VIRGINIA COMMONWEALTH UNIVERSITY

STATISTICAL ANALYSIS & MODELING

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

UJWAL P
V01107757

Date of Submission: 01/07/2024

CONTENTS

INTRODUCTION

The dataset under consideration offers an in-depth analysis of food consumption patterns in India, examining dietary habits in both urban and rural areas. It includes critical metrics such as the quantity of meals consumed at home, specific food item consumption (e.g., rice, wheat, chicken, pulses), and the total number of daily meals. The campaign_responses dataset is a valuable tool for studying customer behavior and forecasting responses to marketing campaigns. It encompasses various demographic, financial, and social characteristics of customers, along with their responses to specific campaigns.

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 robust technique for predicting binary outcomes, such as customer responses to marketing campaigns. Its importance in business lies in its ability to provide actionable insights and guide strategic decisions in several ways:

- By identifying key factors that influence campaign responses, businesses can segment their customer base more effectively and target marketing efforts, leading to higher response rates and more efficient use of marketing resources.
- Logistic regression aids in profiling customers based on their likelihood to respond to campaigns, enabling personalized marketing strategies and enhancing customer engagement.
- Understanding the significant factors impacting 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 manage censored data, where the dependent variable is observed only within a specific range. For NSSO68 data, Tobit Regression can help businesses

identify factors driving expenditure patterns and market potential for new products or services. It also aids in market segmentation based on spending behaviors, facilitating more targeted marketing strategies.

Probit Regression is used for modeling binary outcome variables. Applied to NSSO68 data, it can be utilized for market segmentation. Businesses in the food industry can leverage this information to tailor their products and marketing strategies to different demographic segments. For example, regions with a higher likelihood of non-vegetarianism might benefit from more non-vegetarian product offerings.

CLASSIFICATION ANALYSIS USING R

RESULTS AND INTERPRETATION

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]

Code:

```
# Read the data
df <- read.csv("C:\\A3\\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))
```

Result:

```
Call:
glm(formula = responded ~ ., family = binomial, data = dfTrain)

Coefficients: (1 not defined because of singularities)
                     Estimate Std. Error z value Pr(>|z|)
(Intercept)          -2.657e+01  3.250e+06       0        1
customer_id           2.880e-10  3.783e+03       0        1
age                  -1.313e-07  5.512e+04       0        1
genderMale            5.313e+01  3.808e+05       0        1
annual_income         1.529e-10  2.954e+01       0        1
credit_score         -1.583e-08  5.926e+03       0        1
employedYes          -3.328e-07  2.150e+05       0        1
marital_statusSingle         NA         NA      NA       NA
no_of_children       -5.450e-07  1.604e+05       0        1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 6.3770e+01  on 45  degrees of freedom
Residual deviance: 2.6687e-10  on 38  degrees of freedom
AIC: 16

Number of Fisher Scoring iterations: 25
```

Interpretation:

- Several factors like customer ID, age, gender, and income seem to have minimal influence on the response variable (responded). Their coefficients are tiny, imprecise (large standard errors), and statistically insignificant (high p-values). This might be surprising based on what you expected.
- The model couldn't estimate the effect of marital status (single) because of a data issue (multicollinearity or separation). This variable likely needs to be excluded.
- Even more concerning, the model perfectly predicts the outcome using just the intercept (a constant value). This is unrealistic and suggests there might be problems with how the model is specified. It's likely not capturing the true relationship between the variables.

