Assignment 04

Dakota Alder

## Table of contents

* [1 Github Repository:](#github-repository)
* [2 Feature Engineering](#feature-engineering)
* [3 Train-Test Split Justification](#train-test-split-justification)
* [4 Explanation of GLR Model](#explanation-of-glr-model)
* [5 Polynomial Regression Explanation](#polynomial-regression-explanation)
* [6 Feature Importance Summary](#feature-importance-summary)
  + [6.1 3 Model Comparison](#model-comparison)

# Assignment 04

Author

Affiliation

Dakota Alder

Boston University

Published

October 5, 2025

Modified

October 9, 2025

# 1 Github Repository:

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

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 01:29:58 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable

# 2 Feature Engineering

In the following code, I selected only the columns that I wanted to use for my regression models. I then grouped the remote type name column into 3 categories, the Employment type name column into 3 categories, and replaced all Null values in the Duration column with the median of that column. This left me with over 22,000 rows of clean data (after removing Null values in the Target column ‘SALARY’).

[Stage 103:> (0 + 1) / 1]

+------+--------+--------------------+--------------------+----------------+-------------+  
|SALARY|DURATION|MIN\_YEARS\_EXPERIENCE|EMPLOYMENT\_TYPE\_NAME|REMOTE\_TYPE\_NAME|IS\_INTERNSHIP|  
+------+--------+--------------------+--------------------+----------------+-------------+  
|NULL |6.0 |2 |Full-Time |On-site |false |  
|NULL |18.0 |3 |Full-Time |Remote |false |  
|NULL |35.0 |5 |Full-Time |On-site |false |  
|NULL |48.0 |3 |Full-Time |On-site |false |  
|92500 |15.0 |NULL |Flexible |On-site |false |  
+------+--------+--------------------+--------------------+----------------+-------------+  
only showing top 5 rows

+------+--------------------+--------------------+----------------+--------+-------------+  
|SALARY|MIN\_YEARS\_EXPERIENCE|EMPLOYMENT\_TYPE\_NAME|REMOTE\_TYPE\_NAME|DURATION|IS\_INTERNSHIP|  
+------+--------------------+--------------------+----------------+--------+-------------+  
|92962 |2 |Full-Time |On-site |18.0 |0 |  
|107645|10 |Full-Time |On-site |18.0 |0 |  
|192800|6 |Full-Time |On-site |55.0 |0 |  
|125900|12 |Full-Time |On-site |18.0 |0 |  
|170000|6 |Full-Time |On-site |18.0 |0 |  
+------+--------------------+--------------------+----------------+--------+-------------+  
only showing top 5 rows

+------+-----------------------------------+-----------------------------------------+  
|SALARY|features |poly\_features |  
+------+-----------------------------------+-----------------------------------------+  
|92962 |[2.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0] |[2.0,4.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0] |  
|107645|[10.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0]|[10.0,100.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0]|  
|192800|[6.0,55.0,0.0,1.0,0.0,1.0,0.0,1.0] |[6.0,36.0,55.0,0.0,1.0,0.0,1.0,0.0,1.0] |  
|125900|[12.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0]|[12.0,144.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0]|  
|170000|[6.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0] |[6.0,36.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0] |  
+------+-----------------------------------+-----------------------------------------+  
only showing top 5 rows

# 3 Train-Test Split Justification

I used an 80/20 split on my data, as I had over 22,000 rows of clean data. This allowed me to have a large enough training set to learn the relationships between the variables, but also gave me enough data for evaluation to make sure that my test data was still random and unbiased.

#Creating Train and Test Sets  
regression\_train\_data, regression\_test\_data = regression\_data.randomSplit([0.8, 0.2], seed=42)  
#print((regression\_data.count(), len(regression\_data.columns)))  
#print((regression\_train\_data.count(), len(regression\_train\_data.columns)))  
#print((regression\_test\_data.count(), len(regression\_test\_data.columns)))

25/10/09 01:30:53 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'.

