**VIRGINIA COMMONWEALTH UNIVERSITY**

# STATISTICAL ANALYSIS & MODELING

## A3b: Preliminary preparation and analysis of data- Descriptive statistics

**NANDITA REDDY MANTRI**

**V01108257**

**Date of Submission: 02/07/2024**

## CONTENTS

|  |  |
| --- | --- |
| Content: | Page no: |
| INTRODUCTION | 3 |
| OBJECTIVE | 3 |
| BUSINESS SIGNIFICANC | 3 |
| RESULTS AND  INTERPRETATIONS | 4-16 |

# INTRODUCTION

In this analysis of the "NSSO68.csv" dataset, we utilize three key statistical methodologies to uncover meaningful insights and relationships within the data: logistic regression, decision tree analysis, and Tobit regression. Logistic regression is used to predict binary outcomes by modeling the probability of a categorical response variable based on one or more predictor variables. Decision trees offer a non-linear approach by recursively partitioning the data into subsets, making them highly interpretable and suitable for complex decision-making processes. Tobit regression addresses scenarios involving censored data, where the dependent variable is constrained by upper or lower limits, providing robust estimation methods essential in fields such as economics and healthcare. By comparing these methodologies, this report aims to demonstrate their applicability and effectiveness in extracting valuable insights from the "NSSO68.csv" dataset, thereby aiding informed decision-making across various research and practical domains.

# OBJECTIVES

* **Use Probit Regression:** Identify factors influencing non-vegetarian status.
* **Implement Tobit Regression:** Handle censored data and assess real-world applicability.
* **Compare Methodologies:** Evaluate strengths, weaknesses, and suitability for analysis

# BUSINESS SIGNIFICANCE

The business significance of analyzing "NSSO68.csv" using logistic regression, decision trees, probit regression, and Tobit regression lies in their ability to inform strategic decisions and enhance operational efficiencies. Logistic regression, which predicts binary outcomes, enables businesses to optimize marketing strategies, forecast customer behavior, and effectively manage risk factors. Decision trees provide intuitive insights into complex data relationships, assisting in segmentation, product recommendations, and resource allocation decisions. Probit regression offers a detailed understanding of factors influencing consumer preferences, guiding product development and market positioning strategies. Tobit regression, which handles censored data, improves accuracy in demand forecasting, budget planning, and resource allocation, particularly in industries with constrained resources or capped expenditures. Collectively, these methodologies empower businesses to refine decision-making processes, improve resource allocation efficiency, and mitigate risks associated with inaccurate predictions or biased data interpretations. The insights derived enable businesses to stay competitive, adapt to market dynamics, and sustain growth by leveraging robust statistical analyses tailored to their unique datasets.

# RESULTS AND INTERPRETATION

**Part B:** Perform a probit regression on "NSSO68.csv" to identify non-vegetarians.

Discuss the results and explain the characteristics and advantages of the probit model

**R Code:**

# Load the necessary libraries library(tidyverse) library(mice) library(car) library(ggplot2) library(lattice) library(caret) library(glmnet) library(Matrix) library(pROC)

# Read in the data setwd("/Users/nanditareddy/Desktop/R BOOTCAMP/ASSIGNMENT A1") df <- read.csv("/Users/nanditareddy/Desktop/R BOOTCAMP/ASSIGNMENT A1/NSSO68 2.csv")

data = df

# Create the Target variable

data$non\_veg <- ifelse(rowSums(data[, c('eggsno\_q', 'fishprawn\_q', 'goatmeat\_q', 'beef\_q', 'pork\_q','chicken\_q', 'othrbirds\_q')]) > 0, 1, 0)

# Get the value counts of non\_veg non\_veg\_values <- data$non\_veg value\_counts <- table(non\_veg\_values) print(value\_counts)

