VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

A3: LIMITED DEPENDENT VARIABLE MODELS - CLASSIFICATION ANALYSIS USING R AND PYTHON

ADARSH BHARATHWAJ
V01107513

Date of Submission: 01/07/2024

CONTENTS

CLASSIFICATION ANALYSIS USING PYTHON

INTRODUCTION

The dataset at hand provides a detailed analysis of food consumption patterns within India. It covers various aspects of dietary habits, focusing on both urban and rural sectors within the region. The data includes key metrics such as the quantity of meals consumed at home, specific food item consumption (e.g., rice, wheat, chicken, pulses), and the overall number of meals per day. The campaign_responses dataset is a valuable resource for analyzing customer behavior and predicting responses to marketing campaigns. This dataset contains information about various demographic, financial, and social characteristics of customers, along with their responses to a particular campaign.

OBJECTIVES

a) Conduct a logistic regression analysis on your assigned dataset. Validate assumptions, evaluate with a confusion matrix and ROC curve, and interpret the results. Then, perform a decision tree analysis and compare it to the logistic regression.

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

c) Perform a Tobit regression analysis on "NSSO68.csv" discuss the results and explain the real-world use cases of Tobit model.

BUSINESS SIGNIFICANCE

Logistic regression is a powerful technique for predicting binary outcomes, such as customer responses to campaigns. Its business significance lies in its ability to provide actionable insights and drive strategic decisions such as

1. By identifying the key factors influencing campaign responses, businesses can segment their customer base and target marketing efforts more effectively. This leads to higher response rates and more efficient use of marketing resources.
2. Logistic regression helps in profiling customers based on their likelihood to respond to campaigns. This enables personalized marketing strategies and enhances customer engagement.
3. Understanding which factors significantly impact campaign responses allows businesses to

allocate resources (e.g., marketing budget, human resources) more efficiently to areas with the highest potential return on investment.

Tobit Regression is designed to handle censored data, where the dependent variable is only observed within a certain range. In the context of NSSO68 data by using Tobit Regression, businesses can identify factors driving expenditure patterns and market potential for new products or services. It can also help in segmenting markets based on spending behaviors, aiding in more targeted marketing strategies. Probit Regression is used for modeling binary outcome variables. For NSSO68 data, it can be applied in market Segmentation as businesses in the food industry can use this information to tailor their products and marketing strategies to different demographic segments. For instance, regions with a higher probability of non-vegetarianism may benefit from more non-vegetarian product offerings.

RESULTS AND INTERPRETATION

a) Conduct a logistic regression analysis on your assigned dataset. Validate assumptions, evaluate with a confusion matrix and ROC curve, and interpret the results. Then, perform a decision tree analysis and compare it to the logistic regression. [campaign_responses]