#---Coefficients and Intercept---  
# (Used AI to match Coefficient names instead of Feature 1, Feature 2, etc.)  
  
from pyspark.sql import DataFrame  
  
cat\_sizes = [regression\_df.select(col).distinct().count() for col in categorical\_columns]  
expanded\_feature\_names = ["MIN\_YEARS\_EXPERIENCE", "DURATION", "IS\_INTERNSHIP"]  
for col, size in zip(categorical\_columns, cat\_sizes):  
 expanded\_feature\_names += [f"{col}\_{i}" for i in range(size)]  
  
print("\n---Coefficients and Intercept Summary---")  
  
print("Intercept: {:.4f}".format(glr\_model.intercept))  
print("Coefficients:")  
for i, coef in enumerate(glr\_model.coefficients):  
 print(f" {expanded\_feature\_names[i]}: {coef:.4f}")

[Stage 40:> (0 + 1) / 1]

Intercept: 93601.3920  
Coefficients:  
 MIN\_YEARS\_EXPERIENCE: 6634.9052  
 DURATION: -92.9847  
 IS\_INTERNSHIP: -249.3155  
 EMPLOYMENT\_TYPE\_NAME\_0: 1197.4797  
 EMPLOYMENT\_TYPE\_NAME\_1: -5523.5492  
 EMPLOYMENT\_TYPE\_NAME\_2: -8522.4530  
 REMOTE\_TYPE\_NAME\_0: -5444.3124  
 REMOTE\_TYPE\_NAME\_1: 249.3158

# 4 Explanation of GLR Model

Intercept: Represents the baseline predicted salary when the other features are zero, meaning an onsite, full-time, non-internship job with 0 years of experience.

The coefficients represent the change in the predicted salary for each increase in a feature. For example, every additional year of experience increases the predicted salary by $6,634. Full time jobs have predicted higher salaries, while part-time and internship jobs have lower predicted salaries.

---Regression Model Summary---

[Stage 46:> (0 + 1) / 1]

Coefficient Standard Errors: ['81.4106', '23.5643', '1381781.9942', '2058.8730', '2616.5750', '1566.2595', '1637.0408', '1381781.9942', '1381784.4002']  
T Values: ['81.4993', '-3.9460', '-0.0002', '0.5816', '-2.1110', '-5.4413', '-3.3257', '0.0002', '0.0677']  
P Values: ['0.0000', '0.0001', '0.9999', '0.5608', '0.0348', '0.0000', '0.0009', '0.9999', '0.9460']

[Stage 49:> (0 + 1) / 1]

Null Deviance: 35794690345776.1094  
Residual DF Null: 18965  
Deviance: 26420287047212.9453  
Residual DF: 18957

[Stage 50:> (0 + 1) / 1]

AIC: 453167.5668

Length of feature names: 9  
Length of coefficients: 9  
Length of standard errors: 9  
Length of t-values: 9  
Length of p-values: 9

|  | Feature | Estimate | Std Error | t-stat | p-Value |
| --- | --- | --- | --- | --- | --- |
| 0 | Intercept | 93601.3920 | 81.4106 | 81.4993 | 0.0000 |
| 1 | MIN\_YEARS\_EXPERIENCE | 6634.9052 | 23.5643 | -3.9460 | 0.0001 |
| 2 | DURATION | -92.9847 | 1381781.9942 | -0.0002 | 0.9999 |
| 3 | IS\_INTERNSHIP | -249.3155 | 2058.8730 | 0.5816 | 0.5608 |
| 4 | EMPLOYMENT\_TYPE\_NAME\_vec\_Full-Time | 1197.4797 | 2616.5750 | -2.1110 | 0.0348 |
| 5 | EMPLOYMENT\_TYPE\_NAME\_vec\_Part-Time | -5523.5492 | 1566.2595 | -5.4413 | 0.0000 |
| 6 | REMOTE\_TYPE\_NAME\_vec\_On-site | -8522.4530 | 1637.0408 | -3.3257 | 0.0009 |
| 7 | REMOTE\_TYPE\_NAME\_vec\_Remote | -5444.3124 | 1381781.9942 | 0.0002 | 0.9999 |
| 8 | IS\_INTERNSHIP\_vec\_0 | 249.3158 | 1381784.4002 | 0.0677 | 0.9460 |