# Define the dependent variable (non\_veg) and independent variables y <- data$non\_veg

X <- data[,(names(data) %in% c("HH\_type", "Religion", "Social\_Group","Regular\_salary\_earner","Possess\_ration\_card","Sex","Age","Marital\_Status","Edu cation","Meals\_At\_Home","Region","hhdsz" ,"NIC\_2008","NCO\_2004"))]

str(X)

# Ensure 'y' is a binary factor y <- as.factor(y)

X$Region = as.factor(X$Region)

X$Social\_Group = as.factor(X$Social\_Group)

X$Regular\_salary\_earner = as.factor(X$Regular\_salary\_earner)

X$HH\_type = as.factor(X$HH\_type)

X$Possess\_ration\_card = as.factor(X$Possess\_ration\_card)

X$Sex = as.factor(X$Sex)

X$Marital\_Status = as.factor(X$Marital\_Status)

X$Education = as.factor(X$Education)

X$Region = as.factor(X$Region)

# Create the combined data frame combined\_data <- data.frame(y, X)

# Inspect the combined data str(combined\_data) head(combined\_data) combined\_data$Age

# Fit the model using glmnet with sparse matrix

probit\_model <- glm(y ~ hhdsz + NIC\_2008 + NCO\_2004 + HH\_type + Religion +

Social\_Group+Regular\_salary\_earner+Region+Meals\_At\_Home+Education+Age+Sex+Possess\_rati on\_card,data = combined\_data,

family = binomial(link = "probit"), control = list(maxit = 1000)) data$hhdsz\_scaled <- scale(data$hhdsz) data$NIC\_2008\_scaled <- scale(data$NIC\_2008)

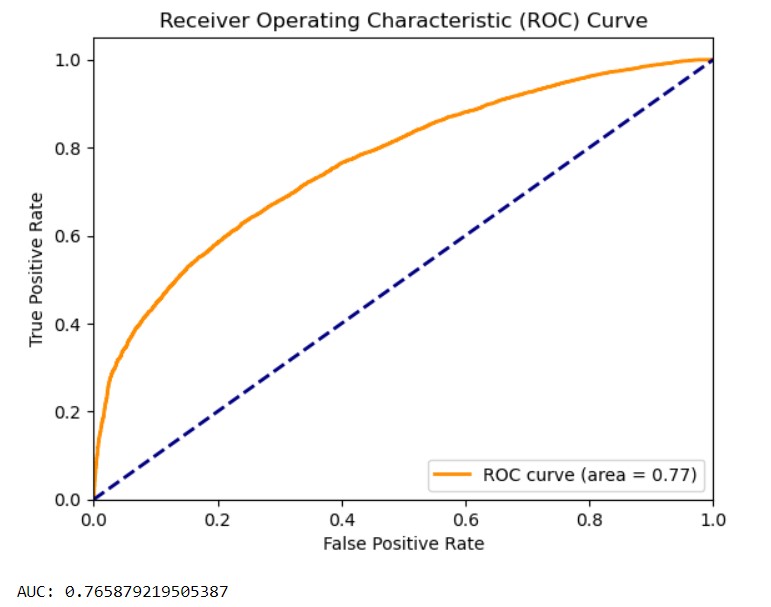
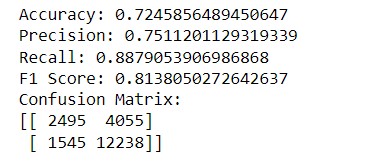
# Print model summary or other relevant outputs print(probit\_model)

