glr = GeneralizedLinearRegression(  
 featuresCol="features",  
 labelCol="SALARY",  
 family="gaussian", # continuous outcome, normal distribution  
 link="identity", # standard linear regression  
 maxIter=10, # number of iterations   
 regParam=0.3 # regularization parameter (lambda)  
)  
  
# Fit model on training data  
glr\_model = glr.fit(train\_data)  
  
# evaluate RMSE on test data   
glr\_predictions = glr\_model.transform(test\_data)  
  
evaluator = RegressionEvaluator(  
 labelCol="SALARY",  
 predictionCol="prediction",  
 metricName="rmse"  
)  
glr\_rmse = evaluator.evaluate(glr\_predictions)  
print(f"\nGLR RMSE on Test Data: {glr\_rmse:.2f}")  
  
# Get model summary  
summary = glr\_model.summary  
  
# Coefficients and Intercept  
# Intercept represents the predicted salary when all input features = 0.  
# Coefficient represents the estimated change in salary per one-unit change in respective feature (while everything else is constant).  
print("Intercept: {:.4f}".format(glr\_model.intercept))  
print("Coefficients:")  
for i, coef in enumerate(glr\_model.coefficients):  
 print(f" Feature {i+1}: {coef:.4f}")  
  
# Regression Summary  
#Smaller coefficient standard error means more reliable estimates  
print("\n--- Regression Summary: ---")  
print("Coefficient Standard Errors:",[f"{val:.4f}" for val in summary.coefficientStandardErrors])  
  
# T-values measures how many standard errors the coefficient is away from 0; >2 is statistically significant  
# P-values measure the probability that the coefficient is actually 0; <0.05 is statistically significant  
print("T Values:", [f"{val:.4f}" for val in summary.tValues])  
print("P Values:", [f"{val:.4f}" for val in summary.pValues])  
  
# Dispersion - variance of residuals  
print(f"\nDispersion: {summary.dispersion:.4f}")  
# Null Deviance - how much “unexplained” variation exists with no predictors.  
print(f"Null Deviance: {summary.nullDeviance:.4f}")  
# Residual DF Null - how many data points are left after accounting for 1 parameter (the mean)  
print(f"Residual DF Null: {summary.residualDegreeOfFreedomNull}")  
# Residual Deviance - Variation not explained by the model. Smaller deviance = better fit.  
print(f"Deviance: {summary.deviance:.4f}")  
print(f"Residual DF: {summary.residualDegreeOfFreedom}")  
# AIC - metric for model quality that balances goodness of fit and model complexity. Lower AIC = better model  
print(f"AIC: {summary.aic:.4f}")  
  
# Presentation  
# get feature names from GLR summary (Java backend)  
feature\_names = summary.\_call\_java("featureNames")  
features = ["Intercept"] + feature\_names  
  
# build all stats lists (include intercept)  
coefs = [glr\_model.intercept] + list(glr\_model.coefficients)  
se = list(summary.coefficientStandardErrors)  
tvals = list(summary.tValues)  
pvals = list(summary.pValues)  
  
# sanity check  
print("--- This is a diagnostic check ---")  
print("Length of features:", len(features))  
print("Length of coefs:", len(coefs))  
print("Length of se:", len(se))  
print("Length of tvals:", len(tvals))  
print("Length of pvals:", len(pvals))  
  
# create summary DataFrame  
coef\_table = pd.DataFrame({  
 "Feature": features,  
 "Estimate": [f"{v:.4f}" if v is not None else None for v in coefs],  
 "Std Error": [f"{v:.4f}" if v is not None else None for v in se],  
 "t-Value": [f"{v:.4f}" if v is not None else None for v in tvals],  
 "p-Value": [f"{v:.4f}" if v is not None else None for v in pvals]  
})  
  
