Assignment 04

Joshua Lawrence

October 8, 2025

# 1. Load the Dataset

from pyspark.sql import SparkSession  
from pyspark.sql.functions import col  
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler  
from pyspark.ml import Pipeline  
from pyspark.ml.regression import LinearRegression, RandomForestRegressor  
from pyspark.ml.evaluation import RegressionEvaluator  
import pandas as pd  
import plotly.express as px  
import plotly.io as pio  
import numpy as np  
import matplotlib.pyplot as plt  
import seaborn as sns  
import math  
  
# Set random seed for reproducibility  
np.random.seed(42)  
pio.renderers.default = "notebook+notebook\_connected+vscode"  
  
# Initialize Spark Session  
spark = SparkSession.builder.appName("LightcastData").getOrCreate()  
  
# Load Data  
df = spark.read.option("header", "true").option("inferSchema", "true").option("multiLine","true").option("escape", "\"").csv("data/lightcast\_job\_postings.csv")  
  
# Show Schema and Sample Data  
# 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).  
25/10/08 23:36:50 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable  
[Stage 0:> (0 + 1) / 1] [Stage 1:> (0 + 1) / 1] 25/10/08 23:37:11 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'.  
[Stage 2:> (0 + 1) / 1]

+--------------------+-----------------+----------------------+----------+--------+---------+--------+--------------------+--------------------+--------------------+-----------+-------------------+--------------------+--------------------+---------------+----------------+--------+--------------------+-----------+-------------------+----------------+---------------------+-------------+-------------------+-------------+------------------+---------------+--------------------+--------------------+--------------------+-------------+------+-----------+----------------+-------------------+---------+-----------+--------------------+--------------------+-------------+------+--------------+-----+--------------------+-----+----------+---------------+--------------------+---------------+--------------------+------------+--------------------+------------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------+--------------------+------------------+-------------------+--------------------+--------------------+--------------------+--------------------+-----------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+--------------------+----------+--------------------+----------+--------------------+----------+---------------+----------+---------------+---------------+--------------------+--------------+--------------------+--------------------------+-------------------------------+--------------------+-------------------------+-----------------------------+----------------------------------+-----------------+----------------------+-----------------------+----------------------------+------------------+-----------------------+-------+--------------------+-------+--------------------+-------+---------------+-------+---------------+-----------------+----------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+------------+--------------------+  
| 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

# 2. Feature Engineering

# Define features and target variable  
continuous\_features = ['MIN\_YEARS\_EXPERIENCE', 'MAX\_YEARS\_EXPERIENCE', 'SALARY\_FROM', 'SALARY\_TO']  
categorical\_features = ['REMOTE\_TYPE\_NAME', 'EMPLOYMENT\_TYPE\_NAME']  
target\_variable = 'SALARY'  
  
# Select columns and drop missing values  
selected\_columns = continuous\_features + categorical\_features + [target\_variable]  
df\_cleaned = df.select(\*selected\_columns).na.drop()  
  
# Create StringIndexers for categorical variables  
indexers = [  
 StringIndexer(inputCol=col\_name, outputCol=col\_name + "\_index", handleInvalid="keep")  
 for col\_name in categorical\_features  
]  
  
# Create OneHotEncoders for indexed categorical variables  
encoders = [  
 OneHotEncoder(inputCol=col\_name + "\_index", outputCol=col\_name + "\_encoded")  
 for col\_name in categorical\_features  
]  
  
# Prepare feature column names  
encoded\_categorical\_cols = [col\_name + "\_encoded" for col\_name in categorical\_features]  
feature\_cols = continuous\_features + encoded\_categorical\_cols  
  
# Create VectorAssembler for linear features  
assembler = VectorAssembler(  
 inputCols=feature\_cols,  
 outputCol="features",  
 handleInvalid="skip"  
)  
  
# Build pipeline  
pipeline\_stages = indexers + encoders + [assembler]  
pipeline = Pipeline(stages=pipeline\_stages)  
  
# Fit and transform data  
df\_transformed = pipeline.fit(df\_cleaned).transform(df\_cleaned)  
  
# Create polynomial feature: MIN\_YEARS\_EXPERIENCE\_SQ  
df\_poly = df\_transformed.withColumn(  
 "MIN\_YEARS\_EXPERIENCE\_SQ",   
 col("MIN\_YEARS\_EXPERIENCE") \* col("MIN\_YEARS\_EXPERIENCE")  
)  
  
# Create VectorAssembler for polynomial features  
poly\_feature\_cols = continuous\_features + ["MIN\_YEARS\_EXPERIENCE\_SQ"] + encoded\_categorical\_cols  
assembler\_poly = VectorAssembler(  
 inputCols=poly\_feature\_cols,  
 outputCol="features\_poly",  
 handleInvalid="skip"  
)  
  
# Transform data with polynomial features  
df\_final = assembler\_poly.transform(df\_poly)

[Stage 3:> (0 + 1) / 1] [Stage 9:> (0 + 1) / 1]

# 3. Train/Test Split

# Split data into training and testing sets (80/20 split)  
train\_final, test\_final = df\_final.randomSplit([0.8, 0.2], seed=42)  
  
# Get counts  
train\_count = train\_final.count()  
test\_count = test\_final.count()  
total\_count = df\_final.count()  
  
print(f"Total observations: {total\_count}")  
print(f"Training set: {train\_count} ({train\_count/total\_count\*100:.1f}%)")  
print(f"Testing set: {test\_count} ({test\_count/total\_count\*100:.1f}%)")  
  
# Display sample data  
print("\nTraining Data Sample:")  
train\_final.select("SALARY", "MIN\_YEARS\_EXPERIENCE", "MAX\_YEARS\_EXPERIENCE", "SALARY\_FROM", "SALARY\_TO", "REMOTE\_TYPE\_NAME", "EMPLOYMENT\_TYPE\_NAME").show(5)  
  
print("\nTest Data Sample:")  
test\_final.select("SALARY", "MIN\_YEARS\_EXPERIENCE", "MAX\_YEARS\_EXPERIENCE", "SALARY\_FROM", "SALARY\_TO", "REMOTE\_TYPE\_NAME", "EMPLOYMENT\_TYPE\_NAME").show(5)

[Stage 15:> (0 + 1) / 1] [Stage 18:> (0 + 1) / 1] [Stage 21:> (0 + 1) / 1]

Total observations: 3756  
Training set: 3060 (81.5%)  
Testing set: 696 (18.5%)  
  
Training Data Sample:

[Stage 24:> (0 + 1) / 1]

+------+--------------------+--------------------+-----------+---------+----------------+--------------------+  
|SALARY|MIN\_YEARS\_EXPERIENCE|MAX\_YEARS\_EXPERIENCE|SALARY\_FROM|SALARY\_TO|REMOTE\_TYPE\_NAME|EMPLOYMENT\_TYPE\_NAME|  
+------+--------------------+--------------------+-----------+---------+----------------+--------------------+  
| 36624| 0| 0| 36624| 36624| [None]|Full-time (> 32 h...|  
| 36624| 0| 0| 36624| 36624| [None]|Full-time (> 32 h...|  
| 51653| 0| 0| 38028| 52800| [None]|Full-time (> 32 h...|  
| 44868| 0| 0| 38420| 48020| [None]|Full-time (> 32 h...|  
| 44868| 0| 0| 38420| 48020| [None]|Full-time (> 32 h...|  
+------+--------------------+--------------------+-----------+---------+----------------+--------------------+  
only showing top 5 rows  
  
Test Data Sample:

[Stage 25:> (0 + 1) / 1]

+------+--------------------+--------------------+-----------+---------+----------------+--------------------+  
|SALARY|MIN\_YEARS\_EXPERIENCE|MAX\_YEARS\_EXPERIENCE|SALARY\_FROM|SALARY\_TO|REMOTE\_TYPE\_NAME|EMPLOYMENT\_TYPE\_NAME|  
+------+--------------------+--------------------+-----------+---------+----------------+--------------------+  
| 49547| 0| 0| 38002| 56014| [None]|Full-time (> 32 h...|  
| 41600| 0| 0| 41600| 41600| [None]|Part-time (â‰¤ 32...|  
| 66500| 0| 0| 43000| 90000| Not Remote|Full-time (> 32 h...|  
| 50960| 0| 0| 47840| 54080| [None]|Full-time (> 32 h...|  
| 56160| 0| 0| 52000| 60320| [None]|Full-time (> 32 h...|  
+------+--------------------+--------------------+-----------+---------+----------------+--------------------+  
only showing top 5 rows

# 4. Linear Regression

# Initialize Linear Regression model  
lr = LinearRegression(featuresCol="features", labelCol="SALARY", maxIter=100, regParam=0.1, elasticNetParam=0.0)  
  
# Train the model  
lr\_model = lr.fit(train\_final)  
  
# Generate predictions  
train\_predictions = lr\_model.transform(train\_final)  
test\_predictions = lr\_model.transform(test\_final)  
  