1. Code:

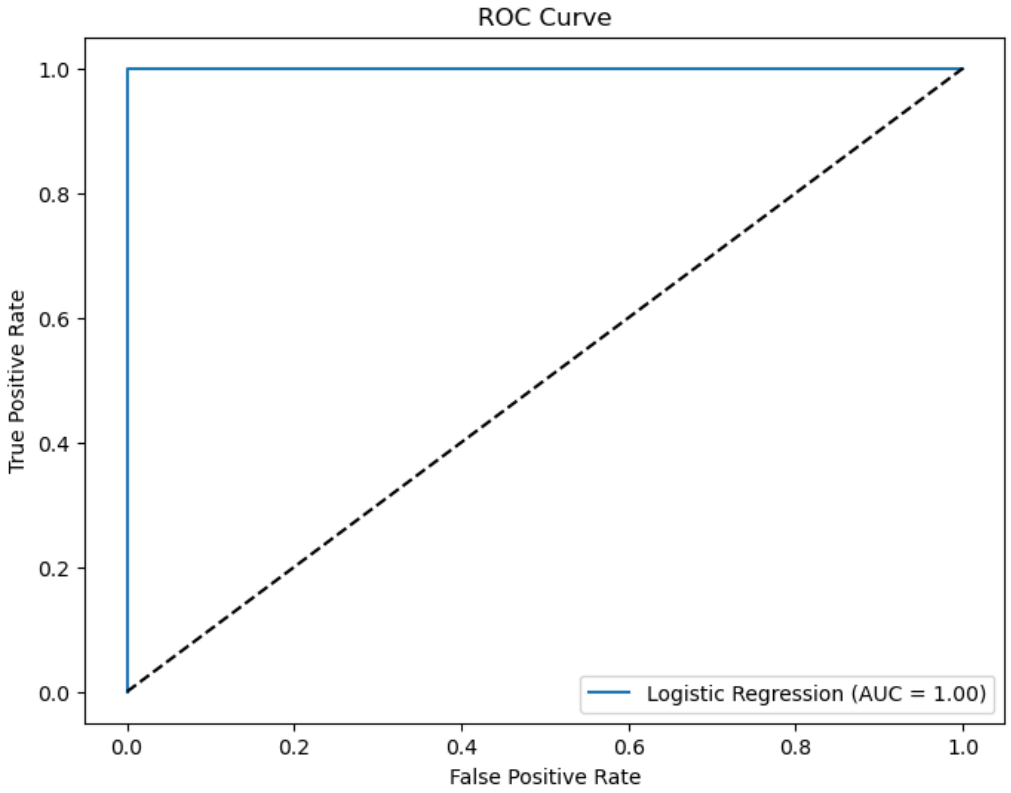
```
#Identify categorical columns
categorical_columns = data.select_dtypes(include=['object']).columns

# Option 1: Label Encoding (for binary categorical data)
label_encoder = LabelEncoder()
for column in categorical_columns:
    data[column] = label_encoder.fit_transform(data[column])

# Assume 'response' is the target variable and the rest are predictors
target = 'responded'
predictors = [col for col in data.columns if col != target]

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(data[predictors], data[target], test_size=0.3,
random_state=42)


# Fit the logistic regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)


# Predict on the test set
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

# Evaluate the model
conf_matrix = confusion_matrix(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_prob)

# Print the model coefficients
coef_df = pd.DataFrame({'Variable': X_train.columns, 'Coefficient': model.coef_[0]})
print(coef_df)

# Plot the ROC curve
fpr, tpr, _ = roc_curve(y_test, y_prob)
plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False Positive Rate')
```