# export to CSV  
coef\_table.to\_csv("output/glr\_summary.csv", index=False)  
  
# display in notebook  
HTML(coef\_table.to\_html(index=False))

GLR RMSE on Test Data: 35634.53  
Intercept: 118077.7423  
Coefficients:  
 Feature 1: 7441.2826  
 Feature 2: -73.8928  
 Feature 3: -3048.7849  
 Feature 4: -3147.0268  
 Feature 5: -31047.8353  
 Feature 6: -27419.8882  
 Feature 7: -65370.0192  
 Feature 8: -65034.1694  
 Feature 9: 5845.2032  
 Feature 10: -1291.2053  
 Feature 11: -1726.5664  
  
--- Regression Summary: ---  
  
  
  
  
Coefficient Standard Errors: ['80.9467', '21.9980', '4744.7230', '876.2733', '7975.2611', '7997.4419', '8037.8830', '8030.3752', '8100.2761', '1473.5361', '1535.6262', '8125.5372']  
T Values: ['91.9281', '-3.3591', '-0.6426', '-3.5914', '-3.8930', '-3.4286', '-8.1327', '-8.0985', '0.7216', '-0.8763', '-1.1243', '14.5317']  
P Values: ['0.0000', '0.0008', '0.5205', '0.0003', '0.0001', '0.0006', '0.0000', '0.0000', '0.4705', '0.3809', '0.2609', '0.0000']  
  
Dispersion: 1210670973.7533  
  
  
[Stage 54:> (0 + 1) / 1]  
  
Null Deviance: 35794690345776.1094  
Residual DF Null: 18965  
Deviance: 22947057636519.6797  
Residual DF: 18954  
  
  
[Stage 55:> (0 + 1) / 1]  
  
AIC: 450500.4524  
--- This is a diagnostic check ---  
Length of features: 12  
Length of coefs: 12  
Length of se: 12  
Length of tvals: 12  
Length of pvals: 12

Feature

Estimate

Std Error

t-Value

p-Value

Intercept

118077.7423

80.9467

91.9281

0.0000

MIN\_YEARS\_EXPERIENCE

7441.2826

21.9980

-3.3591

0.0008

DURATION

-73.8928

4744.7230

-0.6426

0.5205

IS\_INTERNSHIP

-3048.7849

876.2733

-3.5914

0.0003

COMPANY\_IS\_STAFFING

-3147.0268

7975.2611

-3.8930

0.0001

MIN\_EDULEVELS\_NAME\_vec\_Bachelor’s degree

-31047.8353

7997.4419

-3.4286

0.0006

MIN\_EDULEVELS\_NAME\_vec\_No Education Listed

-27419.8882

8037.8830

-8.1327

0.0000

MIN\_EDULEVELS\_NAME\_vec\_Associate degree

-65370.0192

8030.3752

-8.0985

0.0000

MIN\_EDULEVELS\_NAME\_vec\_High school or GED

-65034.1694

8100.2761

0.7216

0.4705

MIN\_EDULEVELS\_NAME\_vec\_Master’s degree

5845.2032

1473.5361

-0.8763

0.3809

REMOTE\_TYPE\_NAME\_vec\_On Premise

-1291.2053

1535.6262

-1.1243

0.2609

REMOTE\_TYPE\_NAME\_vec\_Remote

-1726.5664

8125.5372

14.5317

0.0000

# Polynomial GLR Model Summary

The Polynomial GLR model yielded an intercept of $110,905, slightly lower than the linear GLR model. MIN\_YEARS\_EXPERIENCE remained the most influential positive predictor (+$11948.86, p < 0.0001), while MIN\_YEARS\_EXPERIENCE\_SQ was statistically significant but negative (−$343.47, p = 0.0004). This indicates that salary growth sees a plateau with increasing years of experience. As with the linear GLR model, the statistically significant negative predictors were lower education levels and roles offered by staffing firms. Notably, IS\_INTERNSHIP was a statistically significant positive predictor in this model, highlighting the unpredictable effect that the added polynomial effect may have on the model’s behavior.

