Dante Lambros
dlambros415@gmail.com
Boston, MA

Data Science Salary Study

Variable Selection

I chose variables that are likely to impact a data scientist's salary. Python, SQL, and Excel were included because they are common job requirements and may reflect skill level. Company size helps capture differences in pay between small and large organizations. State was used to account for regional salary differences, and sector helps adjust for pay differences across industries. Together, these variables give a well-rounded view of what affects data scientist salaries

Model 1 (Multivariate Linear Regression)

This model predicts salary using company size (number.of.employee), job location (State), three core skills (Python, SQL, Excel), and industry sector (Sector). The model had an R-squared of 0.3672, meaning it explains about 36.7% of the variation in salaries. The overall model is statistically significant (F-statistic = 6.116, $p < 0.001$). Python had the strongest positive effect, increasing salary by around $22,320. SQL and Excel were both statistically significant but negatively associated with salary, possibly reflecting their use in more entry-level roles. State had a major impact. For example, salaries in California were $69,190 higher on average than the baseline. Sector was not individually significant, but it's helpful as a control to account for industry differences.

Model 2 (Fixed Effects Regression for 'State')

This model uses the same structure as Model 1 but focuses on using State as a fixed effect — meaning it controls for geographic differences in salary without interpreting each state coefficient. The model remains statistically significant overall (F-statistic = 6.116, $p < 0.001$) and maintains an R-squared of 0.3672. Python remains the strongest positive predictor, adding approximately $22,320 to salary. SQL and Excel are both significant but negatively associated with salary. By including State as a fixed effect, this model helps isolate the true effect of skills and company size by accounting for underlying differences in location.

Model 3 (Logarithmic Regression)

In this model, I used the natural log of salary (log(Salary)) as the outcome to smooth out high salary outliers and focus on percentage changes. The model performed slightly better than the previous models, with an R-squared of 0.3803 and an F-statistic of 6.468 ($p < 0.001$), indicating strong overall significance. Python continued to be the most impactful skill, associated with an

estimated 23.7% higher salary. Both SQL and Excel were still statistically significant and slightly negative, with associated decreases of about 5.8% and 5.6% respectively. Company size remained significant, but the interpretation changes: for each additional employee, salary increases slightly on a percentage scale. State effects remained strong, for example, being in California is associated with a salary about 75% higher than the baseline state. This model provides a more nuanced, realistic interpretation by focusing on relative salary differences instead of absolute dollar amounts.

Model 4 (Polynomial Regression)

In this model, I built on the success of the logarithmic regression by introducing a squared term for company size. This polynomial approach allows us to test whether salary growth slows down or changes direction at larger company sizes. The model slightly improved upon the prior results with an R-squared of 0.3816, the highest so far, and an F-statistic of 6.393 ($p < 0.001$), showing strong model fit. The squared term for company size was not statistically significant, suggesting that salary growth does not meaningfully flatten or decline at large firms in this sample. However, the original company size term remained significant, and the key takeaways were consistent: Python skill was associated with ~23.4% higher salary, while SQL and Excel both had small, negative impacts. Many states again showed strong salary effects — for example, California still showed a ~75% higher expected salary, even after controlling for all other factors. This model serves as a final test of diminishing returns from company size and reinforces the core insights from previous models.

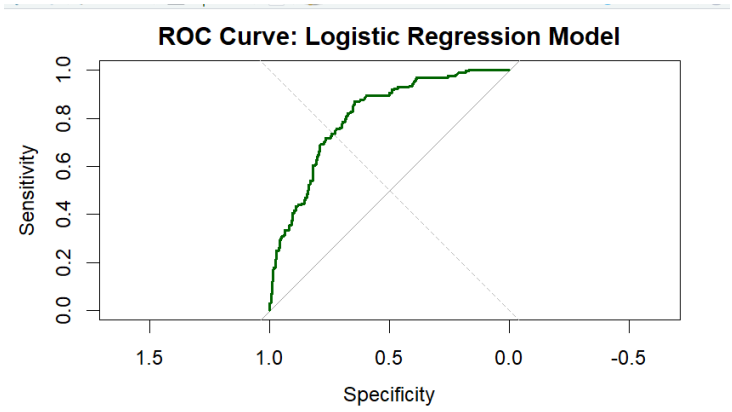
Logit Model (5th model) (Logistic Regression - Predicting Above-Median Salary)

In this model, I shifted to a classification approach by predicting whether a data scientist's salary exceeds the dataset's median of $96,250. I used a logistic regression model with the same independent variables as in the linear models. The model achieved a residual deviance of 764.95 (down from a null deviance of 1009.22), indicating reasonable explanatory power for a binary outcome. The strongest predictor was Python, which was highly statistically significant ($p < 0.001$), showing that job postings requiring Python are much more likely to offer above-median salaries. Excel also had a negative and statistically significant effect ($p < 0.01$), suggesting that jobs requiring Excel may be associated with lower pay. Most sector and state effects were not statistically significant in this model, possibly due to separation or class imbalance, but the overall model still provides a solid framework for classifying salary levels.