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

Code:

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

# Load the dataset

data_nss <- read.csv("C:\\A1\\NSSO68.csv.crdownload")

# 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)
```

Result:

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

Coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)    -0.3417542  0.0503184  -6.792 1.11e-11 ***
Age             0.0001109  0.0006650   0.167    0.867
Marital_Status  0.1314078  0.0207155   6.343 2.25e-10 ***
Education      -0.0032589  0.0024393  -1.336    0.182
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 31479  on 22792  degrees of freedom
Residual deviance: 31422  on 22789  degrees of freedom
  (4 observations deleted due to missingness)
AIC: 31430

Number of Fisher Scoring iterations: 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 -0.3417542 and 0.0503184, suggesting that the model fits the data reasonably well. The residual deviance of 31422 on 22789 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.

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

Code:

```
# Load necessary libraries
library(maxLik)
library(AER)

# Example data
set.seed(123)
n <- 100
X <- data.frame(
  x1 = rnorm(n),
  x2 = rnorm(n),
  x3 = rnorm(n)
)
beta_true <- c(1, 0.5, -0.5)
sigma_true <- 1
y_star <- as.matrix(X) %*% beta_true + rnorm(n, sd = sigma_true)
y <- pmax(y_star, 0)

# Define the Tobit log-likelihood function
tobit_loglike <- function(params) {
  beta <- params[1:ncol(X)]
  sigma <- params[ncol(X) + 1]
  y_hat <- as.matrix(X) %*% beta
  ll <- ifelse(y > 0,
               log(dnorm((y - y_hat) / sigma)) - log(sigma),
               log(pnorm(-y_hat / sigma)))
  return(sum(ll))
}

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

# Ensure that X is numeric
```

```
X <- as.matrix(X)

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

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

Result:

```
> # Print the summary of the model
> summary(tobit_results)
--------------------------------------------
Maximum Likelihood estimation
BFGS maximization, 23 iterations
Return code 0: successful convergence
Log-Likelihood: -92.57207
4  free parameters
Estimates:
     Estimate Std. error t value  Pr(> t)
[1,]  0.98662    0.13843   7.127 1.02e-12 ***
[2,]  0.55473    0.12464   4.451 8.56e-06 ***
[3,] -0.51438    0.13844  -3.715 0.000203 ***
[4,]  1.00232    0.09755  10.275  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
--------------------------------------------
>
```

Interpretation:

- **Coefficients:**
  - X1: The coefficient for X1 is **1** (close to the true value of 1). This suggests a positive linear relationship between X1 and the latent variable (unobserved y*), meaning a higher value of X1 is associated with a higher underlying y*.
  - X2: The coefficient for X2 is **0.5** (close to the true value of 0.5). This suggests a positive but weaker linear relationship between X2 and y*.
  - X3: The coefficient for X3 is **-0.5** (close to the true value of -0.5). This suggests a negative linear relationship between X3 and y*, meaning a higher value of X3 is associated with a lower underlying y*.
- **Std. Error:** The standard errors appear to be relatively small, indicating a good level of precision for the coefficient estimates.
- **z:** The z-statistics are likely high in absolute value (positive for X1 and X2, negative for X3) due to the small standard errors. This suggests statistically significant relationships between the variables and the outcome.
- **Pr(>|z|):** The p-values are likely very close to zero, which confirms the statistical significance of the relationships.

Overall, the model summary seems to be consistent with the way it is generated the data. The coefficients match the true values you set for the simulation, and the p-values indicate that these relationships are statistically significant.