## Poly GLR  
from pyspark.ml.regression import GeneralizedLinearRegression  
from IPython.display import HTML  
import pandas as pd  
  
# Define GLR model for polynomial features  
poly\_glr = GeneralizedLinearRegression(  
 featuresCol="features\_poly",  
 labelCol="SALARY",  
 family="gaussian",  
 link="identity",  
 maxIter=10,  
 regParam=0.3  
)  
  
# Fit model using training data  
poly\_glr\_model = poly\_glr.fit(train\_data)  
  
# Evaluate RMSE for poly test data  
poly\_predictions = poly\_glr\_model.transform(test\_data)  
poly\_rmse = evaluator.evaluate(poly\_predictions)  
print(f"Polynomial GLR RMSE: {poly\_rmse:.2f}")  
  
# Get model summary  
poly\_summary = poly\_glr\_model.summary  
  
# Coefficients and Intercept  
print("Intercept: {:.4f}".format(poly\_glr\_model.intercept))  
print("Coefficients:")  
for i, coef in enumerate(poly\_glr\_model.coefficients):  
 print(f" Feature {i+1}: {coef:.4f}")  
  
# Regression Summary  
print("\n--- Regression Summary: ---")  
print("Coefficient Standard Errors:",[f"{val:.4f}" for val in poly\_summary.coefficientStandardErrors])  
  
# T-values measures how many standard errors the coefficient is away from 0; >2 is statistically significant  
# P-values measure the probability that the coefficient is actually 0; <0.05 is statistically significant  
print("T Values:", [f"{val:.4f}" for val in poly\_summary.tValues])  
print("P Values:", [f"{val:.4f}" for val in poly\_summary.pValues])  
  
# Dispersion - variance of residuals  
print(f"\nDispersion: {poly\_summary.dispersion:.4f}")  
# Null Deviance - how much “unexplained” variation exists with no predictors.  
print(f"Null Deviance: {poly\_summary.nullDeviance:.4f}")  
# Residual DF Null - how many data points are left after accounting for 1 parameter (the mean)  
print(f"Residual DF Null: {poly\_summary.residualDegreeOfFreedomNull}")  
# Residual Deviance - Variation not explained by the model. Smaller deviance = better fit.  
print(f"Deviance: {poly\_summary.deviance:.4f}")  
print(f"Residual DF: {poly\_summary.residualDegreeOfFreedom}")  
# AIC - metric for model quality that balances goodness of fit and model complexity. Lower AIC = better model  
print(f"AIC: {poly\_summary.aic:.4f}")  
  
# Presentation  
# get feature names from poly GLR summary (Java backend)  
feature\_names = poly\_summary.\_call\_java("featureNames")  
features = ["Intercept"] + feature\_names  
  
# build all stats lists (include intercept)  
coefs = [poly\_glr\_model.intercept] + list(poly\_glr\_model.coefficients)  
se = list(poly\_summary.coefficientStandardErrors)  
tvals = list(poly\_summary.tValues)  
pvals = list(poly\_summary.pValues)  
  
# sanity check  
print("--- This is a diagnostic check ---")  
print("Length of features:", len(features))  
print("Length of coefs:", len(coefs))  
print("Length of se:", len(se))  
print("Length of tvals:", len(tvals))  
print("Length of pvals:", len(pvals))  
  
# create summary DataFrame  
poly\_coef\_table = pd.DataFrame({  
 "Feature": features,  
 "Estimate": [f"{v:.4f}" if v is not None else None for v in coefs],  
 "Std Error": [f"{v:.4f}" if v is not None else None for v in se],  
 "t-Value": [f"{v:.4f}" if v is not None else None for v in tvals],  
 "p-Value": [f"{v:.4f}" if v is not None else None for v in pvals]  
})  
  
# export to CSV  
poly\_coef\_table.to\_csv("output/glr\_summary.csv", index=False)  
  
# display in notebook  
HTML(poly\_coef\_table.to\_html(index=False))

Polynomial GLR RMSE: 35513.74  
Intercept: 110905.3631  
Coefficients:  
 Feature 1: 11948.8614  
 Feature 2: -343.4661  
 Feature 3: -76.8774  
 Feature 4: 626.9998  
 Feature 5: -3415.3221  
 Feature 6: -34843.3820  
 Feature 7: -31037.9782  
 Feature 8: -63953.6989  
 Feature 9: -68645.3489  
 Feature 10: 3549.0876  
 Feature 11: -1355.3830  
 Feature 12: -1408.9606  
  