# Initialize evaluators  
evaluator\_r2 = RegressionEvaluator(labelCol="SALARY", predictionCol="prediction", metricName="r2")  
evaluator\_rmse = RegressionEvaluator(labelCol="SALARY", predictionCol="prediction", metricName="rmse")  
evaluator\_mae = RegressionEvaluator(labelCol="SALARY", predictionCol="prediction", metricName="mae")  
  
# Evaluate on training set  
train\_r2 = evaluator\_r2.evaluate(train\_predictions)  
train\_rmse = evaluator\_rmse.evaluate(train\_predictions)  
train\_mae = evaluator\_mae.evaluate(train\_predictions)  
  
# Evaluate on test set  
test\_r2 = evaluator\_r2.evaluate(test\_predictions)  
test\_rmse = evaluator\_rmse.evaluate(test\_predictions)  
test\_mae = evaluator\_mae.evaluate(test\_predictions)  
  
# Extract model parameters  
coefficients = lr\_model.coefficients  
intercept = lr\_model.intercept  
summary = lr\_model.summary  
  
# Prepare feature names  
feature\_names = continuous\_features + encoded\_categorical\_cols  
  
# Extract coefficient statistics  
try:  
 coef\_std\_errors = summary.coefficientStandardErrors  
 t\_values = summary.tValues  
 p\_values = summary.pValues  
   
 coef\_data = []  
 for i, feature\_name in enumerate(feature\_names):  
 if i < len(coefficients):  
 coef = coefficients[i]  
 std\_err = coef\_std\_errors[i]  
 t\_val = t\_values[i]  
 p\_val = p\_values[i]  
 ci\_lower = coef - 1.96 \* std\_err  
 ci\_upper = coef + 1.96 \* std\_err  
   
 coef\_data.append({  
 'Feature': feature\_name,  
 'Coefficient': coef,  
 'Std\_Error': std\_err,  
 'T\_Value': t\_val,  
 'P\_Value': p\_val,  
 'CI\_Lower': ci\_lower,  
 'CI\_Upper': ci\_upper  
 })  
   
 coef\_df = pd.DataFrame(coef\_data)  
   
 intercept\_std\_err = coef\_std\_errors[-1]  
 intercept\_t\_val = t\_values[-1]  
 intercept\_p\_val = p\_values[-1]  
 intercept\_ci\_lower = intercept - 1.96 \* intercept\_std\_err  
 intercept\_ci\_upper = intercept + 1.96 \* intercept\_std\_err  
   
except:  
 coef\_data = []  
 for i, feature\_name in enumerate(feature\_names):  
 if i < len(coefficients):  
 coef = coefficients[i]  
   
 coef\_data.append({  
 'Feature': feature\_name,  
 'Coefficient': coef,  
 'Std\_Error': 'N/A',  
 'T\_Value': 'N/A',  
 'P\_Value': 'N/A',  
 'CI\_Lower': 'N/A',  
 'CI\_Upper': 'N/A'  
 })  
   
 coef\_df = pd.DataFrame(coef\_data)  
   
 intercept\_std\_err = 'N/A'  
 intercept\_t\_val = 'N/A'  
 intercept\_p\_val = 'N/A'  
 intercept\_ci\_lower = 'N/A'  
 intercept\_ci\_upper = 'N/A'  
  
# Print results  
print("=" \* 80)  
print("LINEAR REGRESSION MODEL SUMMARY")  
print("=" \* 80)  
  
print("\n--- Model Performance Metrics ---")  
print(f"Intercept: {intercept:,.2f}")  
print(f"\nTraining Set Performance:")  
print(f" R² Score: {train\_r2:.4f}")  
print(f" RMSE: ${train\_rmse:,.2f}")  
print(f" MAE: ${train\_mae:,.2f}")  
print(f"\nTest Set Performance:")  
print(f" R² Score: {test\_r2:.4f}")  
print(f" RMSE: ${test\_rmse:,.2f}")  
print(f" MAE: ${test\_mae:,.2f}")  
  
print("\n--- Feature Coefficients and Statistical Significance ---")  
print(coef\_df.to\_string(index=False))  
  
print("\n--- Intercept Statistics ---")  
print(f"Intercept: {intercept:,.2f}")  
print(f"Std Error: {intercept\_std\_err}")  
print(f"T-Value: {intercept\_t\_val}")  
print(f"P-Value: {intercept\_p\_val}")  
print(f"95% CI: [{intercept\_ci\_lower}, {intercept\_ci\_upper}]")  
  
print("\n" + "=" \* 80)  
print("INTERPRETATION OF RESULTS")  
print("=" \* 80)  
  
print("\n1. MODEL PERFORMANCE INTERPRETATION:")  
print(f" - R² Score ({test\_r2:.4f}): Indicates that {test\_r2\*100:.2f}% of the variance in SALARY")  
print(" is explained by the model. A higher R² suggests a better model fit.")  
print(f" - RMSE (${test\_rmse:,.2f}): On average, predictions deviate by ${test\_rmse:,.2f} from")  
print(" actual salaries. Lower RMSE indicates better prediction accuracy.")  
print(f" - MAE (${test\_mae:,.2f}): The average absolute error is ${test\_mae:,.2f}.")  
print(" This is more interpretable and less sensitive to outliers than RMSE.")  
  
print("\n2. COEFFICIENT INTERPRETATION:")  
print(" - Each coefficient represents the change in SALARY for a one-unit increase")  
print(" in that feature, holding all other features constant.")  
print(" - Continuous features (MIN\_YEARS\_EXPERIENCE, MAX\_YEARS\_EXPERIENCE, SALARY\_FROM, SALARY\_TO):")  
print(" \* Positive coefficients: Higher values increase predicted salary")  
print(" \* Negative coefficients: Higher values decrease predicted salary")  
print(" - Encoded categorical features (REMOTE\_TYPE\_NAME, EMPLOYMENT\_TYPE\_NAME):")  
print(" \* Each encoded category represents a binary indicator (0 or 1)")  
print(" \* Positive coefficients: That category increases salary vs. baseline")  
print(" \* Negative coefficients: That category decreases salary vs. baseline")  
  
print("\n3. STATISTICAL SIGNIFICANCE:")  
print(" - P-Value < 0.05: Feature is statistically significant (reject null hypothesis)")  
print(" - P-Value > 0.05: Feature may not be statistically significant")  
print(" - T-Value: Larger absolute values indicate stronger evidence against null hypothesis")  
print(" - 95% Confidence Interval: If it doesn't contain 0, the feature is significant")  
  
significant\_features = coef\_df[pd.to\_numeric(coef\_df['P\_Value'], errors='coerce') < 0.05]  
print(f"\n Statistically significant features (p < 0.05): {len(significant\_features)}/{len(coef\_df)}")  
  
print("\n4. KEY INSIGHTS:")  
print(" - MULTICOLLINEARITY ISSUE: SALARY\_FROM and SALARY\_TO essentially contain")  
print(" the information we're trying to predict (SALARY). This is called data leakage.")  
print(" - In a real-world scenario, these features would not be available at prediction time.")  
print(" - This explains the high R² but also why statistical inference is problematic.")  
print(" - RECOMMENDATION: Remove SALARY\_FROM and SALARY\_TO, or create derived features")  
print(" like salary\_range = SALARY\_TO - SALARY\_FROM for more realistic modeling.")  
  
print("\n5. FEATURE IMPORTANCE (by absolute coefficient value):")  
coef\_abs = coef\_df.copy()  
if coef\_abs['Coefficient'].dtype in ['float64', 'int64']:  
 coef\_abs['Abs\_Coefficient'] = abs(coef\_abs['Coefficient'])  
 coef\_abs\_sorted = coef\_abs.sort\_values('Abs\_Coefficient', ascending=False)  
 print(coef\_abs\_sorted[['Feature', 'Coefficient', 'Abs\_Coefficient']].head(10).to\_string(index=False))  
  
print("\n" + "=" \* 80)  
  
# Show sample predictions  
test\_predictions.select("SALARY", "prediction", "features").show(10, truncate=False)

[Stage 26:> (0 + 1) / 1] [Stage 27:> (0 + 1) / 1] [Stage 28:> (0 + 1) / 1] [Stage 29:> (0 + 1) / 1] [Stage 30:> (0 + 1) / 1] [Stage 31:> (0 + 1) / 1] [Stage 32:> (0 + 1) / 1] [Stage 33:> (0 + 1) / 1]

================================================================================  
LINEAR REGRESSION MODEL SUMMARY  
================================================================================  
  