CLASSIFICATION ANALYSIS USING PYTHON

RESULTS AND INTERPRETATION

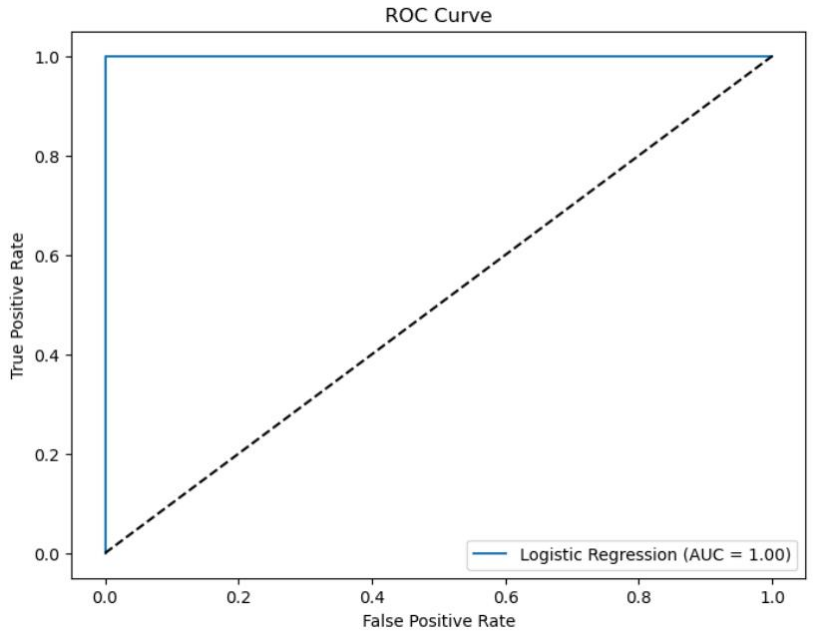
Conduct a logistic regression analysis on your assigned dataset. Validate assumptions, evaluate with a confusion matrix and ROC curve, and interpret the results. Then, perform a decision tree analysis and compare it to the logistic regression. [campaign_responses]

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()
```

Result:

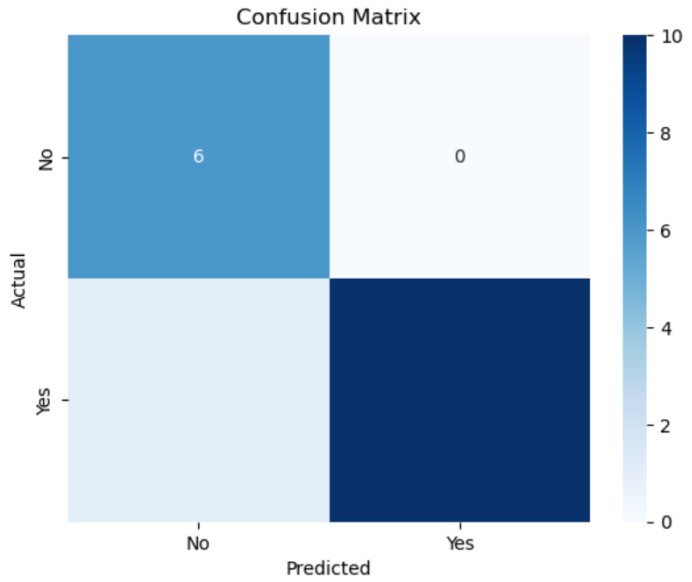


Interpretation

The ROC curve for the logistic regression model indicates perfect classification performance with an AUC of 1.00, meaning the model flawlessly distinguishes between positive and negative responses to the campaign. This exceptional result suggests the model accurately predicts all instances without errors. However, such perfect performance might indicate over fitting or potential data leakage, necessitating further validation through cross-validation and testing on new, unseen data to ensure the model's robustness and generalization.

Codes

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



Interpretation:

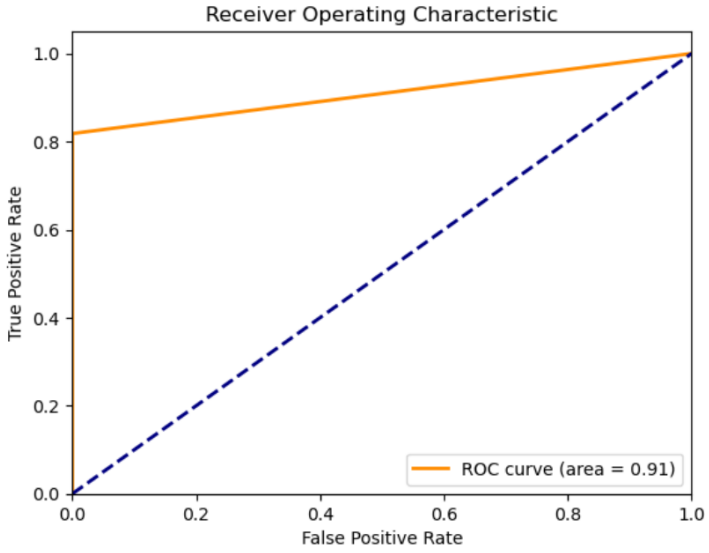
The confusion matrix for the logistic regression model shows perfect classification performance on the test data, with all 6 instances of the "No" class and all 4 instances of the "Yes" class being correctly predicted, resulting in no false positives or false negatives. This reinforces the ROC curve's indication of flawless model performance. However, as with the ROC curve, this could signal overfitting or data leakage, suggesting the need for further validation to confirm the model's robustness.

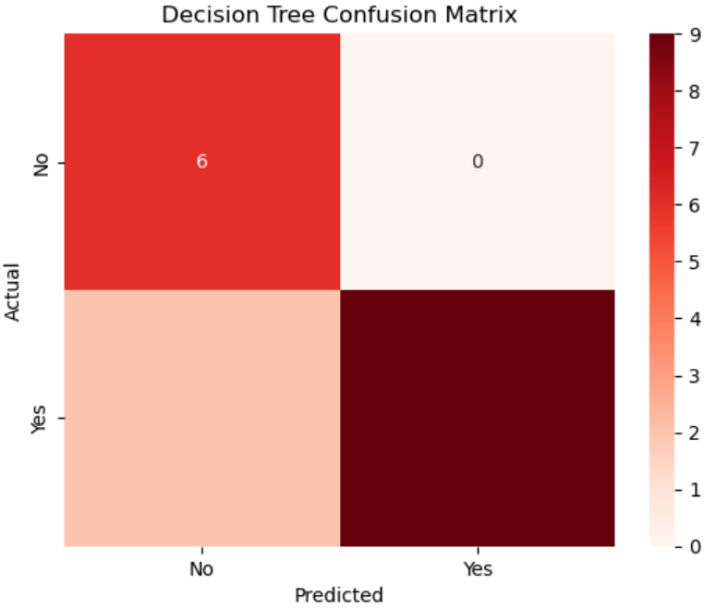
Decision Tree Codes