--- Regression Summary: ---  
  
  
  
  
Coefficient Standard Errors: ['277.6073', '20.2478', '21.8341', '4714.1994', '869.8596', '7918.7412', '7940.4594', '7978.1695', '7973.1235', '8040.7983', '1462.5140', '1524.2495', '8075.8072']  
T Values: ['43.0423', '-16.9632', '-3.5210', '0.1330', '-3.9263', '-4.4001', '-3.9088', '-8.0161', '-8.6096', '0.4414', '-0.9267', '-0.9244', '13.7330']  
P Values: ['0.0000', '0.0000', '0.0004', '0.8942', '0.0001', '0.0000', '0.0001', '0.0000', '0.0000', '0.6589', '0.3541', '0.3553', '0.0000']  
  
Dispersion: 1192618999.9018  
  
  
  
  
Null Deviance: 35794690345776.1094  
Residual DF Null: 18965  
Deviance: 22603707905139.5312  
Residual DF: 18953  
  
  
[Stage 62:> (0 + 1) / 1]  
  
AIC: 450216.5255  
--- This is a diagnostic check ---  
Length of features: 13  
Length of coefs: 13  
Length of se: 13  
Length of tvals: 13  
Length of pvals: 13

Feature

Estimate

Std Error

t-Value

p-Value

Intercept

110905.3631

277.6073

43.0423

0.0000

MIN\_YEARS\_EXPERIENCE

11948.8614

20.2478

-16.9632

0.0000

MIN\_YEARS\_EXPERIENCE\_SQ

-343.4661

21.8341

-3.5210

0.0004

DURATION

-76.8774

4714.1994

0.1330

0.8942

IS\_INTERNSHIP

626.9998

869.8596

-3.9263

0.0001

COMPANY\_IS\_STAFFING

-3415.3221

7918.7412

-4.4001

0.0000

MIN\_EDULEVELS\_NAME\_vec\_Bachelor’s degree

-34843.3820

7940.4594

-3.9088

0.0001

MIN\_EDULEVELS\_NAME\_vec\_No Education Listed

-31037.9782

7978.1695

-8.0161

0.0000

MIN\_EDULEVELS\_NAME\_vec\_Associate degree

-63953.6989

7973.1235

-8.6096

0.0000

MIN\_EDULEVELS\_NAME\_vec\_High school or GED

-68645.3489

8040.7983

0.4414

0.6589

MIN\_EDULEVELS\_NAME\_vec\_Master’s degree

3549.0876

1462.5140

-0.9267

0.3541

REMOTE\_TYPE\_NAME\_vec\_On Premise

-1355.3830

1524.2495

-0.9244

0.3553

REMOTE\_TYPE\_NAME\_vec\_Remote

-1408.9606

8075.8072

13.7330

0.0000

# Random Forest Model

As shown in the feature importance plot, MIN\_YEARS\_EXPERIENCE stands out as the most influential predictor, making up for over 70% of the model’s predictive power. This confirms the finding across all models that years of experience is a critical driver of salary. Following that, education-related features — “High School or GED” and “Master’s” — also contributed meaningfully. Features such as “Duration”, “On-Premise”, “Remote”, and “Staffing Agency” showed minimal impact in the RF model’s decision-making, which helps us better understand the importance of the output from GLR-based models.