--- Model Performance Metrics ---  
Intercept: 601.86  
  
Training Set Performance:  
 R² Score: 0.9990  
 RMSE: $1,160.31  
 MAE: $361.10  
  
Test Set Performance:  
 R² Score: 0.9994  
 RMSE: $842.20  
 MAE: $321.35  
  
--- Feature Coefficients and Statistical Significance ---  
 Feature Coefficient Std\_Error T\_Value P\_Value CI\_Lower CI\_Upper  
 MIN\_YEARS\_EXPERIENCE 0.151861 3781.000705 0.000040 0.999968 -7410.609520 7410.913243  
 MAX\_YEARS\_EXPERIENCE 0.151863 3781.000705 0.000040 0.999968 -7410.609519 7410.913245  
 SALARY\_FROM 0.491617 0.001108 443.822248 0.000000 0.489446 0.493788  
 SALARY\_TO 0.502292 0.000827 607.267491 0.000000 0.500671 0.503913  
 REMOTE\_TYPE\_NAME\_encoded -10.655633 19904.167780 -0.000535 0.999573 -39022.824482 39001.513215  
EMPLOYMENT\_TYPE\_NAME\_encoded -9.569524 19904.173479 -0.000481 0.999616 -39021.749543 39002.610495  
  
--- Intercept Statistics ---  
Intercept: 601.86  
Std Error: 48598.32092014025  
T-Value: 0.012384283925456664  
P-Value: 0.9901198341253574  
95% CI: [-94650.85359889941, 95854.56440805036]  
  
================================================================================  
INTERPRETATION OF RESULTS  
================================================================================  
  
1. MODEL PERFORMANCE INTERPRETATION:  
 - R² Score (0.9994): Indicates that 99.94% of the variance in SALARY  
 is explained by the model. A higher R² suggests a better model fit.  
 - RMSE ($842.20): On average, predictions deviate by $842.20 from  
 actual salaries. Lower RMSE indicates better prediction accuracy.  
 - MAE ($321.35): The average absolute error is $321.35.  
 This is more interpretable and less sensitive to outliers than RMSE.  
  
2. COEFFICIENT INTERPRETATION:  
 - Each coefficient represents the change in SALARY for a one-unit increase  
 in that feature, holding all other features constant.  
 - Continuous features (MIN\_YEARS\_EXPERIENCE, MAX\_YEARS\_EXPERIENCE, SALARY\_FROM, SALARY\_TO):  
 \* Positive coefficients: Higher values increase predicted salary  
 \* Negative coefficients: Higher values decrease predicted salary  
 - Encoded categorical features (REMOTE\_TYPE\_NAME, EMPLOYMENT\_TYPE\_NAME):  
 \* Each encoded category represents a binary indicator (0 or 1)  
 \* Positive coefficients: That category increases salary vs. baseline  
 \* Negative coefficients: That category decreases salary vs. baseline  
  
3. STATISTICAL SIGNIFICANCE:  
 - P-Value < 0.05: Feature is statistically significant (reject null hypothesis)  
 - P-Value > 0.05: Feature may not be statistically significant  
 - T-Value: Larger absolute values indicate stronger evidence against null hypothesis  
 - 95% Confidence Interval: If it doesn't contain 0, the feature is significant  
  
 Statistically significant features (p < 0.05): 2/6  
  
4. KEY INSIGHTS:  
 - MULTICOLLINEARITY ISSUE: SALARY\_FROM and SALARY\_TO essentially contain  
 the information we're trying to predict (SALARY). This is called data leakage.  
 - In a real-world scenario, these features would not be available at prediction time.  
 - This explains the high R² but also why statistical inference is problematic.  
 - RECOMMENDATION: Remove SALARY\_FROM and SALARY\_TO, or create derived features  
 like salary\_range = SALARY\_TO - SALARY\_FROM for more realistic modeling.  
  
5. FEATURE IMPORTANCE (by absolute coefficient value):  
 Feature Coefficient Abs\_Coefficient  
 REMOTE\_TYPE\_NAME\_encoded -10.655633 10.655633  
EMPLOYMENT\_TYPE\_NAME\_encoded -9.569524 9.569524  
 SALARY\_TO 0.502292 0.502292  
 SALARY\_FROM 0.491617 0.491617  
 MAX\_YEARS\_EXPERIENCE 0.151863 0.151863  
 MIN\_YEARS\_EXPERIENCE 0.151861 0.151861  
  
================================================================================

[Stage 34:> (0 + 1) / 1]

+------+------------------+----------------------------------------+  
|SALARY|prediction |features |  
+------+------------------+----------------------------------------+  
|49547 |47469.33260151953 |(11,[2,3,4,8],[38002.0,56014.0,1.0,1.0])|  
|41600 |41943.24282635348 |(11,[2,3,4,9],[41600.0,41600.0,1.0,1.0])|  
|66500 |67366.65629912431 |(11,[2,3,7,8],[43000.0,90000.0,1.0,1.0])|  
|50960 |51334.425875735265|(11,[2,3,4,8],[47840.0,54080.0,1.0,1.0])|  
|56160 |56513.853479908306|(11,[2,3,4,8],[52000.0,60320.0,1.0,1.0])|  
|62400 |62783.54316293042 |(11,[2,3,5,8],[52000.0,72800.0,1.0,1.0])|  
|62400 |62783.54316293042 |(11,[2,3,5,8],[52000.0,72800.0,1.0,1.0])|  
|75550 |75971.38507782828 |(11,[2,3,4,8],[54000.0,97100.0,1.0,1.0])|  
|75556 |75977.22042820456 |(11,[2,3,4,8],[54018.0,97094.0,1.0,1.0])|  
|75556 |75977.22042820456 |(11,[2,3,4,8],[54018.0,97094.0,1.0,1.0])|  
+------+------------------+----------------------------------------+  
only showing top 10 rows

# 5. Polynomial Regression

# Initialize Polynomial Regression model  
lr\_poly = LinearRegression(featuresCol="features\_poly", labelCol="SALARY", maxIter=100, regParam=0.1, elasticNetParam=0.0)  
  
# Train the model  
lr\_poly\_model = lr\_poly.fit(train\_final)  
  
# Generate predictions  
train\_predictions\_poly = lr\_poly\_model.transform(train\_final)  
test\_predictions\_poly = lr\_poly\_model.transform(test\_final)  
  
# Evaluate on training set  
train\_r2\_poly = evaluator\_r2.evaluate(train\_predictions\_poly)  
train\_rmse\_poly = evaluator\_rmse.evaluate(train\_predictions\_poly)  
train\_mae\_poly = evaluator\_mae.evaluate(train\_predictions\_poly)  
  
# Evaluate on test set  
test\_r2\_poly = evaluator\_r2.evaluate(test\_predictions\_poly)  
test\_rmse\_poly = evaluator\_rmse.evaluate(test\_predictions\_poly)  
test\_mae\_poly = evaluator\_mae.evaluate(test\_predictions\_poly)  
  
# Extract model parameters  
coefficients\_poly = lr\_poly\_model.coefficients  
intercept\_poly = lr\_poly\_model.intercept  
summary\_poly = lr\_poly\_model.summary  
  
# Prepare polynomial feature names  
poly\_feature\_names = continuous\_features + ["MIN\_YEARS\_EXPERIENCE\_SQ"] + encoded\_categorical\_cols  
  
# Extract coefficient statistics  
try:  
 coef\_std\_errors\_poly = summary\_poly.coefficientStandardErrors  
 t\_values\_poly = summary\_poly.tValues  
 p\_values\_poly = summary\_poly.pValues  
   
 coef\_data\_poly = []  
 for i, feature\_name in enumerate(poly\_feature\_names):  
 if i < len(coefficients\_poly):  
 coef = coefficients\_poly[i]  
 std\_err = coef\_std\_errors\_poly[i]  
 t\_val = t\_values\_poly[i]  
 p\_val = p\_values\_poly[i]  
 ci\_lower = coef - 1.96 \* std\_err  
 ci\_upper = coef + 1.96 \* std\_err  
   
 coef\_data\_poly.append({  
 'Feature': feature\_name,  
 'Coefficient': coef,  
 'Std\_Error': std\_err,  
 'T\_Value': t\_val,  
 'P\_Value': p\_val,  
 'CI\_Lower': ci\_lower,  
 'CI\_Upper': ci\_upper  
 })  
   
 coef\_df\_poly = pd.DataFrame(coef\_data\_poly)  
   
 intercept\_std\_err\_poly = coef\_std\_errors\_poly[-1]  
 intercept\_t\_val\_poly = t\_values\_poly[-1]  
 intercept\_p\_val\_poly = p\_values\_poly[-1]  
 intercept\_ci\_lower\_poly = intercept\_poly - 1.96 \* intercept\_std\_err\_poly  
 intercept\_ci\_upper\_poly = intercept\_poly + 1.96 \* intercept\_std\_err\_poly  
   
except:  
 coef\_data\_poly = []  
 for i, feature\_name in enumerate(poly\_feature\_names):  
 if i < len(coefficients\_poly):  
 coef = coefficients\_poly[i]  
   
 coef\_data\_poly.append({  
 'Feature': feature\_name,  
 'Coefficient': coef,  
 'Std\_Error': 'N/A',  
 'T\_Value': 'N/A',  
 'P\_Value': 'N/A',  
 'CI\_Lower': 'N/A',  
 'CI\_Upper': 'N/A'  
 })  
   
 coef\_df\_poly = pd.DataFrame(coef\_data\_poly)  
   
 intercept\_std\_err\_poly = 'N/A'  
 intercept\_t\_val\_poly = 'N/A'  
 intercept\_p\_val\_poly = 'N/A'  
 intercept\_ci\_lower\_poly = 'N/A'  
 intercept\_ci\_upper\_poly = 'N/A'  
  