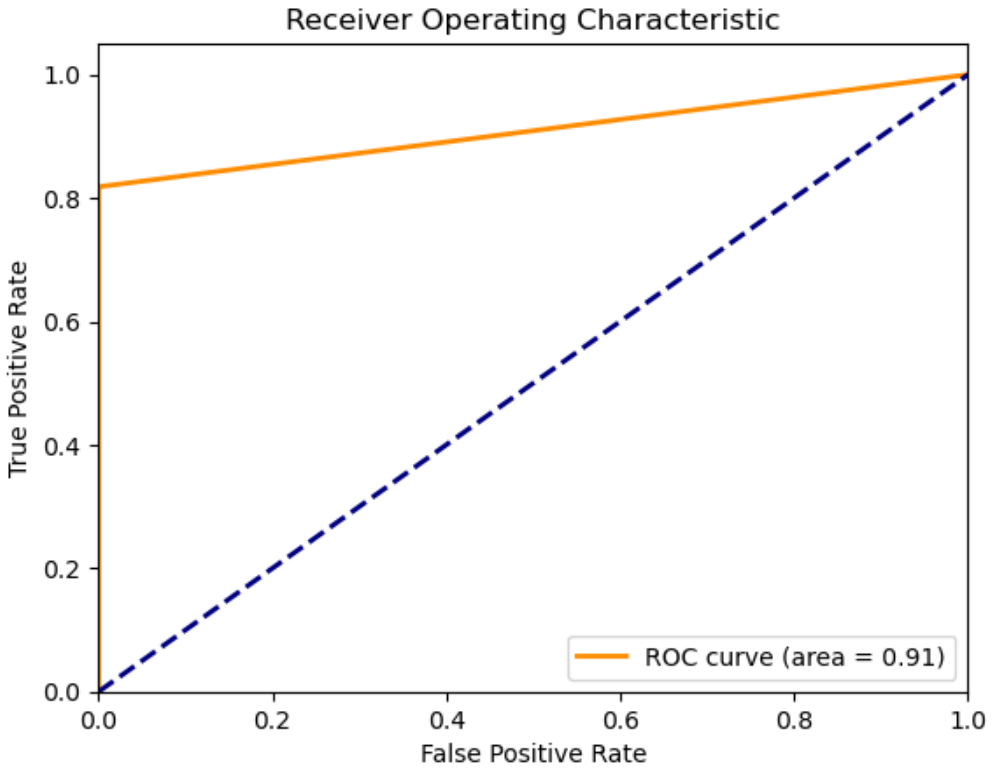
```
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

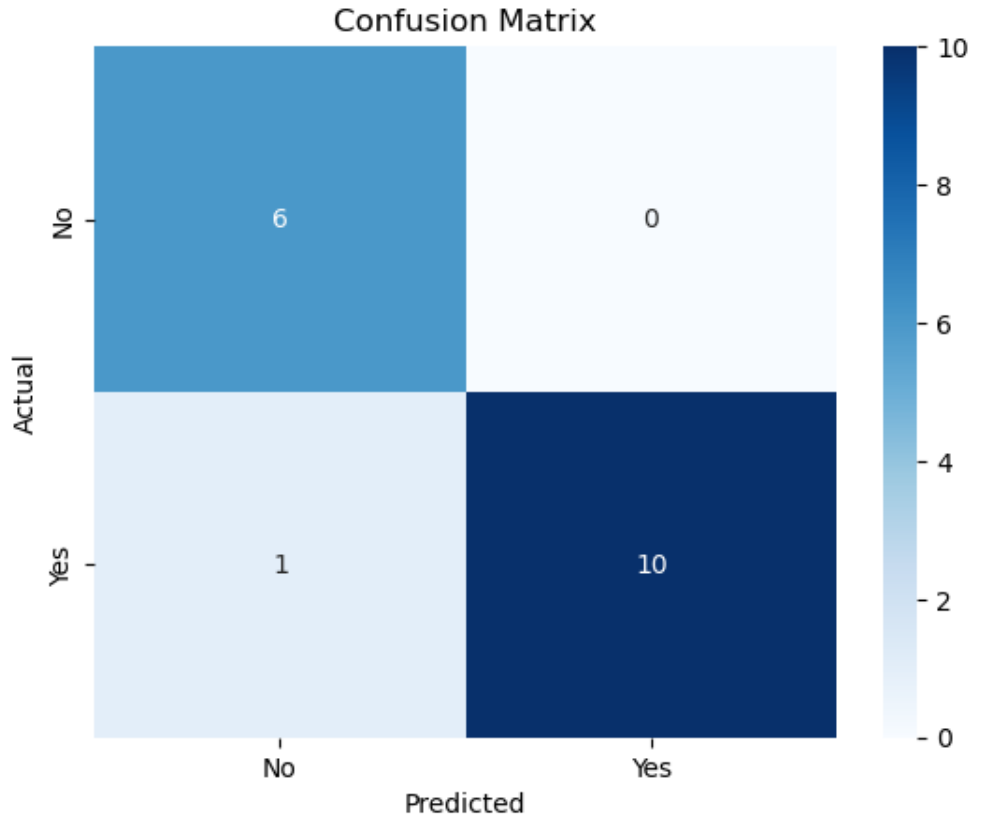