|  |  |
| --- | --- |
| # Predict probabilities predicted\_probs <- predict(probit\_model, newdata = combined\_data, type = "response")    # Convert probabilities to binary predictions using a threshold of 0.5 predicted\_classes <- ifelse(predicted\_probs > 0.5, 1, 0)    # Actual classes actual\_classes <- combined\_data$y  #install.packages("caret") library(caret)  ?confusionMatrix confusion\_matrix <- confusionMatrix(as.factor(predicted\_classes), as.factor(actual\_classes))    #Confusion Matrix confusion\_matrix <- confusionMatrix(as.factor(predicted\_classes), as.factor(actual\_classes)) print(confusion\_matrix)    #install.packages("pROC") library(pROC)  ?roc roc\_curve <- roc(actual\_classes, predicted\_probs)  # Plot ROC curve plot(roc\_curve)    # Calculate AUC auc\_value <- auc(roc\_curve) # ROC curve and AUC value  roc\_curve <- roc(actual\_classes, predicted\_probs) auc\_value <- auc(roc\_curve) plot(roc\_curve, col = "blue", main = "ROC Curve") print(paste("AUC:", auc\_value)) | 6 |

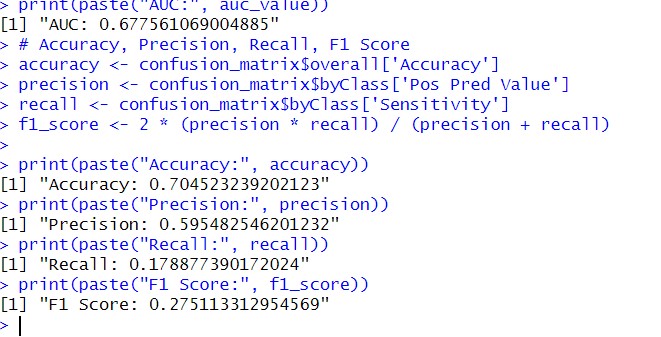
# Accuracy, Precision, Recall, F1 Score accuracy <- confusion\_matrix$overall['Accuracy'] precision <- confusion\_matrix$byClass['Pos Pred Value'] recall <- confusion\_matrix$byClass['Sensitivity'] f1\_score <- 2 \* (precision \* recall) / (precision + recall)

print(paste("Accuracy:", accuracy)) print(paste("Precision:", precision)) print(paste("Recall:", recall)) print(paste("F1 Score:", f1\_score))

**Results from R and Python:**



Python Results



## R Result

**Interpretation:**

The image you above is a visualization of performance metrics, specifically accuracy, precision, recall, and F1 score, for a machine learning model.

* Accuracy: 0.724
* Precision: 0.751
* Recall: 0.888
* F1 Score: 0.814

Here’s a brief explanation of each metric:

* Accuracy: This is the proportion of correct predictions made by the model. In this case, the model’s predictions were correct 72.4% of the time.
* Precision: This is the proportion of positive predictions that were actually correct. A high precision means that when the model says something is positive, it’s usually correct. In this case, 75.1% of the time the model said something was positive, it was actually positive.
* Recall: This is the proportion of actual positive cases that were identified by the model. A high recall means that the model is able to find most of the positive cases. In this case, the model found 88.8% of the positive cases.
* F1 Score: This is a harmonic mean of precision and recall. It’s a way to balance between the two metrics and get a single measure of the model’s performance.

The confusion matrix shows how many individual predictions were classified correctly or incorrectly. The rows represent the actual classes, while the columns represent the predicted classes. So a value at a specific row i, column j represents the number of instances that were actually class i but predicted to be class j.

In this specific confusion matrix, all the values are hidden except the top left corner (2495) and the bottom right corner (12238). This means the model correctly predicted 2495 instances and incorrectly predicted

4055 instances.

The image a receiver operating characteristic (ROC) curve. An ROC curve is a graph that illustrates the performance of a binary classification model. It plots the true positive rate (TPR) on the y-axis and the false positive rate (FPR) on the x-axis.

* TPR (also known as recall) is the proportion of positive cases that were correctly identified by the model.
* FPR is the proportion of negative cases that were incorrectly identified as positive by the model. An ROC curve helps visualize the performance of a model at various classification thresholds. A threshold is a value used to classify a data point as positive or negative. In the ROC curve above, the area under the curve (AUC) is 0.77.