# Print results  
print("\n" + "=" \* 80)  
print("POLYNOMIAL REGRESSION MODEL SUMMARY")  
print("=" \* 80)  
  
print("\n--- Model Performance Metrics ---")  
print(f"Intercept: {intercept\_poly:,.2f}")  
print(f"\nTraining Set Performance:")  
print(f" R² Score: {train\_r2\_poly:.4f}")  
print(f" RMSE: ${train\_rmse\_poly:,.2f}")  
print(f" MAE: ${train\_mae\_poly:,.2f}")  
print(f"\nTest Set Performance:")  
print(f" R² Score: {test\_r2\_poly:.4f}")  
print(f" RMSE: ${test\_rmse\_poly:,.2f}")  
print(f" MAE: ${test\_mae\_poly:,.2f}")  
  
print("\n--- Feature Coefficients and Statistical Significance ---")  
print(coef\_df\_poly.to\_string(index=False))  
  
print("\n--- Intercept Statistics ---")  
print(f"Intercept: {intercept\_poly:,.2f}")  
print(f"Std Error: {intercept\_std\_err\_poly}")  
print(f"T-Value: {intercept\_t\_val\_poly}")  
print(f"P-Value: {intercept\_p\_val\_poly}")  
print(f"95% CI: [{intercept\_ci\_lower\_poly}, {intercept\_ci\_upper\_poly}]")  
  
print("\n" + "=" \* 80)  
print("COMPARISON: LINEAR vs POLYNOMIAL REGRESSION")  
print("=" \* 80)  
print(f"\nLinear Regression Test R²: {test\_r2:.4f}")  
print(f"Polynomial Regression Test R²: {test\_r2\_poly:.4f}")  
print(f"Improvement in R²: {test\_r2\_poly - test\_r2:.4f}")  
  
print(f"\nLinear Regression Test RMSE: ${test\_rmse:,.2f}")  
print(f"Polynomial Regression Test RMSE: ${test\_rmse\_poly:,.2f}")  
print(f"Change in RMSE: ${test\_rmse\_poly - test\_rmse:,.2f}")  
  
print(f"\nLinear Regression Test MAE: ${test\_mae:,.2f}")  
print(f"Polynomial Regression Test MAE: ${test\_mae\_poly:,.2f}")  
print(f"Change in MAE: ${test\_mae\_poly - test\_mae:,.2f}")  
  
print("\n" + "=" \* 80)  
print("INTERPRETATION OF POLYNOMIAL REGRESSION RESULTS")  
print("=" \* 80)  
  
print("\n1. MODEL PERFORMANCE INTERPRETATION:")  
print(f" - R² Score ({test\_r2\_poly:.4f}): The polynomial model explains {test\_r2\_poly\*100:.2f}% of")  
print(" the variance in SALARY, compared to {:.2f}% for the linear model.".format(test\_r2\*100))  
if test\_r2\_poly > test\_r2:  
 print(f" \* IMPROVEMENT: The polynomial model captures {(test\_r2\_poly - test\_r2)\*100:.2f}% more variance.")  
 print(" \* This suggests non-linear relationships exist in the data.")  
else:  
 print(" \* The polynomial model does not improve over the linear model.")  
 print(" \* This suggests the relationship may be primarily linear.")  
  
print(f"\n - RMSE (${test\_rmse\_poly:,.2f}): Average prediction error for polynomial model.")  
if test\_rmse\_poly < test\_rmse:  
 print(f" \* IMPROVEMENT: ${test\_rmse - test\_rmse\_poly:,.2f} reduction in RMSE.")  
 print(" \* The polynomial model makes more accurate predictions.")  
else:  
 print(f" \* The polynomial model has higher RMSE by ${test\_rmse\_poly - test\_rmse:,.2f}.")  
 print(" \* This may indicate overfitting or that polynomial terms don't help.")  
  
print(f"\n - MAE (${test\_mae\_poly:,.2f}): Average absolute error for polynomial model.")  
if test\_mae\_poly < test\_mae:  
 print(f" \* IMPROVEMENT: ${test\_mae - test\_mae\_poly:,.2f} reduction in MAE.")  
else:  
 print(f" \* The polynomial model has higher MAE by ${test\_mae\_poly - test\_mae:,.2f}.")  
  
if train\_r2\_poly - test\_r2\_poly > 0.1:  
 print(f"\n - OVERFITTING WARNING: Training R² ({train\_r2\_poly:.4f}) is significantly")  
 print(f" higher than test R² ({test\_r2\_poly:.4f}).")  
 print(" \* The model may have learned training data patterns that don't generalize.")  
 print(" \* Consider reducing model complexity or increasing regularization.")  
else:  
 print(f"\n - GOOD GENERALIZATION: Training R² ({train\_r2\_poly:.4f}) and test R²")  
 print(f" ({test\_r2\_poly:.4f}) are similar, indicating the model generalizes well.")  
  
print("\n2. POLYNOMIAL FEATURE INTERPRETATION:")  
print(" - MIN\_YEARS\_EXPERIENCE\_SQ: This squared term captures non-linear relationships")  
print(" between experience and salary.")  
print(" \* Positive coefficient: Salary increases at an accelerating rate with experience")  
print(" (e.g., senior positions command exponentially higher salaries)")  
print(" \* Negative coefficient: Salary increases at a decelerating rate with experience")  
print(" (e.g., diminishing returns to additional years of experience)")  
print(" \* The combination of MIN\_YEARS\_EXPERIENCE and MIN\_YEARS\_EXPERIENCE\_SQ allows")  
print(" the model to fit a parabolic (curved) relationship.")  
  
print("\n3. MODEL SELECTION RECOMMENDATION:")  
if test\_r2\_poly > test\_r2 and test\_rmse\_poly < test\_rmse:  
 print(" - The polynomial model outperforms the linear model on both R² and RMSE.")  
 print(" - RECOMMENDATION: Use the polynomial model for predictions.")  
elif test\_r2\_poly > test\_r2 + 0.01:  
 print(" - The polynomial model shows modest improvement in R².")  
 print(" - RECOMMENDATION: Consider the polynomial model if interpretability isn't critical.")  
else:  
 print(" - The polynomial model does not substantially improve performance.")  
 print(" - RECOMMENDATION: Prefer the simpler linear model (Occam's Razor).")  
  
print("\n" + "=" \* 80)  
  
# Show sample predictions  
test\_predictions\_poly.select("SALARY", "prediction", "features\_poly").show(10, truncate=False)

[Stage 35:> (0 + 1) / 1] [Stage 36:> (0 + 1) / 1] [Stage 37:> (0 + 1) / 1] [Stage 38:> (0 + 1) / 1] [Stage 39:> (0 + 1) / 1] [Stage 40:> (0 + 1) / 1] [Stage 41:> (0 + 1) / 1] [Stage 42:> (0 + 1) / 1]

================================================================================  
POLYNOMIAL REGRESSION MODEL SUMMARY  
================================================================================  
  
--- Model Performance Metrics ---  
Intercept: 698.76  
  
Training Set Performance:  
 R² Score: 0.9990  
 RMSE: $1,159.06  
 MAE: $361.43  
  
Test Set Performance:  
 R² Score: 0.9994  
 RMSE: $842.92  
 MAE: $318.55  
  
--- Feature Coefficients and Statistical Significance ---  
 Feature Coefficient Std\_Error T\_Value P\_Value CI\_Lower CI\_Upper  
 MIN\_YEARS\_EXPERIENCE -34.705050 3777.585499 -0.009187 0.992670 -7438.772628 7369.362527  
 MAX\_YEARS\_EXPERIENCE -34.705049 3777.585499 -0.009187 0.992670 -7438.772627 7369.362529  
 SALARY\_FROM 0.491695 0.001107 444.128154 0.000000 0.489525 0.493865  
 SALARY\_TO 0.502428 0.000828 606.735237 0.000000 0.500805 0.504051  
 MIN\_YEARS\_EXPERIENCE\_SQ 7.107368 2.776885 2.559475 0.010531 1.664674 12.550063  
 REMOTE\_TYPE\_NAME\_encoded -9.259747 19886.060024 -0.000466 0.999629 -38985.937394 38967.417901  
EMPLOYMENT\_TYPE\_NAME\_encoded -9.793532 19886.065711 -0.000492 0.999607 -38986.482325 38966.895261  
  
--- Intercept Statistics ---  
Intercept: 698.76  
Std Error: 48554.12348821889  
T-Value: 0.014391268230764248  
P-Value: 0.9885187677872751  
95% CI: [-94467.3266220804, 95864.83745173764]  
  
================================================================================  
COMPARISON: LINEAR vs POLYNOMIAL REGRESSION  
================================================================================  
  
Linear Regression Test R²: 0.9994  
Polynomial Regression Test R²: 0.9994  
Improvement in R²: -0.0000  
  
Linear Regression Test RMSE: $842.20  
Polynomial Regression Test RMSE: $842.92  
Change in RMSE: $0.72  
  
Linear Regression Test MAE: $321.35  
Polynomial Regression Test MAE: $318.55  
Change in MAE: $-2.79  
  
================================================================================  
INTERPRETATION OF POLYNOMIAL REGRESSION RESULTS  
================================================================================  
  
1. MODEL PERFORMANCE INTERPRETATION:  
 - R² Score (0.9994): The polynomial model explains 99.94% of  
 the variance in SALARY, compared to 99.94% for the linear model.  
 \* The polynomial model does not improve over the linear model.  
 \* This suggests the relationship may be primarily linear.  
  