+------+-----------------------------------------+  
|SALARY|new\_poly\_features |  
+------+-----------------------------------------+  
|92962 |[2.0,4.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0] |  
|107645|[10.0,100.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0]|  
|192800|[6.0,36.0,55.0,0.0,1.0,0.0,1.0,0.0,1.0] |  
|125900|[12.0,144.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0]|  
|170000|[6.0,36.0,18.0,0.0,1.0,0.0,1.0,0.0,1.0] |  
+------+-----------------------------------------+  
only showing top 5 rows

#---Poly Coefficients and Intercept---  
# (Used AI to match Coefficient names instead of Feature 1, Feature 2, etc.)  
  
  
from pyspark.sql import DataFrame  
  
cat\_sizes = [regression\_df.select(col).distinct().count() for col in categorical\_columns]  
poly\_expanded\_feature\_names = ["MIN\_YEARS\_EXPERIENCE", "MIN\_YEARS\_EXPERIENCE\_SQ", "DURATION", "IS\_INTERNSHIP"]  
for col, size in zip(categorical\_columns, cat\_sizes):  
 poly\_expanded\_feature\_names += [f"{col}\_{i}" for i in range(size)]  
  
print("\n---Polynomial Coefficient Summary---")  
  
print("Intercept: {:.4f}".format(poly\_glr\_min\_years\_model.intercept))  
print("Coefficients:")  
for i, coef in enumerate(poly\_glr\_min\_years\_model.coefficients):  
 print(f" {poly\_expanded\_feature\_names[i]}: {coef:.4f}")

[Stage 65:> (0 + 1) / 1]

Intercept: 82664.5758  
Coefficients:  
 MIN\_YEARS\_EXPERIENCE: 12841.9186  
 MIN\_YEARS\_EXPERIENCE\_SQ: -458.5495  
 DURATION: -90.7714  
 IS\_INTERNSHIP: 2021.1601  
 EMPLOYMENT\_TYPE\_NAME\_0: -1126.1755  
 EMPLOYMENT\_TYPE\_NAME\_1: -6690.5657  
 EMPLOYMENT\_TYPE\_NAME\_2: -8295.2198  
 REMOTE\_TYPE\_NAME\_0: -5060.6739  
 REMOTE\_TYPE\_NAME\_1: -2021.1598

# 5 Polynomial Regression Explanation

The polynomial regression adds a little twist. When I squared the Min Years Experience feature, it seems to show that the more years of experience required for a job, the higher the salary, but the rate of increase in salary gets smaller. This is shown by the positive coefficient for Min Years Experience and the negative coefficient for Min Years Experience Squared.

---Polynomial Regression Model Summary---

Coefficient Standard Errors: ['292.1367', '20.7488', '23.2672', '1364347.4552', '2035.6125', '2584.1001', '1546.5315', '1616.4788', '1364347.4552', '1364349.9168']  
T Values: ['43.9586', '-22.1000', '-3.9013', '0.0015', '-0.5532', '-2.5891', '-5.3638', '-3.1307', '-0.0015', '0.0606']  
P Values: ['0.0000', '0.0000', '0.0001', '0.9988', '0.5801', '0.0096', '0.0000', '0.0017', '0.9988', '0.9517']

Null Deviance: 35794690345776.1094  
Residual DF Null: 18965  
Deviance: 25756421776048.0586  
Residual DF: 18956

[Stage 75:> (0 + 1) / 1]