ROC Curve v1

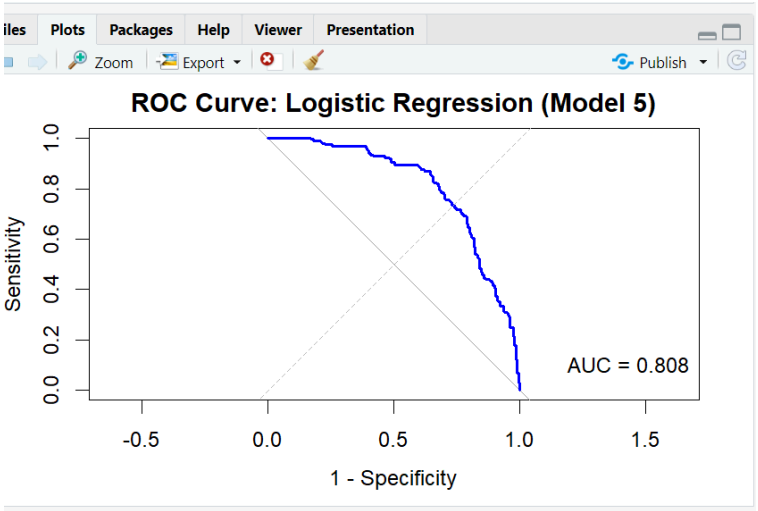
This ROC curve visualizes the performance of the initial logistic regression model predicting whether a data scientist earns above the median salary. The curve is fairly smooth and arcs above the 45° diagonal line, showing that the model performs better than random guessing. While the shape is promising, the axis scaling appears reversed and lacks a visible AUC (Area

Under Curve) value. These presentation issues limit interpretability but still suggest a moderately effective classification model.



ROC Curve v2

In this refined version of the ROC curve, I addressed formatting issues and standardized the axis labels. The final ROC plot shows a cleaner, properly scaled curve with a strong arc and a calculated AUC of 0.808, indicating solid model performance. This AUC value suggests that the logistic regression model has a good ability to distinguish between high and low salary classes. The improved visual presentation and strong AUC reinforce that this version provides a clearer and more interpretable view of classification quality.



Confusion Matrix – Logistic Regression (Model 4)

To evaluate the classification performance of Model 5, I generated a confusion matrix using a 0.5 probability threshold. The model correctly predicted 285 true positives (above-median salaries) and 252 true negatives (below-median salaries), with 112 false positives and 79 false negatives. This resulted in an overall accuracy of 73.8%, indicating that the model correctly classifies salary status in nearly three out of four cases.

The confusion matrix shows that the model performs slightly better at identifying above-median salaries than below-median ones, though both outcomes are captured with reasonable balance. Combined with the ROC AUC of 0.808, this confirms that the logistic regression model has practical value for salary classification.

```
> print(conf_matrix)
         Actual
Predicted   0   1
        0 252  79
        1 112 285
>
> # Optional: Calculate accuracy
> accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
> cat("Accuracy:", round(accuracy, 3), "\n")
Accuracy: 0.738
>
```

Conclusion (c)

The goal of this project was to understand what factors influence data scientist salaries and to build models that can both predict exact salary levels and classify whether a salary is above or below the median. I began with linear regression models to capture continuous salary trends, then explored log and polynomial transformations to better handle skewed data and nonlinear effects. Finally, I shifted to a logistic model to classify salary levels. Across all models, I found that Python skill and job location had the strongest, most consistent impact. This sequence of models reflects a logical progression from interpreting salary drivers to building a predictive classification tool.