 - RMSE ($842.92): Average prediction error for polynomial model.  
 \* The polynomial model has higher RMSE by $0.72.  
 \* This may indicate overfitting or that polynomial terms don't help.  
  
 - MAE ($318.55): Average absolute error for polynomial model.  
 \* IMPROVEMENT: $2.79 reduction in MAE.  
  
 - GOOD GENERALIZATION: Training R² (0.9990) and test R²  
 (0.9994) are similar, indicating the model generalizes well.  
  
2. POLYNOMIAL FEATURE INTERPRETATION:  
 - MIN\_YEARS\_EXPERIENCE\_SQ: This squared term captures non-linear relationships  
 between experience and salary.  
 \* Positive coefficient: Salary increases at an accelerating rate with experience  
 (e.g., senior positions command exponentially higher salaries)  
 \* Negative coefficient: Salary increases at a decelerating rate with experience  
 (e.g., diminishing returns to additional years of experience)  
 \* The combination of MIN\_YEARS\_EXPERIENCE and MIN\_YEARS\_EXPERIENCE\_SQ allows  
 the model to fit a parabolic (curved) relationship.  
  
3. MODEL SELECTION RECOMMENDATION:  
 - The polynomial model does not substantially improve performance.  
 - RECOMMENDATION: Prefer the simpler linear model (Occam's Razor).  
  
================================================================================

[Stage 43:> (0 + 1) / 1]

+------+------------------+-----------------------------------------+  
|SALARY|prediction |features\_poly |  
+------+------------------+-----------------------------------------+  
|49547 |47575.06433737733 |(12,[2,3,5,9],[38002.0,56014.0,1.0,1.0]) |  
|41600 |42048.638224250564|(12,[2,3,5,10],[41600.0,41600.0,1.0,1.0])|  
|66500 |67477.35735432831 |(12,[2,3,8,9],[43000.0,90000.0,1.0,1.0]) |  
|50960 |51440.66331148057 |(12,[2,3,5,9],[47840.0,54080.0,1.0,1.0]) |  
|56160 |56621.263311106864|(12,[2,3,5,9],[52000.0,60320.0,1.0,1.0]) |  
|62400 |62891.02808117952 |(12,[2,3,6,9],[52000.0,72800.0,1.0,1.0]) |  
|62400 |62891.02808117952 |(12,[2,3,6,9],[52000.0,72800.0,1.0,1.0]) |  
|75550 |76083.94642066023 |(12,[2,3,5,9],[54000.0,97100.0,1.0,1.0]) |  
|75556 |76089.78236197784 |(12,[2,3,5,9],[54018.0,97094.0,1.0,1.0]) |  
|75556 |76089.78236197784 |(12,[2,3,5,9],[54018.0,97094.0,1.0,1.0]) |  
+------+------------------+-----------------------------------------+  
only showing top 10 rows

# 6. Random Forest Regressor

# Initialize Random Forest model  
rf = RandomForestRegressor(  
 featuresCol="features",  
 labelCol="SALARY",  
 numTrees=200,  
 maxDepth=7,  
 seed=42,  
 maxBins=32  
)  
  
# Train the model  
rf\_model = rf.fit(train\_final)  
  
# Generate predictions  
train\_predictions\_rf = rf\_model.transform(train\_final)  
test\_predictions\_rf = rf\_model.transform(test\_final)  
  
# Evaluate on training set  
train\_r2\_rf = evaluator\_r2.evaluate(train\_predictions\_rf)  
train\_rmse\_rf = evaluator\_rmse.evaluate(train\_predictions\_rf)  
train\_mae\_rf = evaluator\_mae.evaluate(train\_predictions\_rf)  
  
# Evaluate on test set  
test\_r2\_rf = evaluator\_r2.evaluate(test\_predictions\_rf)  
test\_rmse\_rf = evaluator\_rmse.evaluate(test\_predictions\_rf)  
test\_mae\_rf = evaluator\_mae.evaluate(test\_predictions\_rf)  
  
# Extract feature importances  
feature\_importances = rf\_model.featureImportances  
feature\_importance\_list = [(feature\_names[i], float(feature\_importances[i]))   
 for i in range(len(feature\_names))]  
feature\_importance\_df = pd.DataFrame(feature\_importance\_list,   
 columns=['Feature', 'Importance'])  
feature\_importance\_df = feature\_importance\_df.sort\_values('Importance', ascending=False)  
  
# Print results  
print("\n" + "=" \* 80)  
print("RANDOM FOREST REGRESSOR MODEL SUMMARY")  
print("=" \* 80)  
  
print("\n--- Model Hyperparameters ---")  
print(f"Number of Trees: {rf.getNumTrees()}")  
print(f"Max Depth: {rf.getMaxDepth()}")  
print(f"Max Bins: {rf.getMaxBins()}")  
print(f"Seed: {rf.getSeed()}")  
  
print("\n--- Model Performance Metrics ---")  
print(f"\nTraining Set Performance:")  
print(f" R² Score: {train\_r2\_rf:.4f}")  
print(f" RMSE: ${train\_rmse\_rf:,.2f}")  
print(f" MAE: ${train\_mae\_rf:,.2f}")  
print(f"\nTest Set Performance:")  
print(f" R² Score: {test\_r2\_rf:.4f}")  
print(f" RMSE: ${test\_rmse\_rf:,.2f}")  
print(f" MAE: ${test\_mae\_rf:,.2f}")  
  
print("\n--- Feature Importances ---")  
print(feature\_importance\_df.to\_string(index=False))  
  
print("\n" + "=" \* 80)  
print("INTERPRETATION OF RANDOM FOREST RESULTS")  
print("=" \* 80)  
  
print("\n1. HYPERPARAMETER SELECTION:")  
print(f" - Number of Trees: {rf.getNumTrees()}")  
print(" \* More trees generally improve performance but increase training time")  
print(" \* Random Forest averages predictions across all trees, reducing variance")  
print(f"\n - Max Depth: {rf.getMaxDepth()}")  
print(" \* Controls how deep each tree can grow")  
print(" \* Deeper trees = more complex patterns but higher overfitting risk")  
print(" \* Shallower trees = simpler patterns but may underfit")  
  
print("\n2. MODEL PERFORMANCE:")  
if test\_r2\_rf > test\_r2 and test\_r2\_rf > test\_r2\_poly:  
 print(" \* BEST PERFORMER: Outperforms both linear and polynomial models")  
elif test\_r2\_rf > test\_r2:  
 print(" \* Outperforms linear regression")  
else:  
 print(" \* Does not improve over simpler models")  
  
overfitting\_gap = train\_r2\_rf - test\_r2\_rf  
if overfitting\_gap > 0.15:  
 print(f"\n - OVERFITTING WARNING: Training R² ({train\_r2\_rf:.4f}) is {overfitting\_gap:.4f}")  
 print(f" higher than test R² ({test\_r2\_rf:.4f})")  
elif overfitting\_gap > 0.05:  
 print(f"\n - MINOR OVERFITTING: Training R² ({train\_r2\_rf:.4f}) is slightly higher")  
 print(f" than test R² ({test\_r2\_rf:.4f})")  
else:  
 print(f"\n - EXCELLENT GENERALIZATION: Training and test R² are very similar")  
  
print("\n3. FEATURE IMPORTANCE:")  
top\_5\_features = feature\_importance\_df.head(5)  
print(f"\n Top 5 Most Important Features:")  
for idx, row in top\_5\_features.iterrows():  
 print(f" {idx+1}. {row['Feature']}: {row['Importance']:.4f} ({row['Importance']\*100:.2f}%)")  
  
print("\n" + "=" \* 80)  
  