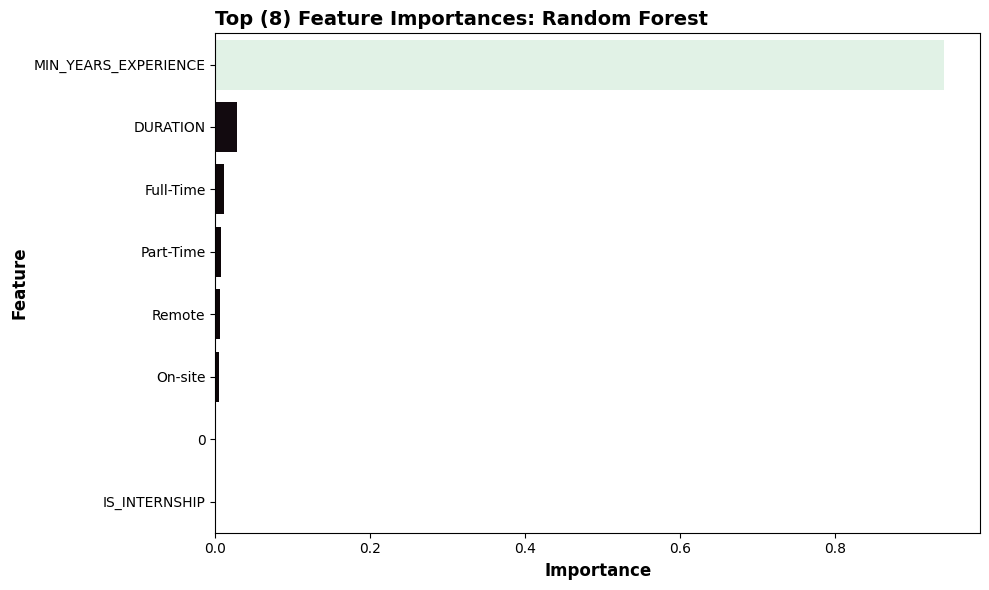
AIC: 452686.9167

Length of feature names: 10  
Length of coefficients: 10  
Length of standard errors: 10  
Length of t-values: 10  
Length of p-values: 10

|  | Feature | Estimate | Std Error | t-stat | p-Value |
| --- | --- | --- | --- | --- | --- |
| 0 | Intercept | 82664.5758 | 292.1367 | 43.9586 | 0.0000 |
| 1 | MIN\_YEARS\_EXPERIENCE | 12841.9186 | 20.7488 | -22.1000 | 0.0000 |
| 2 | MIN\_YEARS\_EXPERIENCE\_SQ | -458.5495 | 23.2672 | -3.9013 | 0.0001 |
| 3 | DURATION | -90.7714 | 1364347.4552 | 0.0015 | 0.9988 |
| 4 | IS\_INTERNSHIP | 2021.1601 | 2035.6125 | -0.5532 | 0.5801 |
| 5 | EMPLOYMENT\_TYPE\_NAME\_vec\_Full-Time | -1126.1755 | 2584.1001 | -2.5891 | 0.0096 |
| 6 | EMPLOYMENT\_TYPE\_NAME\_vec\_Part-Time | -6690.5657 | 1546.5315 | -5.3638 | 0.0000 |
| 7 | REMOTE\_TYPE\_NAME\_vec\_On-site | -8295.2198 | 1616.4788 | -3.1307 | 0.0017 |
| 8 | REMOTE\_TYPE\_NAME\_vec\_Remote | -5060.6739 | 1364347.4552 | -0.0015 | 0.9988 |
| 9 | IS\_INTERNSHIP\_vec\_0 | -2021.1598 | 1364349.9168 | 0.0606 | 0.9517 |