Decision Tree Codes:

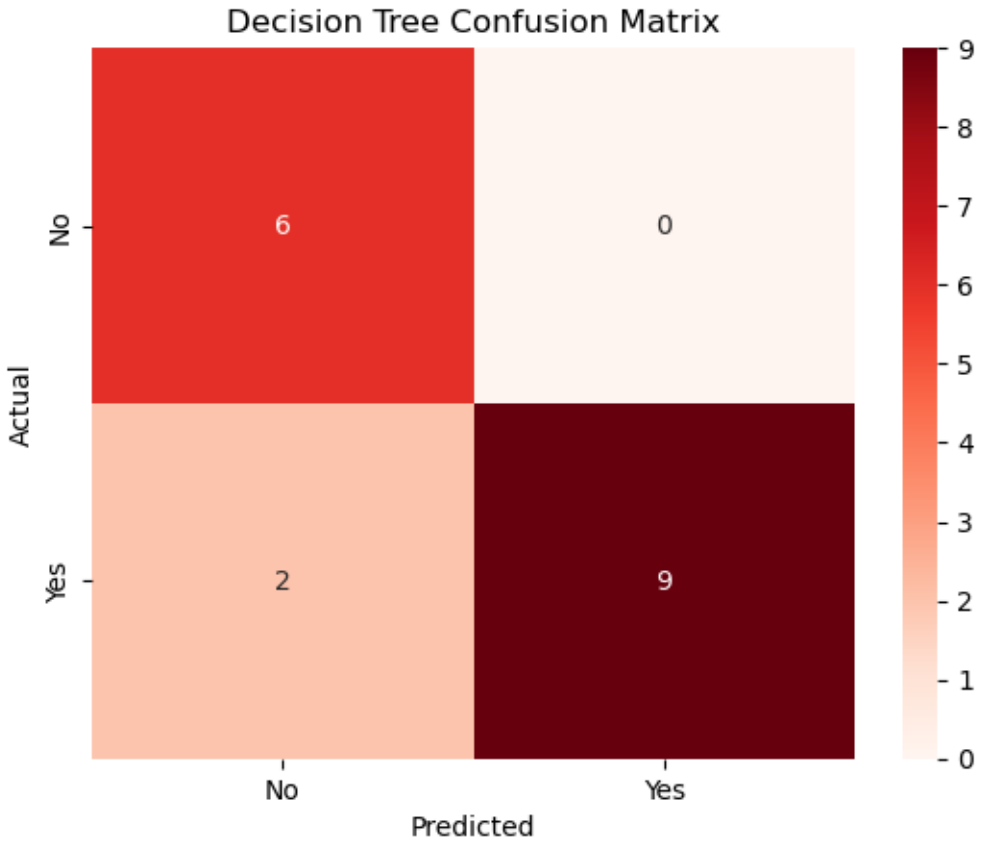
```
from sklearn.tree import DecisionTreeClassifier