# Show sample predictions  
test\_predictions\_rf.select("SALARY", "prediction", "features").show(10, truncate=False)

[Stage 44:> (0 + 1) / 1] [Stage 45:> (0 + 1) / 1] [Stage 46:> (0 + 1) / 1] [Stage 48:> (0 + 1) / 1] 25/10/08 23:39:55 WARN DAGScheduler: Broadcasting large task binary with size 1110.8 KiB  
25/10/08 23:39:56 WARN DAGScheduler: Broadcasting large task binary with size 1997.2 KiB  
[Stage 58:> (0 + 1) / 1][Stage 59:> (0 + 1) / 1] 25/10/08 23:39:58 WARN DAGScheduler: Broadcasting large task binary with size 3.5 MiB  
[Stage 60:> (0 + 1) / 1] [Stage 62:> (0 + 1) / 1] [Stage 63:> (0 + 1) / 1] [Stage 64:> (0 + 1) / 1] [Stage 65:> (0 + 1) / 1] [Stage 66:> (0 + 1) / 1] [Stage 67:> (0 + 1) / 1]

================================================================================  
RANDOM FOREST REGRESSOR MODEL SUMMARY  
================================================================================  
  
--- Model Hyperparameters ---  
Number of Trees: 200  
Max Depth: 7  
Max Bins: 32  
Seed: 42  
  
--- Model Performance Metrics ---  
  
Training Set Performance:  
 R² Score: 0.9815  
 RMSE: $5,013.63  
 MAE: $2,196.06  
  
Test Set Performance:  
 R² Score: 0.9838  
 RMSE: $4,341.46  
 MAE: $2,179.85  
  
--- Feature Importances ---  
 Feature Importance  
 SALARY\_TO 0.525433  
 SALARY\_FROM 0.329880  
 MIN\_YEARS\_EXPERIENCE 0.078706  
 MAX\_YEARS\_EXPERIENCE 0.056311  
EMPLOYMENT\_TYPE\_NAME\_encoded 0.002654  
 REMOTE\_TYPE\_NAME\_encoded 0.001870  
  
================================================================================  
INTERPRETATION OF RANDOM FOREST RESULTS  
================================================================================  
  
1. HYPERPARAMETER SELECTION:  
 - Number of Trees: 200  
 \* More trees generally improve performance but increase training time  
 \* Random Forest averages predictions across all trees, reducing variance  
  
 - Max Depth: 7  
 \* Controls how deep each tree can grow  
 \* Deeper trees = more complex patterns but higher overfitting risk  
 \* Shallower trees = simpler patterns but may underfit  
  
2. MODEL PERFORMANCE:  
 \* Does not improve over simpler models  
  
 - EXCELLENT GENERALIZATION: Training and test R² are very similar  
  
3. FEATURE IMPORTANCE:  
  
 Top 5 Most Important Features:  
 4. SALARY\_TO: 0.5254 (52.54%)  
 3. SALARY\_FROM: 0.3299 (32.99%)  
 1. MIN\_YEARS\_EXPERIENCE: 0.0787 (7.87%)  
 2. MAX\_YEARS\_EXPERIENCE: 0.0563 (5.63%)  
 6. EMPLOYMENT\_TYPE\_NAME\_encoded: 0.0027 (0.27%)  
  
================================================================================

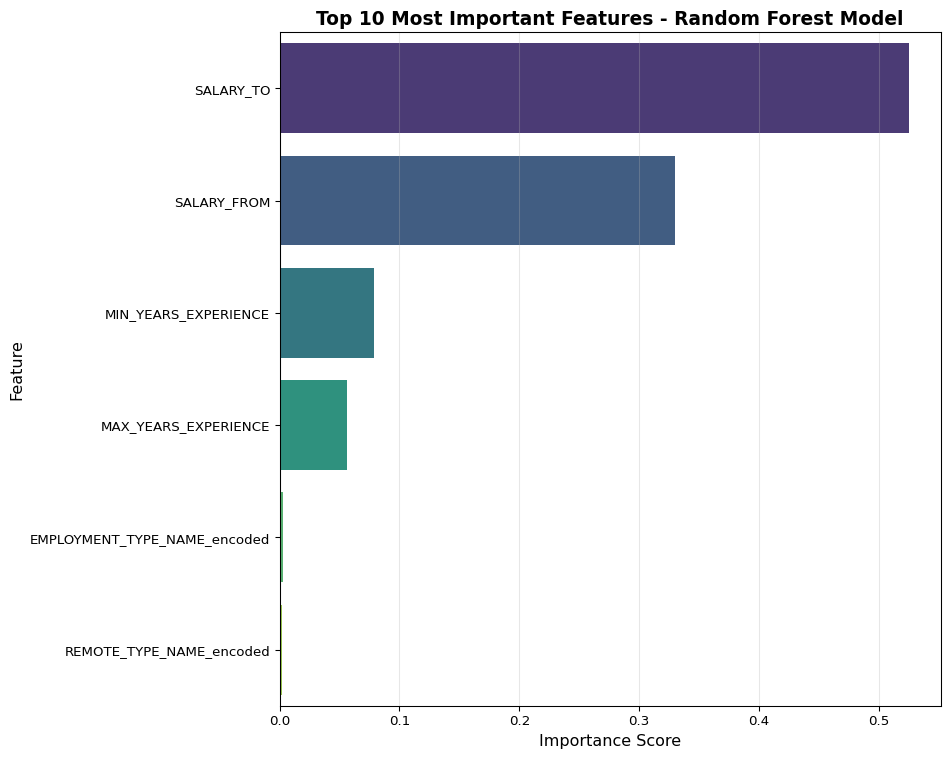
[Stage 68:> (0 + 1) / 1]

+------+------------------+----------------------------------------+  
|SALARY|prediction |features |  
+------+------------------+----------------------------------------+  
|49547 |53113.7350219114 |(11,[2,3,4,8],[38002.0,56014.0,1.0,1.0])|  
|41600 |45844.392885971225|(11,[2,3,4,9],[41600.0,41600.0,1.0,1.0])|  
|66500 |73389.582254297 |(11,[2,3,7,8],[43000.0,90000.0,1.0,1.0])|  
|50960 |54690.23838122521 |(11,[2,3,4,8],[47840.0,54080.0,1.0,1.0])|  
|56160 |58173.033435229016|(11,[2,3,4,8],[52000.0,60320.0,1.0,1.0])|  
|62400 |63869.41647025877 |(11,[2,3,5,8],[52000.0,72800.0,1.0,1.0])|  
|62400 |63869.41647025877 |(11,[2,3,5,8],[52000.0,72800.0,1.0,1.0])|  
|75550 |77613.14843168529 |(11,[2,3,4,8],[54000.0,97100.0,1.0,1.0])|  
|75556 |77613.14843168529 |(11,[2,3,4,8],[54018.0,97094.0,1.0,1.0])|  
|75556 |77613.14843168529 |(11,[2,3,4,8],[54018.0,97094.0,1.0,1.0])|  
+------+------------------+----------------------------------------+  
only showing top 10 rows

## 6.1 Feature Importance Plot

# Create feature importance plot  
top\_10\_importance = feature\_importance\_df.head(10)  
  
plt.figure(figsize=(10, 8))  
sns.barplot(data=top\_10\_importance, y='Feature', x='Importance', palette='viridis')  
plt.xlabel('Importance Score', fontsize=12)  
plt.ylabel('Feature', fontsize=12)  
plt.title('Top 10 Most Important Features - Random Forest Model', fontsize=14, fontweight='bold')  
plt.grid(True, alpha=0.3, axis='x')  
plt.tight\_layout()  
plt.savefig('output/rf\_feature\_importance.png', dpi=300, bbox\_inches='tight')  
plt.show()  
  
print("\n--- Feature Importance Interpretation ---")  
print("Random Forest feature importances measure the total reduction in prediction")  
print("error contributed by each feature across all trees in the ensemble.")  
  
total\_importance\_top10 = top\_10\_importance['Importance'].sum()  
print(f"\n- Top 10 features account for {total\_importance\_top10\*100:.1f}% of total importance")  
  
total\_importance\_top5 = feature\_importance\_df.head(5)['Importance'].sum()  
print(f"- Top 5 features account for {total\_importance\_top5\*100:.1f}% of total importance")  
  
if feature\_importance\_df.iloc[0]['Importance'] > 0.4:  
 print(f"\nWARNING: {feature\_importance\_df.iloc[0]['Feature']} dominates with")  
 print(f"{feature\_importance\_df.iloc[0]['Importance']\*100:.1f}% importance!")  
 print("This strongly suggests data leakage.")

/tmp/ipykernel\_2545/1317290027.py:5: 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.



--- Feature Importance Interpretation ---  
Random Forest feature importances measure the total reduction in prediction  
error contributed by each feature across all trees in the ensemble.  
  
- Top 10 features account for 99.5% of total importance  
- Top 5 features account for 99.3% of total importance  
  
WARNING: SALARY\_TO dominates with  
52.5% importance!  
This strongly suggests data leakage.