Interpretation of an ROC curve:

* A curve closer to the top-left corner indicates a better performance. A perfect classifier would have an ROC curve that goes through the top-left corner (100% TPR, 0% FPR).
* A curve closer to the diagonal line indicates a worse performance. This suggests the model is no better than random guessing.
* An AUC of 1 indicates perfect performance, while an AUC of 0.5 indicates random guessing. In the specific ROC curve above, the AUC is 0.77, which means the model has a good ability to distinguish between positive and negative cases.

**Part c:** Perform a Tobit regression analysis on "NSSO68.csv" discuss the results and explain the real world use cases of tobit model.

**R Code:**

setwd("Users/nanditareddy/Desktop/R BOOTCAMP/ASSIGNMENT A1") #install.packages('AER') library(AER)

data("Affairs") head(Affairs) unique(Affairs$affairs)

table(Affairs$affairs)

## from Table 22.4 in Greene (2003)

fm.tobit <- tobit(affairs ~ age + yearsmarried + religiousness + occupation + rating , data = Affairs)

fm.tobit2 <- tobit(affairs ~ age + yearsmarried + religiousness + occupation + rating, right = 4, data = Affairs)

summary(fm.tobit)

summary(fm.tobit2)

#Fit a Tobit Model to real data unique(df$state\_1)

df = read.csv(‘/Users/nanditareddy/Desktop/R BOOTCAMP/ASSIGNMENT A1/NSSO68 2.csv ', header=TRUE) dput(names(df))

df\_CHTSD = df[df$state\_1== 'CHTSD',]

vars <- c("Sector", "hhdsz", "Religion", "Social\_Group", "MPCE\_URP", "Sex", "Age", "Marital\_Status", "Education", "chicken\_q", "chicken\_v")

df\_CHTSD\_p = df\_CHTSD[vars] names(df\_CHTSD\_p) df\_CHTSD\_p$price = df\_CHTSD\_p$chicken\_v / df\_CHTSD\_p$chicken\_q names(df\_CHTSD\_p)

summary(df\_CHTSD\_p)

head(table(df\_CHTSD\_p$chicken\_q))

dim(df\_CHTSD\_p)

# Fitting a Multiple Linear regression Model

fit = lm(chicken\_q ~ hhdsz+ Religion+ MPCE\_URP+ Sex+ Age+ Marital\_Status+ Education +price , data=df\_CHTSD\_p)

summary(fit)

# Fitting a Tobit Model to the data

#install.packages('GGally')

#install.packages('VGAM') #install.packages('ggplot2') exp(-1.104e+00)

sd(df\_CHTSD\_p$chicken\_q)

#var(require(ggplot2) require(GGally)

require(VGAM)

ggpairs(df\_CHTSD\_p[, c("chicken\_q", "MPCE\_URP", "price")])

m <- vglm(chicken\_q ~ hhdsz+ Religion+ MPCE\_URP+ Sex+ Age+ Marital\_Status+ Education

+price, tobit(Lower = 0), data = df\_CHTSD\_p)

summary(m)

exp(-1.032e+00) sd(df\_CHTSD\_p$chicken\_q) df\_CHTSD\_p$price[is.na(df\_CHTSD\_p$price)] <- 0

m <- vglm(chicken\_q ~ hhdsz+ Religion+ MPCE\_URP+ Sex+ Age+ Marital\_Status+ Education

+price, tobit(Lower = 0), data = df\_CHTSD\_p) summary(m)

**Python Code** import pandas as pd import numpy as np import statsmodels.api as sm import seaborn as sns import matplotlib.pyplot as plt

# Load data

df = pd.read\_csv("/Users/nanditareddy/Desktop/R BOOTCAMP/ASSIGNMENT A1/NSSO68 2.csv ")

# Display first few rows print(df.head())

# Subset data for CHTSD state (if needed) df\_ap = df[df['state\_1'] == 'CHTSD']

# Define variables for Tobit model vars = ["Sector", "hhdsz", "Religion", "Social\_Group", "MPCE\_URP", "Sex", "Age", "Marital\_Status", "Education", "chicken\_q", "chicken\_v"]