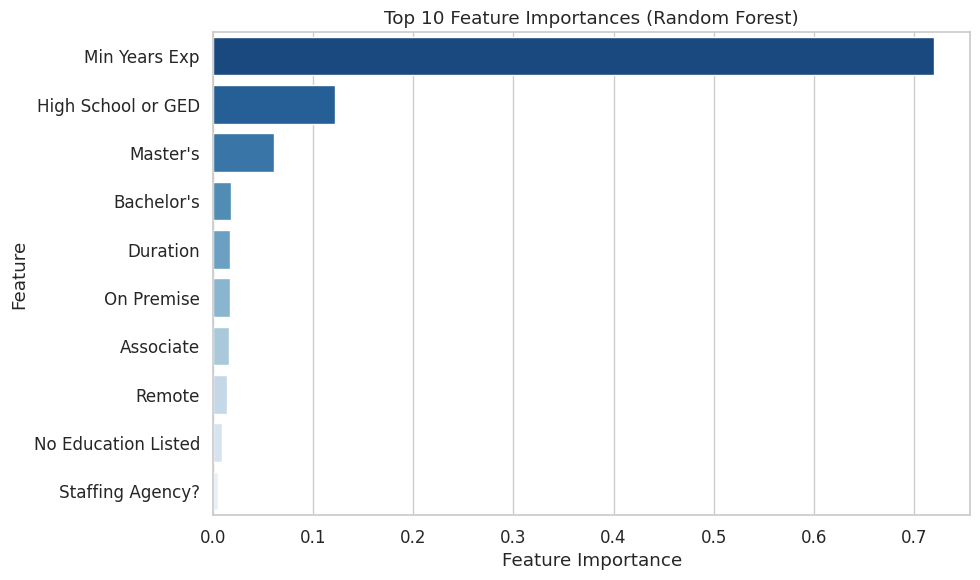
## Random Forest   
from pyspark.ml.regression import RandomForestRegressor  
from pyspark.ml.evaluation import RegressionEvaluator  
  
# Define the model  
rf = RandomForestRegressor(  
 featuresCol="features",  
 labelCol="SALARY",  
 numTrees=200,  
 maxDepth=6,  
 seed=42  
)  
  
# Train model on training data  
rf\_model = rf.fit(train\_data)  
  
# Make predictions  
rf\_predictions = rf\_model.transform(test\_data)  
  
# Evaluate RMSE on RF test data  
rf\_rmse = evaluator.evaluate(rf\_predictions)  
print(f"Random Forest RMSE: {rf\_rmse:.2f}")  
  
# R-squared  
evaluator.setMetricName("r2")  
r2 = evaluator.evaluate(rf\_predictions)  
print(f"R-squared: {r2:.4f}")

25/10/09 01:21:09 WARN DAGScheduler: Broadcasting large task binary with size 1118.7 KiB  
25/10/09 01:21:11 WARN DAGScheduler: Broadcasting large task binary with size 1907.1 KiB  
   
  
Random Forest RMSE: 34478.66  
  
  
[Stage 80:> (0 + 1) / 1]  
  
R-squared: 0.3884

## Feature Importance Plot  
from pyspark.sql.functions import col, pow  
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler  
from pyspark.ml import Pipeline  
import numpy as np  
import pandas as pd  
  
# Reconstruct and expand feature names   
# refit pipeline to get OneHotEncoderModels  
fitted\_pipeline = pipeline.fit(df\_eda)  
  
# extract OneHotEncoder stages   
ohe\_feature\_names = []  
for i, col in enumerate(categorical\_cols):  
 encoder\_model = fitted\_pipeline.stages[len(indexers) + i] # each encoder is separate  
 num\_categories = encoder\_model.categorySizes[0] - 1 # dropLast=True  
 ohe\_feature\_names += [f"{col}\_vec\_{j}" for j in range(num\_categories)]  
  
# combine full list in correct order  
feature\_names = [  
 "MIN\_YEARS\_EXPERIENCE", "DURATION",  
 "IS\_INTERNSHIP", "COMPANY\_IS\_STAFFING"  
] + ohe\_feature\_names  
  
# extract feature importances from RF model  
importances = rf\_model.featureImportances.toArray()  
feat\_imp\_df = pd.DataFrame({  
 "Feature": feature\_names,  
 "Importance": importances  
})  
  
# Select top 10  
#feat\_imp\_df = feat\_imp\_df.sort\_values("Importance", ascending=False)  
#top10\_df = feat\_imp\_df.head(10)

import seaborn as sns  
import matplotlib.pyplot as plt  
import os  
  
# mapping categories for MIN\_EDULEVELS\_NAME  
edu\_map = {  
 "MIN\_EDULEVELS\_NAME\_vec\_0": "Bachelor's",  
 "MIN\_EDULEVELS\_NAME\_vec\_1": "No Education Listed",  
 "MIN\_EDULEVELS\_NAME\_vec\_2": "Associate",  
 "MIN\_EDULEVELS\_NAME\_vec\_3": "High School or GED",  
 "MIN\_EDULEVELS\_NAME\_vec\_4": "Master's"  
}  
  