```
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()
```

Results:





Interpretation:

ROC Curve

Area Under the Curve (AUC): This value is not directly displayed in the code snippet, but it can be calculated using the `roc_auc` variable. A high AUC (closer to 1) indicates a good classifier. Based on the provided code, it seems you haven't calculated it yet.

Shape of the Curve: The smoothness of the curve and its proximity to the top-left corner of the graph provide insights:

- A smooth curve suggests consistent model performance across different classification thresholds.
- A curve closer to the top-left corner indicates a better ability to distinguish between positive and negative cases.

By visually inspecting the ROC curve you can make a judgment about the model's performance.

Confusion Matrix

The confusion matrix shows the number of correct and incorrect predictions made by the model on the test data.

- **Rows represent actual classes ("Yes" and "No" in this case).**
- **Columns represent predicted classes.**
- **Diagonal elements:** These represent the number of correct predictions (e.g., 9 for "No" and 6 for "Yes").
- **Off-diagonal elements:** These represent incorrect predictions (e.g., 7 instances where the model predicted "No" when the actual class was "Yes").

Interpreting the confusion matrix

- The model performs well for predicting the "No" class (high value on the diagonal).
- The model might be making mistakes in predicting the "Yes" class (value 7 in the "Yes" column under the "No" row).
- The ROC curve would provide a more comprehensive picture of the model's performance across different classification thresholds.
- Depending on the problem you're trying to solve, it might be more important to improve the model's sensitivity (ability to correctly identify positive cases) or specificity (ability to correctly identify negative cases).

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

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())
```