CODE

```
df <- read.csv("C:/Users/bclam/OneDrive/Documents/Data Scientist Salary_2021.csv")

# Check the first few rows

head(df)



# Get the column names

colnames(df)
```

```
# Create the State variable from Job.Location

df$State <- substr(df$Job.Location, nchar(df$Job.Location) - 1, nchar(df$Job.Location))


# Drop rows with any missing values in selected columns

model_data <- df[!is.na(df$Salary) &

                   !is.na(df$number.of.employee) &

                   !is.na(df$Python) &

                   !is.na(df$sql) &

                   !is.na(df$excel) &

                   !is.na(df$Sector) &

                   !is.na(df$State),

                 c("Salary", "State", "number.of.employee", "Python", "sql", "excel", "Sector")]

# Convert Sector and State to factor variables

model_data$Sector <- as.factor(model_data$Sector)

model_data$State <- as.factor(model_data$State)

# Run multivariate linear regression

model1 <- lm(Salary ~ State + number.of.employee + Python + sql + excel + Sector, data =
model_data)


# View summary

summary(model1)


# Model 2: Fixed Effects Regression using State
```

```
model2 <- lm(Salary ~ number.of.employee + Python + sql + excel + Sector + State, data = model_data)


# View the summary

summary(model2)


# Add log of salary column

model_data$log_salary <- log(model_data$Salary)

# Model 3: Logarithmic Regression

model3 <- lm(log_salary ~ number.of.employee + Python + sql + excel + Sector + State, data = model_data)


# View the results

summary(model3)


# Add a squared term for company size

model_data$company_size_sq <- model_data$number.of.employee^2


# Model 4: Polynomial Regression with squared company size

model4 <- lm(log_salary ~ number.of.employee + company_size_sq + Python + sql + excel + Sector + State, data = model_data)


# View the summary

summary(model4)

#visual exploration + model 5

median_salary <- median(model_data$Salary, na.rm = TRUE)
```

```
print(median_salary)


model_data$above_median <- ifelse(model_data$Salary > 96250, 1, 0)


logit_model <- glm(above_median ~ number.of.employee + Python + sql + excel + Sector +
State,
                   data = model_data,
                   family = binomial)


summary(logit_model)


#visual summaries


# Load necessary library
library(ggplot2)
library(broom)


# Tidy the model results
model_tidy <- tidy(logit_model)  # Replace with your model variable


# Remove intercept if you don't want it in the plot
model_tidy <- subset(model_tidy, term != "(Intercept)")


# Create the plot
ggplot(model_tidy, aes(x = estimate, y = reorder(term, estimate))) +
```

```
  geom_point() +
  geom_errorbarh(aes(xmin = estimate - std.error, xmax = estimate + std.error), height = 0.2) +
  geom_vline(xintercept = 0, linetype = "dashed") +
  labs(
    title = "Logistic Regression Coefficients",
    x = "Log-Odds Estimate",
    y = "Predictor"
  ) +
  theme_minimal(base_size = 12)
# Load necessary library
library(ggplot2)

# Get predicted probabilities from the model
model_data$predicted_prob <- predict(logit_model, type = "response")

# Plot predicted probability by Python skill
ggplot(model_data, aes(x = factor(Python), y = predicted_prob)) +
  geom_boxplot(fill = "skyblue") +
  labs(title = "Predicted Probability of Salary > Median by Python Skill",
       x = "Knows Python (0 = No, 1 = Yes)",
       y = "Predicted Probability") +
  theme_minimal()

class(data)
```

```
# Load required package
library(pROC)

# Predict probabilities using logistic model
predicted_probs <- predict(logit_model, newdata = model_data, type = "response")

# Generate ROC object
roc_obj <- roc(model_data$above_median ~ predicted_probs)

# Plot ROC curve with improvements
plot(
  roc_obj,
  col = "blue",
  lwd = 2,
  main = "ROC Curve: Logistic Regression (Model 5)",
  xlim = c(0, 1),
  ylim = c(0, 1),
  legacy.axes = TRUE
)

# Add AUC text on the plot
auc_value <- auc(roc_obj)
text(0.6, 0.2, paste("AUC =", round(auc_value, 3)), cex = 1.2)
```

```
# Create binary predictions using 0.5 cutoff
predicted_class <- ifelse(predicted_probs >= 0.5, 1, 0)

# Generate the confusion matrix
conf_matrix <- table(Predicted = predicted_class, Actual = model_data$above_median)

# Print the confusion matrix
print(conf_matrix)

# Optional: Calculate accuracy
accuracy <- sum(diag(conf_matrix)) / sum(conf_matrix)
cat("Accuracy:", round(accuracy, 3), "\n")
```