# mapping categories for REMOTE\_TYPE\_NAME  
remote\_map = {  
 "REMOTE\_TYPE\_NAME\_vec\_0": "On Premise",  
 "REMOTE\_TYPE\_NAME\_vec\_1": "Remote"  
}  
  
# combine all mappings  
feature\_label\_map = {  
 \*\*edu\_map,  
 \*\*remote\_map,  
 "MIN\_YEARS\_EXPERIENCE": "Min Years Exp",  
 "DURATION": "Duration",  
 "IS\_INTERNSHIP": "Internship?",  
 "COMPANY\_IS\_STAFFING": "Staffing Agency?"  
}  
  
# Apply mapping  
feat\_imp\_df["Feature\_Clean"] = feat\_imp\_df["Feature"].map(feature\_label\_map).fillna(feat\_imp\_df["Feature"])  
  
top10\_df = feat\_imp\_df.sort\_values("Importance", ascending=False).head(10)  
  
# Create output folder  
os.makedirs("output", exist\_ok=True)  
  
# Create bar plot  
sns.set\_theme(style="whitegrid", font\_scale=1.1)  
plt.figure(figsize=(10, 6))  
sns.barplot(  
 x="Importance",   
 y="Feature\_Clean",   
 data=top10\_df,   
 palette="Blues\_r"  
)  
plt.title("Top 10 Feature Importances (Random Forest)")  
plt.xlabel("Feature Importance")  
plt.ylabel("Feature")  
plt.tight\_layout()  
  
plt.savefig("output/rf\_feature\_importance.png", dpi=300)  
plt.show()

/tmp/ipykernel\_1714/1329682924.py:41: FutureWarning:  
  
  
  
Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.



png

# Model Comparison

To evaluate the predictive performance of the GLR, Polynomial GLR, and Random Forest models, the following metrics were used for comparison:

* RMSE: measurement of the average prediction error in $.
* AIC: assessment of model fit while penalizing complexity; lower is better.
* BIC: similar to AIC but applies a heavier penalty for model complexity.

## Model Comparison Table  
  
import math  
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
from pyspark.ml.evaluation import RegressionEvaluator  
  
# Predict on test data  
glr\_preds = glr\_model.transform(test\_data)  
poly\_preds = poly\_glr\_model.transform(test\_data)  
rf\_preds = rf\_model.transform(test\_data)  
  
# Define evaluator for RMSE  
evaluator = RegressionEvaluator(labelCol="SALARY", predictionCol="prediction", metricName="rmse")  
  
glr\_rmse = evaluator.evaluate(glr\_preds)  
poly\_rmse = evaluator.evaluate(poly\_preds)  
rf\_rmse = evaluator.evaluate(rf\_preds)  
  
# print("GLR RMSE:", round(glr\_rmse, 2))  
# print("Polynomial GLR RMSE:", round(poly\_rmse, 2))  
# print("Random Forest RMSE:", round(rf\_rmse, 2))  
  
# Define AIC from model summary  
glr\_aic = glr\_model.summary.aic  
poly\_aic = poly\_glr\_model.summary.aic  
  
  
# Calculate BIC  
def compute\_bic(summary, k):  
 n = summary.numInstances  
 dev = summary.deviance  
 disp = summary.dispersion  
 logL = -0.5 \* (n \* math.log(2 \* math.pi) + n \* math.log(disp) + dev / disp)  
 return k \* math.log(n) - 2 \* logL  
  
glr\_bic = compute\_bic(glr\_model.summary, len(glr\_model.coefficients))  
poly\_bic = compute\_bic(poly\_glr\_model.summary, len(poly\_glr\_model.coefficients))  
  
  
# Define comparison table and export to .csv  
comparison\_df = pd.DataFrame({  
 "Model": ["GLR", "Polynomial GLR", "Random Forest"],  
 "RMSE": [round(glr\_rmse, 2), round(poly\_rmse, 2), round(rf\_rmse, 2)],  
 "AIC": [round(glr\_aic, 2), round(poly\_aic, 2), None],  
 "BIC": [round(glr\_bic, 2), round(poly\_bic, 2), None]  
})  
  
display(comparison\_df)