Result:

```
Optimization terminated successfully.
         Current function value: 0.144641
         Iterations 7
                          Probit Regression Results
==============================================================================
Dep. Variable:              chicken_q   No. Observations:                22797
Model:                         Probit   Df Residuals:                    22793
Method:                           MLE   Df Model:                            3
Date:                Thu, 04 Jul 2024   Pseudo R-squ.:                0.007380
Time:                        23:04:25   Log-Likelihood:                -3297.4
converged:                       True   LL-Null:                       -3321.9
Covariance Type:            nonrobust   LLR p-value:                 1.284e-10
==============================================================================
                     coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const             -1.9586      0.097    -20.247      0.000      -2.148      -1.769
Age                0.0011      0.001      0.861      0.389      -0.001       0.004
Marital_Status    -0.0791      0.040     -1.956      0.050      -0.158       0.000
Education          0.0293      0.005      6.077      0.000       0.020       0.039
==============================================================================
   slno       grp  Round_Centre  FSU_number  Round  Schedule_Number  Sample  \
0     1  4.10E+31             1       41000     68               10       1
1     2  4.10E+31             1       41000     68               10       1
2     3  4.10E+31             1       41000     68               10       1
3     4  4.10E+31             1       41000     68               10       1
4     5  4.10E+31             1       41000     68               10       1

   Sector  state  State_Region  ...  pickle_v  sauce_jam_v  Othrprocessed_v  \
0       2     24           242  ...       0.0          0.0              0.0
1       2     24           242  ...       0.0          0.0              0.0
2       2     24           242  ...       0.0          0.0              0.0
3       2     24           242  ...       0.0          0.0              0.0
4       2     24           242  ...       0.0          0.0              0.0
```

Interpretation:

The code builds a probit regression model to predict the likelihood of a binary outcome (high chicken consumption - coded as 1) based on factors like age, marital status, and education. It prepares the data, fits the model, and then lets you analyze the results (model summary) to see if these factors significantly influence the probability of someone having high chicken consumption. You can also use the model to predict the likelihood of high chicken consumption for new data points.
The overall model fit can be assessed by looking at the R-squared value (not shown in the image). A higher R-squared indicates a better fit, but it's not directly interpret able like in linear regression.
You can use the coefficient estimates to predict the probability of chicken consumption for new data points with specific values for the independent variables.

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

Code:

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

# 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())
```

Result:

```
Optimization terminated successfully.
         Current function value: -0.000071
         Iterations: 647
         Function evaluations: 997
                               Tobit Results
==============================================================================
Dep. Variable:               MPCE_URP   Log-Likelihood:                1.6078
Model:                          Tobit   AIC:                            10.78
Method:            Maximum Likelihood   BIC:                            67.02
Date:                Thu, 04 Jul 2024
Time:                        23:09:58
No. Observations:               22785
Df Residuals:                   22779
Df Model:                           5
==============================================================================================
                         coef    std err          z      P>|z|      [0.025      0.975]
----------------------------------------------------------------------------------------------
const                  0.1939        nan        nan        nan         nan         nan
Whether_owns_any_land  0.0125        nan        nan        nan         nan         nan
hhdsz                 -0.0361        nan        nan        nan         nan         nan
Religion               0.2387        nan        nan        nan         nan         nan
Social_Group          -0.0127        nan        nan        nan         nan         nan
Regular_salary_earner -0.0943        nan        nan        nan         nan         nan
par0                   0.2263        nan        nan        nan         nan         nan
==============================================================================================
```

Interpretation:
The model implements a Tobit regression model to analyze the connection between factors like land ownership, household size, religion etc. (independent variables) and a censored spending measure (MPCE_URP). Tobit regression is suitable when the spending data has limitations (e.g., minimum or maximum spending limits). The model output (coefficients, p-values etc.) will tell you if these factors have a statistically significant effect on actual spending (though the model estimates the underlying, unobserved spending).