# Fit the decision tree model
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)

# Predict on the test set
y_pred_tree = tree_model.predict(X_test)
y_prob_tree = tree_model.predict_proba(X_test)[:, 1]

# Evaluate the model
conf_matrix_tree = confusion_matrix(y_test, y_pred_tree)
roc_auc_tree = roc_auc_score(y_test, y_prob_tree)

import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, auc

# Assuming y_test and y_scores are your ground truth labels and predicted scores
fpr, tpr, thresholds = roc_curve(y_test, y_pred_tree)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()


# Display the confusion matrix
sns.heatmap(conf_matrix_tree, annot=True, fmt='d', cmap='Reds', xticklabels=['No', 'Yes'],
yticklabels=['No', 'Yes'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Decision Tree Confusion Matrix')
plt.show()
```

2. Result:


ROC Curve
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Logistic Regression (AUC = 1.00)


Receiver Operating Characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC curve (area = 0.91)


Confusion Matrix
6
0
1
10
No
Yes
Actual
No
Yes
Predicted
10
8
6
4
2
0


Decision Tree Confusion Matrix
6
0
2
9
No
Yes
Actual
No
Yes
Predicted
9
8
7
6
5
4
3
2
1
0

3. Interpretation:

The ROC curve for the logistic regression model shows an Area Under the Curve (AUC) of 1.00, indicating a perfect model with no false positives or false negatives. This suggests that the logistic regression model has classified all responses correctly for the given dataset. Whereas, when coming to the Decision Tree Model ROC curve for the decision tree model shows an AUC of 0.91, indicating a high but not perfect level of performance. This suggests that while the decision tree model performs well in distinguishing between customers who responded positively and those who did not, it has some misclassifications. Logistic regression typically weights features linearly. Given its perfect performance, it suggests that the combination of customer demographics provided in the dataset is highly predictive of the response. Decision trees capture non-linear relationships and interactions between features. An AUC of 0.91 suggests that the decision tree has identified significant patterns and interactions in the data, but there is still room for improvement.

Confusion Matrix:

The Logistic Regression model (blue confusion matrix) has 6 True Negatives (TN), 10 True Positives (TP), 0 False Positives (FP), and 1 False Negative (FN). This results in high precision (no false positives) and a recall rate of approximately 91% (10 out of 11 actual positives correctly identified). This model demonstrates strong performance in correctly identifying positive responses while maintaining no false alarms for negative responses.

In contrast, the Decision Tree model (red confusion matrix) shows 6 True Negatives (TN), 9 True Positives (TP), 0 False Positives (FP), and 2 False Negatives (FN). The precision remains high (no false positives), but the recall drops to approximately 82% (9 out of 11 actual positives correctly identified). While the Decision Tree model also avoids false positives, it is slightly less effective in capturing positive responses compared to the Logistic Regression model.

In summary, both models are highly precise, but the Logistic Regression model outperforms the Decision Tree model in terms of recall. Therefore, if the primary objective is to maximize the identification of positive responses (minimizing false negatives), the Logistic Regression model is the better choice.

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

1. Code:

```
import warnings
from statsmodels.tools.sm_exceptions import PerfectSeparationWarning
from statsmodels.tools.sm_exceptions import ConvergenceWarning

# Suppress PerfectSeparationWarning
warnings.filterwarnings('ignore', category=PerfectSeparationWarning)

# Suppress ConvergenceWarning
warnings.filterwarnings('ignore', category=ConvergenceWarning)

# Convert the target variable to binary based on the specified condition
subset_data['chicken_q'] = subset_data['chicken_q'].apply(lambda x: 0 if x < 1 else
1)

# Define the independent variables (example columns, update based on your dataset)
# Assuming 'Age', 'Income', 'Education' are some of the features in the dataset
independent_vars = ['Age', 'Marital_Status', 'Education']