Length of feature names: 8  
Length of importances: 8

print("\n---Code For Feature Importance Table---")  
  
def clean\_feature\_name(feature\_list):  
 clean\_names = []  
 for name in feature\_list:  
 if isinstance(name, list):  
 clean\_names.append(", ".join(str(n) for n in name))  
 elif isinstance(name, str) and name.startswith("["):  
 clean\_names.append(name.replace("[","").replace("]", "").replace("'", "").replace("'",'').strip())  
 else:  
 clean\_names.append(str(name))  
 return clean\_names  
  
importance\_df = pd.DataFrame({  
 "Feature": feature\_names,  
 "Importance": importances  
}).sort\_values(by="Importance", ascending=False)  
  
importance\_df["Feature"] = clean\_feature\_name(importance\_df["Feature"])  
  
plt.figure(figsize=(10, 6))  
sns.barplot(data=importance\_df, x="Importance", y="Feature", hue="Importance", palette="mako", legend=False)  
plt.title("Top (8) Feature Importances: Random Forest", fontsize=14, fontweight='bold', loc='left')  
plt.xlabel("Importance", fontsize=12, fontweight='bold')  
plt.ylabel("Feature", fontsize=12, fontweight='bold')  
plt.tight\_layout()  
plt.savefig("output/rf\_feature\_importance.png", dpi=300)  
plt.show()