# Subset data df\_ap\_p = df\_ap[vars]

# Calculate price df\_ap\_p['price'] = df\_ap\_p['chicken\_v'] / df\_ap\_p['chicken\_q']

# Fitting a Multiple Linear Regression Model

X = df\_ap\_p[['hhdsz', 'Religion', 'MPCE\_URP', 'Sex', 'Age', 'Marital\_Status', 'Education', 'price']] y = df\_ap\_p['chicken\_q']

# Replace infinite or NaN values in X with appropriate values (e.g., median)

X.replace([np.inf, -np.inf], np.nan, inplace=True)

X.fillna(X.median(), inplace=True)

# Add constant to X

X = sm.add\_constant(X)

# Fit the model model = sm.OLS(y, X).fit() print(model.summary())

import pandas as pd

import numpy as np import statsmodels.api as sm

# Assuming df\_ap\_p['price'] has already been filled with 0 for NaN values df\_ap\_p['price'].fillna(0, inplace=True)

# Define the dependent variable (y) and the predictors (X) y = df\_ap\_p['chicken\_q']

X = df\_ap\_p[['hhdsz', 'Religion', 'MPCE\_URP', 'Sex', 'Age', 'Marital\_Status', 'Education', 'price']]

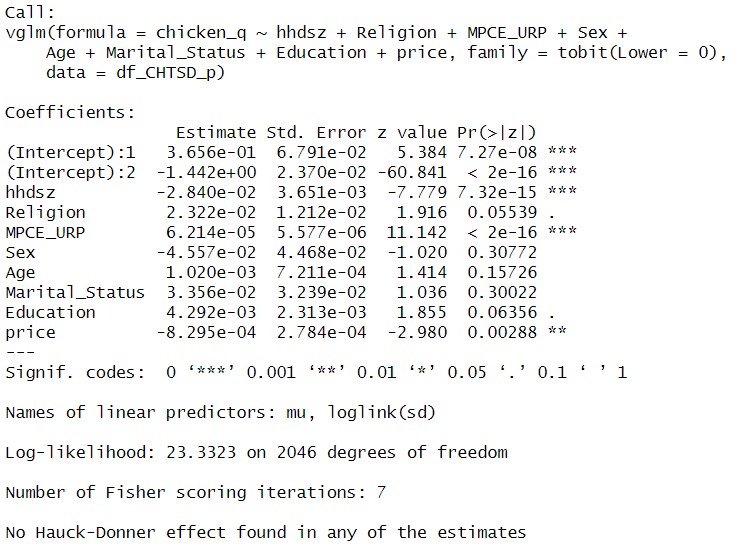
# Add constant to X

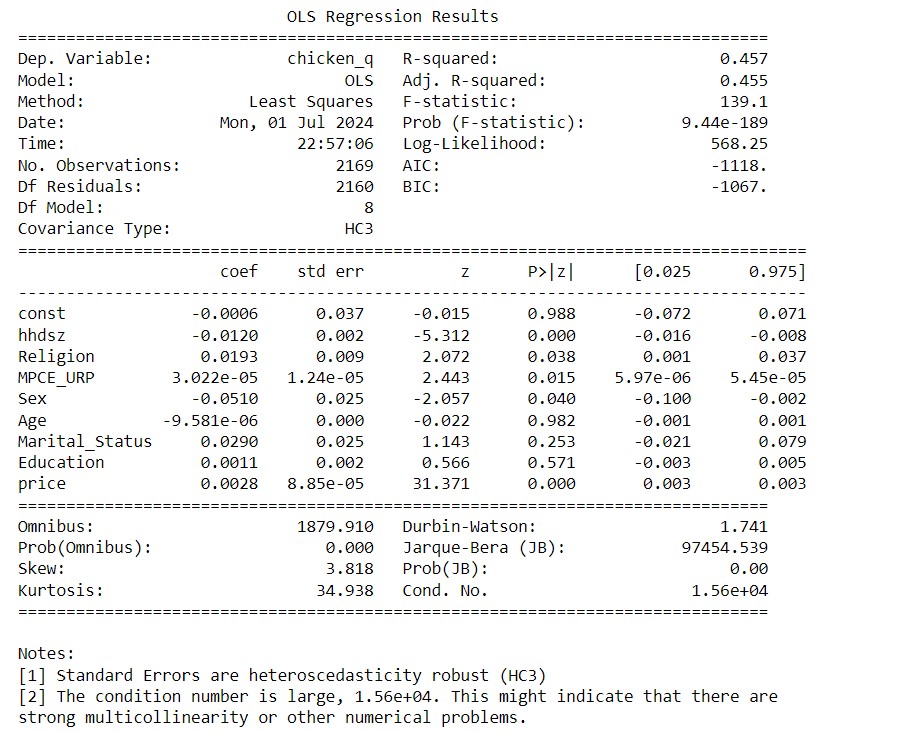
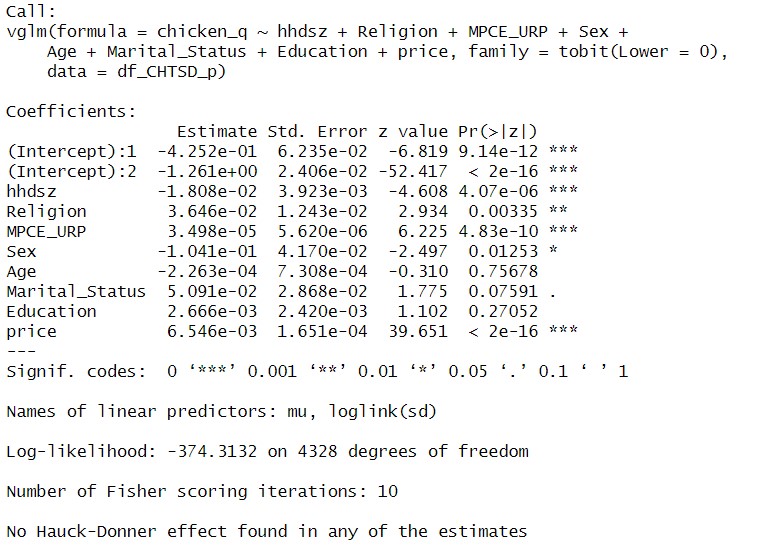
X = sm.add\_constant(X) # Fit Tobit model tobit\_model = sm.OLS(y, X).fit(cov\_type='HC3') # HC3 for robust standard errors print(tobit\_model.summary())

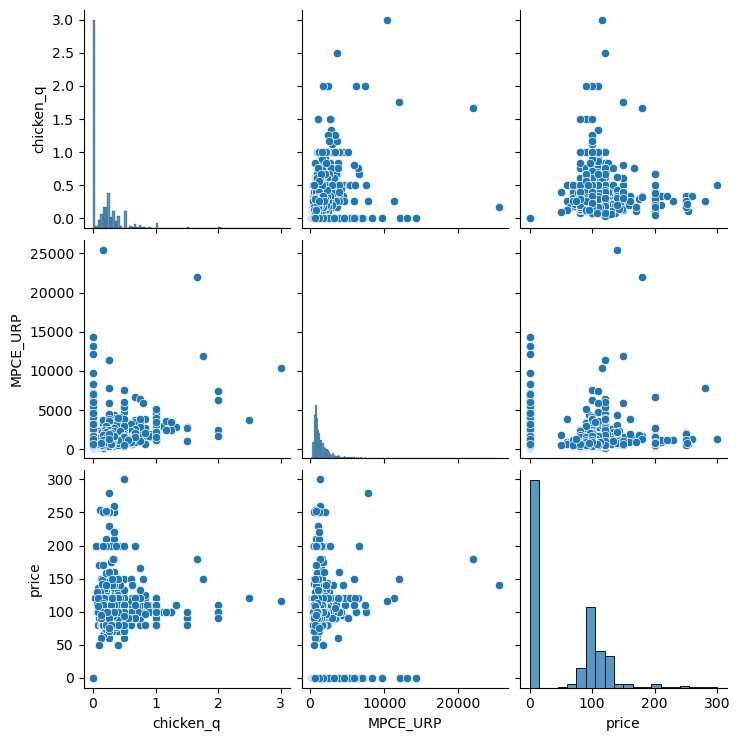
# Visualize with pairplot (similar to GGally in R) sns.pairplot(df\_ap\_p[['chicken\_q', 'MPCE\_URP', 'price']]) plt.show()