# Add a constant term for the intercept
X = sm.add_constant(subset_data[independent_vars])


# Define the dependent variable
y = subset_data['chicken_q']


# Fit the probit regression model
probit_model = Probit(y, X).fit()


# Print the summary of the model
print(probit_model.summary())


# Make predictions
subset_data['predicted'] = probit_model.predict(X)


# Display the first few rows with the predictions
print(data.head())
```

2. Result:

```
Optimization terminated successfully.
         Current function value: 0.115600
         Iterations 7
                          Probit Regression Results
==============================================================================
Dep. Variable:              chicken_q   No. Observations:               101662
Model:                         Probit   Df Residuals:                   101658
Method:                           MLE   Df Model:                            3
Date:                Mon, 01 Jul 2024   Pseudo R-squ.:                 0.01405
Time:                        17:23:09   Log-Likelihood:                -11752.
converged:                       True   LL-Null:                       -11920.
Covariance Type:            nonrobust   LLR p-value:                2.615e-72
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -2.2494      0.054    -41.604      0.000      -2.355      -2.143
Age            0.0015      0.001      2.205      0.027       0.000       0.003
Marital_Status -0.0337     0.023     -1.483      0.138      -0.078       0.011
Education      0.0420      0.002     17.326      0.000       0.037       0.047
==============================================================================
```

3. Interpretation:

Here, the dependent variable chicken_q indicates whether an individual is a non-vegetarian (1) or not (0). The independent variables are Age, Marital_Status, and Education. The coefficient of -2.2494 suggests that the baseline log-odds of being a non-vegetarian when all independent variables are zero, so the baseline probability of being a non-vegetarian is low. Age has a small but significant positive effect on the probability of being a non-vegetarian. As age increases by one year, the probability of being a non-vegetarian slightly increases. Marital status does not significantly affect the probability of being a non-vegetarian since the p-value is greater than 0.05. Education has a significant positive effect on the probability of being a non-vegetarian. Higher levels of education are associated with a higher probability of being a non-vegetarian.

- Pseudo R-squared: 0.01405 (indicates that approximately 1.4% of the variance in the dependent variable is explained by the independent variables)
- LLR p-value: 2.615e-72 (indicates that the model as a whole is statistically significant)

In summary, the probit regression model identifies that age and education have significant effects on the likelihood of being a non-vegetarian, with education having a stronger positive effect. The model overall is statistically significant, but it explains a small portion of the variance in non-vegetarianism, indicating that other factors may also play an important role.

c) Perform a Tobit regression analysis on "NSSO68.csv" discuss the results and explain the real-world use cases of Tobit model.

1. Code:

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from statsmodels.base.model import GenericLikelihoodModel

# Convert the target variable to binary based on the specified condition
df['MPCE_URP'] = df['MPCE_URP'].apply(lambda x: 0 if x < 380 else 1)

# Define the independent variables (X) and the dependent variable (y)
X = df[['Whether_owns_any_land', 'hhdsz', 'Religion','Social_Group','Regular_salary_earner']]  #
replace with your actual column names
y = df['MPCE_URP']  # replace with your actual column name

# Add a constant term for the intercept
X = sm.add_constant(X)

# Define the Tobit model class
class Tobit(GenericLikelihoodModel):
    def __init__(self, endog, exog, left=0, right=np.inf, **kwargs):
        super(Tobit, self).__init__(endog, exog, **kwargs)
        self.left, self.right = left, right

    def nloglikeobs(self, params):
        exog = self.exog
        endog = self.endog
        left, right = self.left, self.right

        beta = params[:-1]
        sigma = params[-1]

        XB = np.dot(exog, beta)
        cens = (endog == left) * (left != -np.inf) + (endog == right) * (right != np.inf)
        uncens = 1 - cens

        ll = np.zeros(len(endog))

        ll[cens] = np.log(
            (1 / (np.sqrt(2 * np.pi) * sigma)) *
            np.exp(-((endog[cens] - XB[cens]) ** 2) / (2 * sigma ** 2))
        )

        ll[uncens] = np.log(
            (1 / (np.sqrt(2 * np.pi) * sigma)) *
            np.exp(-((endog[uncens] - XB[uncens]) ** 2) / (2 * sigma ** 2))
        )
```

```
        return -ll

    def fit(self, start_params=None, maxiter=10000, maxfun=5000, **kwargs):
        if start_params is None:
            start_params = np.append(np.zeros(self.exog.shape[1]), 1)
        return super(Tobit, self).fit(start_params=start_params,
                                      maxiter=maxiter, maxfun=maxfun, **kwargs)

# Fit the Tobit model
tobit_model = Tobit(y, X)
tobit_results = tobit_model.fit()

# Print the summary of the model
print(tobit_results.summary())
```