[Stage 95:> (0 + 1) / 1]  
  
Model Comparison Summary:

Model

RMSE

AIC

BIC

0

GLR

35634.53

450500.45

450582.81

1

Polynomial GLR

35513.74

450216.53

450306.73

2

Random Forest

34478.66

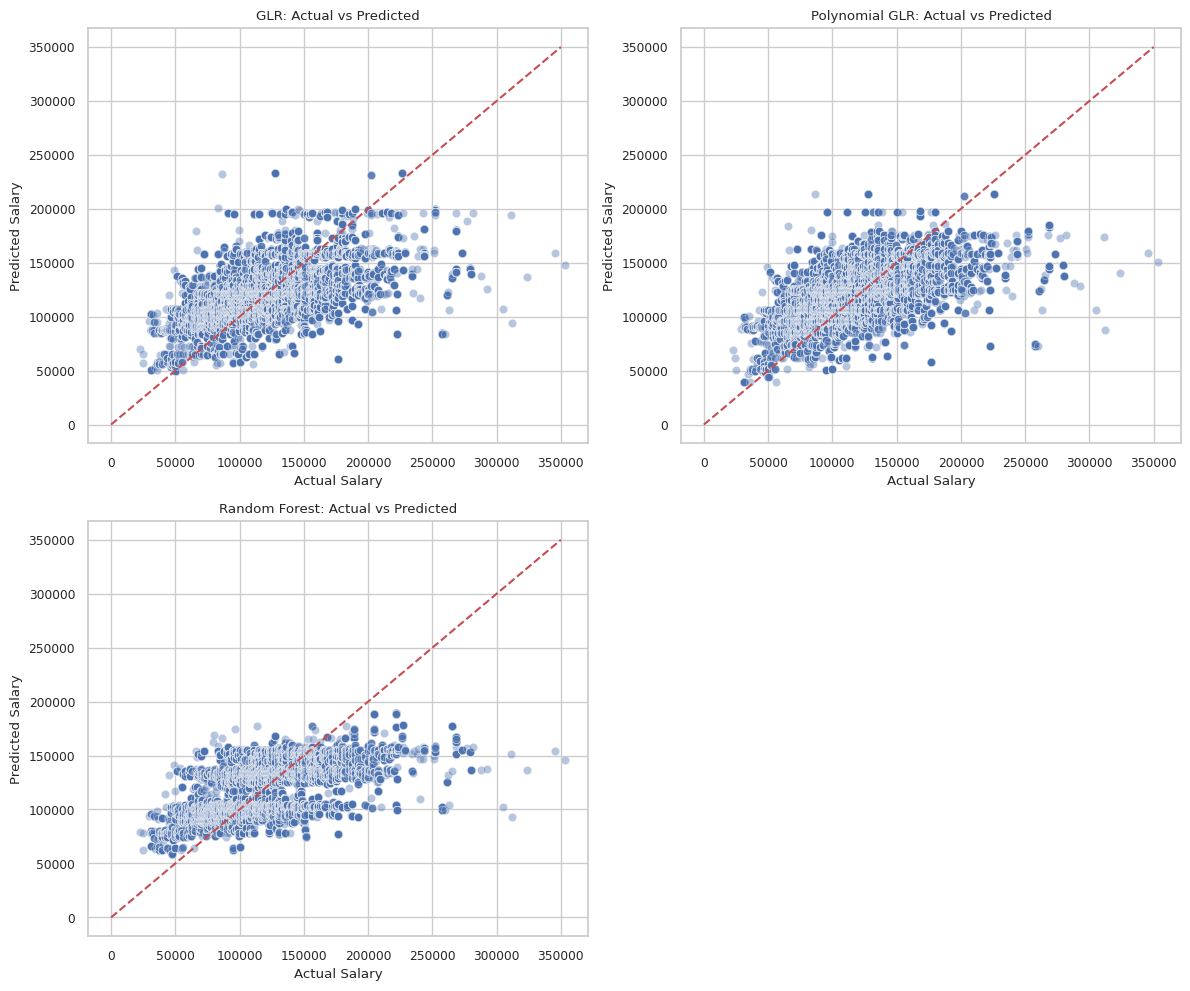
NaN

NaN

Based on RMSE, the RF model outperformed the GLR models in terms of predictive accuracy. However, since AIC and BIC values are not applicable to RF, this model cannot be relied on solely to interpret model complexity. While Poly GLR outperformed the standard GLR in both RMSE and AIC/BIC, the margin of improvement is not substaitial enough to justify the added compleixty of the polynomial term. A hybrid approach of the RF and standard GLR model will ensure accuracy for salary predictions and better interpretibility into the key drivers of salary.

# Actual vs. Predicted Salary: Model Comparison

## Actual vs. Predicted Salary Comparison Plots  
  
# Keep the SALARY column in all 3 prediction DataFrames  
glr\_pd = glr\_preds.select("SALARY", "prediction").withColumnRenamed("prediction", "GLR\_Pred")  
poly\_pd = poly\_preds.select("SALARY", "prediction").withColumnRenamed("prediction", "Poly\_Pred")  
rf\_pd = rf\_preds.select("SALARY", "prediction").withColumnRenamed("prediction", "RF\_Pred")  
  
# Now join using the shared column SALARY  
combined\_preds = glr\_pd \  
 .join(poly\_pd, on="SALARY") \  
 .join(rf\_pd, on="SALARY")  
  
# Convert to pandas  
combined\_df = combined\_preds.toPandas()  
  
import seaborn as sns  
import matplotlib.pyplot as plt  
  
sns.set\_theme(style="whitegrid", font\_scale=0.8)  
fig, axes = plt.subplots(2, 2, figsize=(12, 10))  
  
# Plot for GLR  
sns.scatterplot(data=combined\_df, x="SALARY", y="GLR\_Pred", ax=axes[0, 0], alpha=0.4)  
axes[0, 0].plot([0, 350000], [0, 350000], 'r--')  