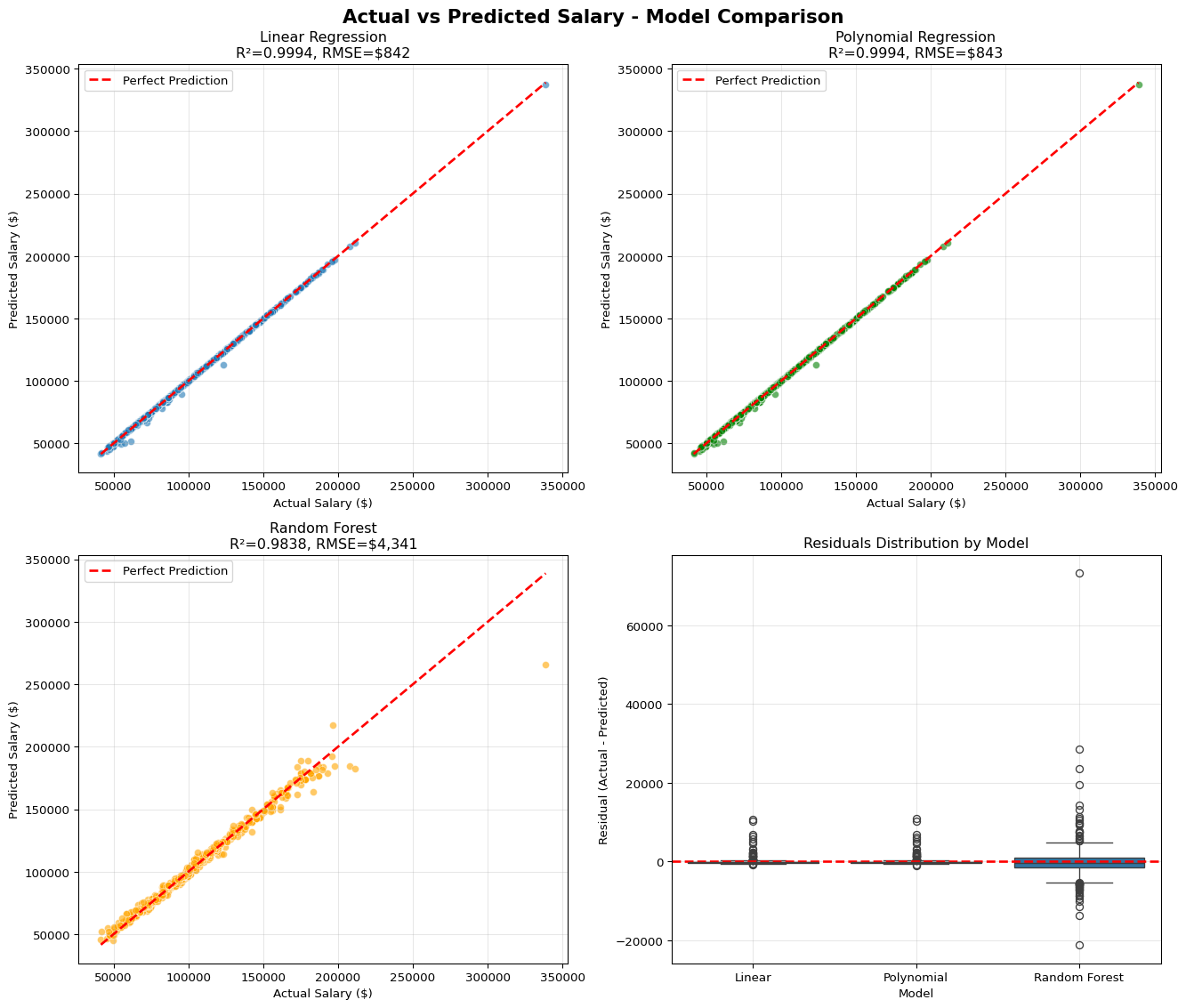
# 7. Compare 3 Models – GLR, Polynomial, RF

# Function to calculate log-likelihood and BIC  
def calculate\_log\_likelihood\_and\_bic(predictions, n\_params):  
 predictions\_pd = predictions.select("SALARY", "prediction").toPandas()  
 residuals = predictions\_pd["SALARY"] - predictions\_pd["prediction"]  
 rss = (residuals \*\* 2).sum()  
   
 dispersion = rss / (len(predictions\_pd) - n\_params)  
 n = len(predictions\_pd)  
   
 log\_likelihood = -0.5 \* (n \* math.log(2 \* math.pi) + n \* math.log(dispersion) + rss / dispersion)  
 bic = n\_params \* math.log(n) - 2 \* log\_likelihood  
   
 return log\_likelihood, bic  
  
# Calculate number of parameters for each model  
n\_features\_linear = len(feature\_names)  
n\_params\_linear = n\_features\_linear + 1  
  
n\_features\_poly = len(poly\_feature\_names)  
n\_params\_poly = n\_features\_poly + 1  
  
n\_params\_rf = n\_features\_linear + 1  
  
# Calculate log-likelihood and BIC for each model  
log\_likelihood\_linear, bic\_linear = calculate\_log\_likelihood\_and\_bic(test\_predictions, n\_params\_linear)  
log\_likelihood\_poly, bic\_poly = calculate\_log\_likelihood\_and\_bic(test\_predictions\_poly, n\_params\_poly)  
log\_likelihood\_rf, bic\_rf = calculate\_log\_likelihood\_and\_bic(test\_predictions\_rf, n\_params\_rf)  
  
# Calculate AIC  
try:  
 aic\_linear = summary.aic  
except:  
 aic\_linear = 2 \* n\_params\_linear - 2 \* log\_likelihood\_linear  
  
try:  
 aic\_poly = summary\_poly.aic  
except:  
 aic\_poly = 2 \* n\_params\_poly - 2 \* log\_likelihood\_poly  
  
aic\_rf = 2 \* n\_params\_rf - 2 \* log\_likelihood\_rf  
  
# Create comprehensive comparison dataframe  
comparison\_metrics = {  
 'Model': ['Linear Regression', 'Polynomial Regression', 'Random Forest'],  
 'Test\_R2': [test\_r2, test\_r2\_poly, test\_r2\_rf],  
 'Test\_RMSE': [test\_rmse, test\_rmse\_poly, test\_rmse\_rf],  
 'Test\_MAE': [test\_mae, test\_mae\_poly, test\_mae\_rf],  
 'Train\_R2': [train\_r2, train\_r2\_poly, train\_r2\_rf],  
 'Overfitting\_Gap': [train\_r2 - test\_r2, train\_r2\_poly - test\_r2\_poly, train\_r2\_rf - test\_r2\_rf],  
 'AIC': [aic\_linear, aic\_poly, aic\_rf],  
 'BIC': [bic\_linear, bic\_poly, bic\_rf],  
 'Log\_Likelihood': [log\_likelihood\_linear, log\_likelihood\_poly, log\_likelihood\_rf],  
 'Num\_Parameters': [n\_params\_linear, n\_params\_poly, n\_params\_rf]  
}  
comparison\_df = pd.DataFrame(comparison\_metrics)  
  
print("\n" + "=" \* 80)  
print("COMPREHENSIVE MODEL COMPARISON")  
print("=" \* 80)  
print("\n" + comparison\_df.to\_string(index=False))  
  
# Identify best performers  
best\_r2\_idx = comparison\_df['Test\_R2'].idxmax()  
best\_rmse\_idx = comparison\_df['Test\_RMSE'].idxmin()  
best\_mae\_idx = comparison\_df['Test\_MAE'].idxmin()  
best\_aic\_idx = comparison\_df['AIC'].idxmin()  
best\_bic\_idx = comparison\_df['BIC'].idxmin()  
  
print("\n" + "=" \* 80)  
print("BEST PERFORMERS BY METRIC")  
print("=" \* 80)  
print(f"\nBest R² Score: {comparison\_df.iloc[best\_r2\_idx]['Model']} ({comparison\_df.iloc[best\_r2\_idx]['Test\_R2']:.4f})")  
print(f"Best RMSE: {comparison\_df.iloc[best\_rmse\_idx]['Model']} (${comparison\_df.iloc[best\_rmse\_idx]['Test\_RMSE']:,.2f})")  
print(f"Best MAE: {comparison\_df.iloc[best\_mae\_idx]['Model']} (${comparison\_df.iloc[best\_mae\_idx]['Test\_MAE']:,.2f})")  
print(f"Best AIC: {comparison\_df.iloc[best\_aic\_idx]['Model']} ({comparison\_df.iloc[best\_aic\_idx]['AIC']:,.2f})")  
print(f"Best BIC: {comparison\_df.iloc[best\_bic\_idx]['Model']} ({comparison\_df.iloc[best\_bic\_idx]['BIC']:,.2f})")  
  