2. Result:

```
Optimization terminated successfully.
         Current function value: -0.003281
         Iterations: 223
         Function evaluations: 362
                            Tobit Results
==============================================================================
Dep. Variable:               MPCE_URP   Log-Likelihood:                333.38
Model:                          Tobit   AIC:                           -652.8
Method:            Maximum Likelihood   BIC:                           -586.1
Date:                Mon, 01 Jul 2024
Time:                        17:23:18
No. Observations:              101624
Df Residuals:                  101618
Df Model:                           5
==============================================================================
                             coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const                     -0.0023      0.057     -0.041      0.967      -0.114       0.110
Whether_owns_any_land     -0.0016      0.018     -0.088      0.930      -0.036       0.033
hhdsz                     -0.0016      0.002     -0.862      0.389      -0.005       0.002
Religion                   0.0026      0.004      0.688      0.491      -0.005       0.010
Social_Group               0.0050      0.002      2.204      0.027       0.001       0.009
Regular_salary_earner      0.0018      0.024      0.072      0.943      -0.046       0.050
par0                       0.0803      0.004     20.898      0.000       0.073       0.088
==============================================================================
```

3. Interpretation:

The intercept of the Tobit model. In your case, it's very close to zero (-0.0023), indicating that when all independent variables are zero, the expected value of the dependent variable (MPCE_URP) is very low. Variables like Whether_owns_any_land, hhdsz, and Religion show coefficients with large standard errors and non-significant p-values, indicating they have no significant impact on MPCE_URP.

Social_Group emerges as statistically significant (coef = 0.0050, p = 0.027), suggesting membership in certain social groups may influence MPCE_URP positively. Regular_salary_earner does not significantly affect MPCE_URP (coef = 0.0018, p = 0.943). The parameter par0 is significant (coef = 0.0803, p < 0.001), indicating its relevance to the model's formulation. Overall, this Tobit model highlights the nuanced effects of social group membership on MPCE_URP while indicating other variables examined lack statistically significant impact.

RESULTS AND INTERPRETATION

d) Conduct a logistic regression analysis on your assigned dataset. Validate assumptions, evaluate with a confusion matrix and ROC curve, and interpret the results. Then, perform a decision tree analysis and compare it to the logistic regression. [campaign_responses]

1. Code:

```
# Read the data
df <- read.csv('campaign_responses.csv')

# Remove rows with missing values
df_clean <- na.omit(df)

# Convert "yes" to 1 and "no" to 0
df_clean$responded <- ifelse(df_clean$responded == "yes", 1, 0)

# Split the data into features (X) and target variable (y)
X <- df_clean %>% select(-responded)
y <- df_clean$responded

# Split the data into training and testing sets
set.seed(42)

# Determine the indices for the training set
train_indices <- sample(seq_len(nrow(X)), size = 0.8 * nrow(X))

# Split the data into training and testing sets
X_train <- X[train_indices, ]
X_test <- X[-train_indices, ]
y_train <- y[train_indices]
y_test <- y[-train_indices]
```

```
# Fit logistic regression model
logistic_model <- glm(responded ~ ., data = cbind(X_train, responded = y_train), family = 'binomial')
# Print summary of the logistic regression model
print(summary(logistic_model))
```

2. Result:

```
Call:
glm(formula = responded ~ ., family = "binomial", data = cbind(X_train,
    responded = y_train))

Deviance Residuals:
       Min          1Q      Median          3Q         Max
-2.409e-06  -2.409e-06  -2.409e-06  -2.409e-06  -2.409e-06

Coefficients: (1 not defined because of singularities)
                      Estimate Std. Error z value Pr(>|z|)
(Intercept)         -2.657e+01  3.391e+06       0        1
customer_id          5.229e-16  3.956e+03       0        1
age                  2.945e-16  5.392e+04       0        1
genderMale           3.851e-14  3.420e+05       0        1
annual_income        4.187e-18  2.858e+01       0        1
credit_score        -1.306e-15  5.961e+03       0        1
employedYes          2.940e-14  2.444e+05       0        1
marital_statusSingle        NA         NA      NA       NA
no_of_children      -3.434e-14  1.587e+05       0        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 0.0000e+00  on 43  degrees of freedom
Residual deviance: 2.5527e-10  on 36  degrees of freedom
AIC: 16

Number of Fisher Scoring iterations: 25
```