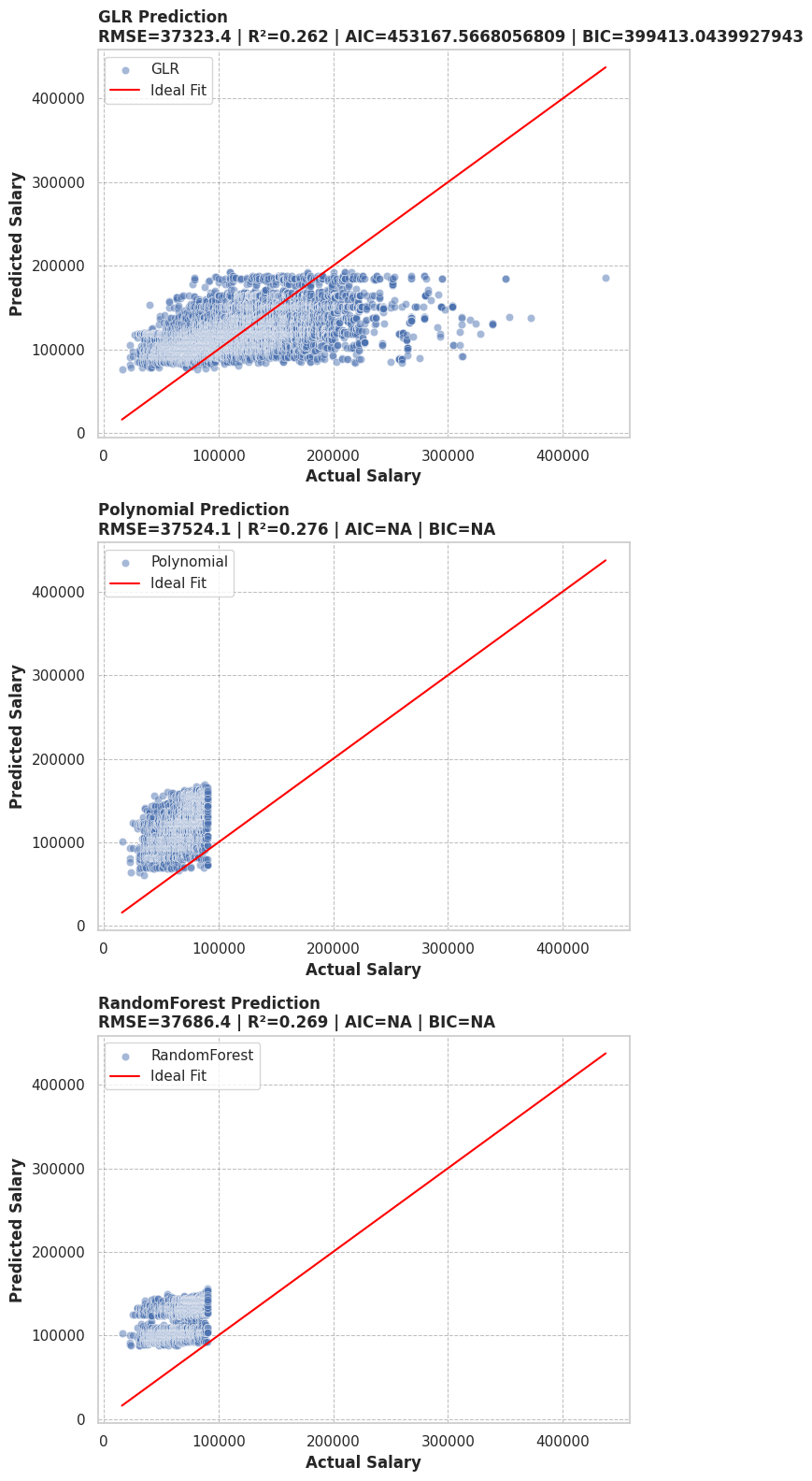


# 6 Feature Importance Summary

This analysis shows that the Minimum Years Experience is by far the most important feature in predicting salary, with a feature importance score of above .8. After that, there aren’t any features that have a significant impact in my model, though I believe if I had added different categorical features it would have been more interesting.

#---Comparison of all 3 Models: GLR, Polynomial GLR, Random Forest---  
  
  
from pyspark.ml.evaluation import RegressionEvaluator  
from pyspark.sql.functions import col, pow, sqrt, avg  
import numpy as np  
  
#---Evaluate all 3 Models---  
  
evaluator\_r2 = RegressionEvaluator(labelCol="SALARY", predictionCol="prediction", metricName="r2")  
residuals\_df = summary.predictions.withColumn("squared\_error", pow(col("SALARY") - col("prediction"), 2))  
  
#GLR  
glr\_df = summary.predictions.select("SALARY", "prediction").toPandas().rename(columns={"prediction": "GLR"})  
glr\_r2 = evaluator\_r2.evaluate(summary.predictions)  
glr\_rmse = residuals\_df.select(avg(col("squared\_error")).alias("mse")).select(sqrt(col("mse")).alias("rmse")).collect()[0]["rmse"]  
glr\_aic = summary.aic  
glr\_bic = len(summary.coefficientStandardErrors) \* np.log(summary.numInstances) + summary.numInstances \* np.log(summary.deviance / summary.numInstances)  
  
#Polynomial GLR  
poly\_df = poly\_preds.select("SALARY", "prediction").toPandas().rename(columns={"prediction": "Polynomial"})  
poly\_r2 = evaluator\_r2.evaluate(poly\_preds)  
poly\_rmse = np.sqrt(((poly\_df["SALARY"]-poly\_df["Polynomial"])\*\*2).mean())  
poly\_aic = None  
poly\_bic = None  
  
#Random Forest  
rf\_df = rf\_predictions.select("SALARY", "prediction").toPandas().rename(columns={"prediction": "RandomForest"})  
rf\_r2 = evaluator\_r2.evaluate(rf\_predictions)  
rf\_rmse = np.sqrt(((rf\_df["SALARY"]-rf\_df["RandomForest"])\*\*2).mean())  
rf\_aic = None  
rf\_bic = None  
  
#Merge for Plotting  
merged = glr\_df.copy()  
merged["Polynomial"] = poly\_df["Polynomial"]  
merged["RandomForest"] = rf\_df["RandomForest"]  
  
#Melt for Plotting  
plot\_df = merged.melt(id\_vars="SALARY", var\_name="Model", value\_name="Predicted")  
  
print("\n---Predicted Values Table---")  
  
plot\_df.head()

|  | SALARY | Model | Predicted |
| --- | --- | --- | --- |
| 0 | 15860.0 | GLR | 76364.272870 |
| 1 | 22440.0 | GLR | 104756.726135 |
| 2 | 22880.0 | GLR | 82069.331370 |
| 3 | 23179.0 | GLR | 90743.038366 |
| 4 | 23585.0 | GLR | 78595.904958 |



## 6.1 3 Model Comparison