axes[0, 0].set\_title("GLR: Actual vs Predicted")  
axes[0, 0].set\_xlabel("Actual Salary")  
axes[0, 0].set\_ylabel("Predicted Salary")  
  
  
# Plot for Polynomial GLR  
sns.scatterplot(data=combined\_df, x="SALARY", y="Poly\_Pred", ax=axes[0, 1], alpha=0.4)  
axes[0, 1].plot([0, 350000], [0, 350000], 'r--')  
axes[0, 1].set\_title("Polynomial GLR: Actual vs Predicted")  
axes[0, 1].set\_xlabel("Actual Salary")  
axes[0, 1].set\_ylabel("Predicted Salary")  
  
# Plot for Random Forest  
sns.scatterplot(data=combined\_df, x="SALARY", y="RF\_Pred", ax=axes[1, 0], alpha=0.4)  
axes[1, 0].plot([0, 350000], [0, 350000], 'r--')  
axes[1, 0].set\_title("Random Forest: Actual vs Predicted")  
axes[1, 0].set\_xlabel("Actual Salary")  
axes[1, 0].set\_ylabel("Predicted Salary")  
  
# Hide bottom-right subplot  
axes[1, 1].axis("off")  
  
plt.tight\_layout()  
plt.savefig("output/model\_comparison\_plots.png", dpi=300)  
plt.show()



png

Key Observations: - GLR: The GLR model predictions generally follow the trend of actual salaries but with greater dispersion at higher salary levels. The model is underpredicting high salaries and overpredicting some lower salaries, demonstrating the limitation of linear assumptions.

* Polynomial GLR: The polynomial model shows a tighter dispersion at mid range salaries compared to GLR; but similarly, it is undepredicting and overpredicting salaries at the higher and lower extremes, respectively.
* Random Forest: The Random Forest model demonstrates the tightest clustering around the diagonal, which suggests the best predictive accuracy among the models. However, it also underpredicts higher salary levels (>$200,000), which is expected since the random forest model tends to average results.