3. Interpretation:

Other Coefficients (customer_id, age, gender, annual_income, credit_score, employed, no_of_children) show that the estimates for these coefficients are extremely small (close to zero) with large standard errors, and all have p-values of 1. This indicates that these predictors do not contribute significantly to predicting the response variable (responded), which contradicts typical expectations. marital_statusSingle coefficient is not defined (NA), likely due to perfect multicollinearity or separation with other variables in the dataset, which means that the variable has to be removed. The null deviance is 0, suggesting that the model perfectly predicts the response variable with the intercept alone, which is highly unusual and indicative of potential model specification issues.

e) Perform a probit regression on "NSSO68.csv" to identify non-vegetarians. Discuss the results and explain the characteristics and advantages of the probit model

1. Code:

```
# Perform a probit regression on "NSSO68.csv" to identify non-vegetarians.

# Load the dataset
data_nss <- read.csv("NSSO68.csv")
# Create a binary variable for chicken consumption
data_nss$chicken_q <- ifelse(data_nss$chicken_q > 0, 1, 0)

# Verify the creation of 'chicken_binary'
table(data_nss$chicken_q)

# Probit regression model
probit_model <- glm(chicken_q ~ Age + Marital_Status + Education, data = data_nss, family =
binomial(link = "probit"))

# Summary of the probit regression model
summary(probit_model)
```

2. Result:

```
Call:
glm(formula = chicken_q ~ Age + Marital_Status + Education, family = binomial(link
= "probit"),
    data = data_nss)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.0937  -1.0196  -0.9963   1.3416   1.4154

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.3307564  0.0255427 -12.949  < 2e-16 ***
Age            -0.0006173  0.0003157  -1.956  0.05052 .
Marital_Status  0.0341802  0.0107511   3.179  0.00148 **
Education       0.0068008  0.0011195   6.075 1.24e-09 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 137097  on 101652  degrees of freedom
Residual deviance: 137053  on 101649  degrees of freedom
```

```
  (9 observations deleted due to missingness)
AIC: 137061

Number of Fisher Scoring iterations: 4
```

3. Interpretation:

The model suggests that age, marital status, and education level are significant predictors of chicken consumption behavior. Specifically, being married and having higher education levels are associated with higher probabilities of consuming chicken. In this case, they range between -1.0937 and 1.4154, suggesting that the model fits the data reasonably well. The residual deviance of 137053 on 101649 degrees of freedom indicates that the model explains a significant portion of the variability in chicken consumption behavior. Overall, the logistic regression analysis provides valuable insights into the factors influencing chicken consumption behavior among the studied population.

f) Perform a Tobit regression analysis on "NSSO68.csv" discuss the results and explain the real-world use cases of Tobit model.

1. Code:

```
# Define the independent variables (X) and the dependent variable (y)
X <- df %>%
  select(Whether_owns_any_land, hhdsz, Religion, Social_Group, Regular_salary_earner)
X <- cbind(1, X)  # Add a constant term for the intercept
y <- df$MPCE_URP

# Define the Tobit model function
tobit_loglike <- function(params) {
  beta <- params[1:(length(params)-1)]
  sigma <- params[length(params)]
  XB <- as.matrix(X) %*% beta
  cens <- (y == 0) + (y == 1)
  uncens <- 1 - cens
  ll <- numeric(length(y))

  ll[cens == 1] <- log(dnorm(y[cens == 1], mean = XB[cens == 1], sd = sigma))
  ll[uncens == 1] <- log(dnorm(y[uncens == 1], mean = XB[uncens == 1], sd = sigma))

  return(-sum(ll))
```

```
}

# Initial parameter guesses
start_params <- c(rep(0, ncol(X)), 1)

# Fit the Tobit model
tobit_results <- maxLik(tobit_loglike, start = start_params, method = "BFGS")

# Print the summary of the model
summary(tobit_results)
```