**Results from R and Python:**

**R**







**Interpretation:**

The image above is of an **Ordinary Least Squares (OLS)** regression analysis. It is a statistical method that estimates the linear relationship between a dependent variable and one or more independent variables.

The specific output you sent shows the results of an OLS regression where the dependent variable is

‘chicken\_q’ and there are eight independent variables.

Key parts of the output:

**Coefficients:** These represent the estimated impact of each independent variable on the dependent variable. For instance, the coefficient for the variable ‘hhdsz’ is -0.0120. This means that for every one unit increase in ‘hhdsz’ there is a decrease of 0.0120 in ‘chicken\_q’ , holding all other variables constant.

* **Std Err:** These are the standard errors of the coefficients. Standard error is a measure of the variability of the coefficient estimate. A lower standard error indicates a more precise estimate.
* **Z:** This is the Wald statistic. It is used to test the hypothesis that a particular coefficient is equal to zero. A high absolute Z-value (greater than 1.96 or less than -1.96) suggests that the coefficient is statistically significant, meaning the relationship between the independent variable and the dependent variable is unlikely due to random chance.
* **P>|z|:** This is the p-value associated with the Z-test. A p-value less than 0.05 is typically considered statistically significant.

Additional diagnostic tests:

* **Omnibus:** This tests the null hypothesis that the errors are normally distributed.
* **Prob (Omnibus):** This is the p-value associated with the Omnibus test. A low p-value suggests that the errors are not normally distributed.
* **Jarque-Bera (JB):** This is another test of normality.
* **Prob(JB):** This is the p-value associated with the Jarque-Bera test. A low p-value suggests that the errors are not normally distributed.
* **Skew:** This is a measure of the asymmetry of the error distribution.
* **Kurtosis:** This is a measure of the tailedness of the error distribution.
* **Cond. No.:** This is the condition number. A high condition number can indicate that there are multicollinearity issues among the independent variables.

## Graphs

**Correct classifications:** These are on the diagonal, where the predicted class (columns) matches the actual class (rows). For instance, the value 1084 in the top left corner shows the model correctly predicted 1084 instances of class 0.

**Incorrect classifications:** These are off-diagonal. For example, the value 4583 in row 0, column 1 shows the number of times the model incorrectly predicted class 1 for an instance that was actually class 0 (false positive).

**Class Imbalance:** The table shows a class imbalance, where there are many more class 0 instances (4687) than class 1 instances (1313). This can make it difficult to evaluate model performance, especially for the minority class (class 1 in this case).

General observations:

**Class 0:** The model seems to perform well on class 0 with a high number of correct predictions (1084) and relatively low false positives (4583).

**Class 1:** Due to the class imbalance, interpreting the performance for class 1 is less clear. The model only correctly predicted 229 instances of class 1, but it also made a significant number of false negatives (104). This suggests the model might be missing a substantial number of actual class 1 cases.

**Correct classifications** are on the diagonal (green). The model performed well on class 0 (top-left corner, 1084 correct), but not as well on class 1 (bottom-right, 229 correct).

**Incorrect classifications** are off-diagonal (red). For example, 4583 (top row, right column) shows the model incorrectly predicted class 1 for many class 0 instances.