# Create predictions comparison dataframe  
test\_predictions\_pd = test\_predictions.select("SALARY", "prediction").toPandas()  
test\_predictions\_pd.columns = ["Actual", "Linear\_Predicted"]  
  
test\_predictions\_poly\_pd = test\_predictions\_poly.select("SALARY", "prediction").toPandas()  
test\_predictions\_poly\_pd.columns = ["Actual\_Poly", "Polynomial\_Predicted"]  
  
test\_predictions\_rf\_pd = test\_predictions\_rf.select("SALARY", "prediction").toPandas()  
test\_predictions\_rf\_pd.columns = ["Actual\_RF", "RF\_Predicted"]  
  
predictions\_df = pd.concat([  
 test\_predictions\_pd,  
 test\_predictions\_poly\_pd["Polynomial\_Predicted"],  
 test\_predictions\_rf\_pd["RF\_Predicted"]  
], axis=1)  
  
# Create visualization  
fig, axes = plt.subplots(2, 2, figsize=(14, 12))  
fig.suptitle('Actual vs Predicted Salary - Model Comparison', fontsize=16, fontweight='bold')  
  
# Linear Regression plot  
sns.scatterplot(data=predictions\_df, x="Actual", y="Linear\_Predicted", alpha=0.6, ax=axes[0, 0])  
axes[0, 0].plot([predictions\_df["Actual"].min(), predictions\_df["Actual"].max()],  
 [predictions\_df["Actual"].min(), predictions\_df["Actual"].max()],  
 'r--', lw=2, label='Perfect Prediction')  
axes[0, 0].set\_title(f'Linear Regression\nR²={test\_r2:.4f}, RMSE=${test\_rmse:,.0f}')  
axes[0, 0].set\_xlabel('Actual Salary ($)')  
axes[0, 0].set\_ylabel('Predicted Salary ($)')  
axes[0, 0].legend()  
axes[0, 0].grid(True, alpha=0.3)  
  
# Polynomial Regression plot  
sns.scatterplot(data=predictions\_df, x="Actual", y="Polynomial\_Predicted", alpha=0.6, ax=axes[0, 1], color='green')  
axes[0, 1].plot([predictions\_df["Actual"].min(), predictions\_df["Actual"].max()],  
 [predictions\_df["Actual"].min(), predictions\_df["Actual"].max()],  
 'r--', lw=2, label='Perfect Prediction')  
axes[0, 1].set\_title(f'Polynomial Regression\nR²={test\_r2\_poly:.4f}, RMSE=${test\_rmse\_poly:,.0f}')  
axes[0, 1].set\_xlabel('Actual Salary ($)')  
axes[0, 1].set\_ylabel('Predicted Salary ($)')  
axes[0, 1].legend()  
axes[0, 1].grid(True, alpha=0.3)  
  
# Random Forest plot  
sns.scatterplot(data=predictions\_df, x="Actual", y="RF\_Predicted", alpha=0.6, ax=axes[1, 0], color='orange')  
axes[1, 0].plot([predictions\_df["Actual"].min(), predictions\_df["Actual"].max()],  
 [predictions\_df["Actual"].min(), predictions\_df["Actual"].max()],  
 'r--', lw=2, label='Perfect Prediction')  
axes[1, 0].set\_title(f'Random Forest\nR²={test\_r2\_rf:.4f}, RMSE=${test\_rmse\_rf:,.0f}')  
axes[1, 0].set\_xlabel('Actual Salary ($)')  
axes[1, 0].set\_ylabel('Predicted Salary ($)')  
axes[1, 0].legend()  
axes[1, 0].grid(True, alpha=0.3)  
  
# Residuals distribution  
residuals\_data = pd.DataFrame({  
 'Linear': predictions\_df["Actual"] - predictions\_df["Linear\_Predicted"],  
 'Polynomial': predictions\_df["Actual"] - predictions\_df["Polynomial\_Predicted"],  
 'Random Forest': predictions\_df["Actual"] - predictions\_df["RF\_Predicted"]  
})  
  
residuals\_melted = residuals\_data.melt(var\_name='Model', value\_name='Residual')  
sns.boxplot(data=residuals\_melted, x='Model', y='Residual', ax=axes[1, 1])  
axes[1, 1].axhline(y=0, color='r', linestyle='--', lw=2)  
axes[1, 1].set\_title('Residuals Distribution by Model')  
axes[1, 1].set\_xlabel('Model')  
axes[1, 1].set\_ylabel('Residual (Actual - Predicted)')  
axes[1, 1].grid(True, alpha=0.3)  
  
plt.tight\_layout()  
plt.show()  
  
print("\n" + "=" \* 80)  
print("INTERPRETATION OF MODEL COMPARISON")  
print("=" \* 80)  
  
print("\n1. INFORMATION CRITERIA INTERPRETATION:")  
print(" - AIC (Akaike Information Criterion):")  
print(" \* Measures model quality balancing fit and complexity")  
print(" \* LOWER is better - penalizes adding unnecessary parameters")  
print(" \* Best for prediction-focused model selection")  
  
print("\n - BIC (Bayesian Information Criterion):")  
print(" \* Similar to AIC but penalizes complexity more strongly")  
print(" \* LOWER is better - stronger penalty for model complexity")  
print(" \* Best for identifying the 'true' model structure")  
  
print("\n2. FINAL RECOMMENDATION:")  
votes = {'Linear Regression': 0, 'Polynomial Regression': 0, 'Random Forest': 0}  
votes[comparison\_df.iloc[best\_r2\_idx]['Model']] += 1  
votes[comparison\_df.iloc[best\_rmse\_idx]['Model']] += 1  
votes[comparison\_df.iloc[best\_aic\_idx]['Model']] += 1  
votes[comparison\_df.iloc[best\_bic\_idx]['Model']] += 1  
  
winner = max(votes, key=votes.get)  
print(f" Based on R², RMSE, AIC, and BIC: {winner} wins {votes[winner]}/4 metrics")  
  
if winner == 'Linear Regression':  
 print("\n RECOMMENDATION: Use Linear Regression")  
 print(" - Simplest model with competitive performance")  
 print(" - Most interpretable coefficients")  
elif winner == 'Polynomial Regression':  
 print("\n RECOMMENDATION: Use Polynomial Regression")  
 print(" - Captures non-linear relationships")  
 print(" - Still interpretable")  
else:  
 print("\n RECOMMENDATION: Use Random Forest")  
 print(" - Best predictive performance")  
 print(" - Handles complex non-linear patterns")  
  
print("\n IMPORTANT: All models suffer from data leakage (SALARY\_FROM/SALARY\_TO)")  
print(" Remove these features and retrain for realistic production deployment")  
  
print("\n" + "=" \* 80)

[Stage 69:> (0 + 1) / 1] [Stage 70:> (0 + 1) / 1] [Stage 71:> (0 + 1) / 1]

================================================================================  
COMPREHENSIVE MODEL COMPARISON  
================================================================================  
  
 Model Test\_R2 Test\_RMSE Test\_MAE Train\_R2 Overfitting\_Gap AIC BIC Log\_Likelihood Num\_Parameters  
 Linear Regression 0.999390 842.199336 321.346659 0.999007 -0.000383 11365.733162 11397.550610 -5675.866581 7  
Polynomial Regression 0.999389 842.917956 318.554628 0.999009 -0.000380 11368.931295 11405.294093 -5676.465648 8  
 Random Forest 0.983800 4341.455231 2179.852597 0.981460 -0.002339 13648.540987 13680.358434 -6817.270493 7  
  
================================================================================  
BEST PERFORMERS BY METRIC  
================================================================================  
  
Best R² Score: Linear Regression (0.9994)  
Best RMSE: Linear Regression ($842.20)  
Best MAE: Polynomial Regression ($318.55)  
Best AIC: Linear Regression (11,365.73)  
Best BIC: Linear Regression (11,397.55)

[Stage 72:> (0 + 1) / 1] [Stage 73:> (0 + 1) / 1] [Stage 74:> (0 + 1) / 1]



================================================================================  
INTERPRETATION OF MODEL COMPARISON  
================================================================================  
  
1. INFORMATION CRITERIA INTERPRETATION:  
 - AIC (Akaike Information Criterion):  
 \* Measures model quality balancing fit and complexity  
 \* LOWER is better - penalizes adding unnecessary parameters  
 \* Best for prediction-focused model selection  
  
 - BIC (Bayesian Information Criterion):  
 \* Similar to AIC but penalizes complexity more strongly  
 \* LOWER is better - stronger penalty for model complexity  
 \* Best for identifying the 'true' model structure  
  
2. FINAL RECOMMENDATION:  
 Based on R², RMSE, AIC, and BIC: Linear Regression wins 4/4 metrics  
  
 RECOMMENDATION: Use Linear Regression  
 - Simplest model with competitive performance  
 - Most interpretable coefficients  
  
 IMPORTANT: All models suffer from data leakage (SALARY\_FROM/SALARY\_TO)  
 Remove these features and retrain for realistic production deployment  
  
================================================================================