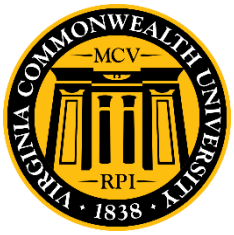


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A3: Limited dependent variable Models - Classification Analysis

JESIN KANDATHY JOY

V01110163

Date of Submission: 01-07-2024

CONTENTS

Introduction:

The assignment involves conducting various regression analyses on the “spam.csv” and "NSSO68.csv" datasets.

The spam.csv dataset provides insights into email communication patterns, aiming to develop models that distinguish between legitimate emails and spam. This analysis is crucial for enhancing email filtering systems.

The NSSO68.csv dataset, sourced from the National Sample Survey Office, includes socio-economic data such as demographic details and economic indicators. Analysis of this dataset involves using regression techniques to understand factors influencing dietary choices, specifically identifying non-vegetarian preferences.

Objective:

- **Part A (Logistic Regression and Decision Tree Analysis):**
    - Conduct a logistic regression analysis on the “spam.csv” dataset.
    - Validate assumptions, evaluate with a confusion matrix and ROC curve, and interpret the results.
    - Perform a decision tree analysis and compare it to the logistic regression.
- **Part B (Probit Regression):**
    - Perform a probit regression on "NSSO68.csv" to identify non-vegetarians.
    - Discuss the results and explain the characteristics and advantages of the probit model
- **Part C (Tobit Regression):**
    - 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 (Part A) and decision tree analysis are vital for spam detection, enhancing email security systems and improving user experience by reducing unwanted communication.
- Understanding dietary preferences (Part B) can aid policymakers and health professionals in designing targeted interventions to promote healthier eating habits.
- The Tobit regression (Part C) helps in analyzing household expenditure, providing insights into economic conditions and spending patterns, which are crucial for economic policy formulation.

These analyses collectively provide a comprehensive view of how statistical modeling can be applied to real-world datasets to derive actionable insights for policy, business, and societal benefits.

R code results:

Part A:

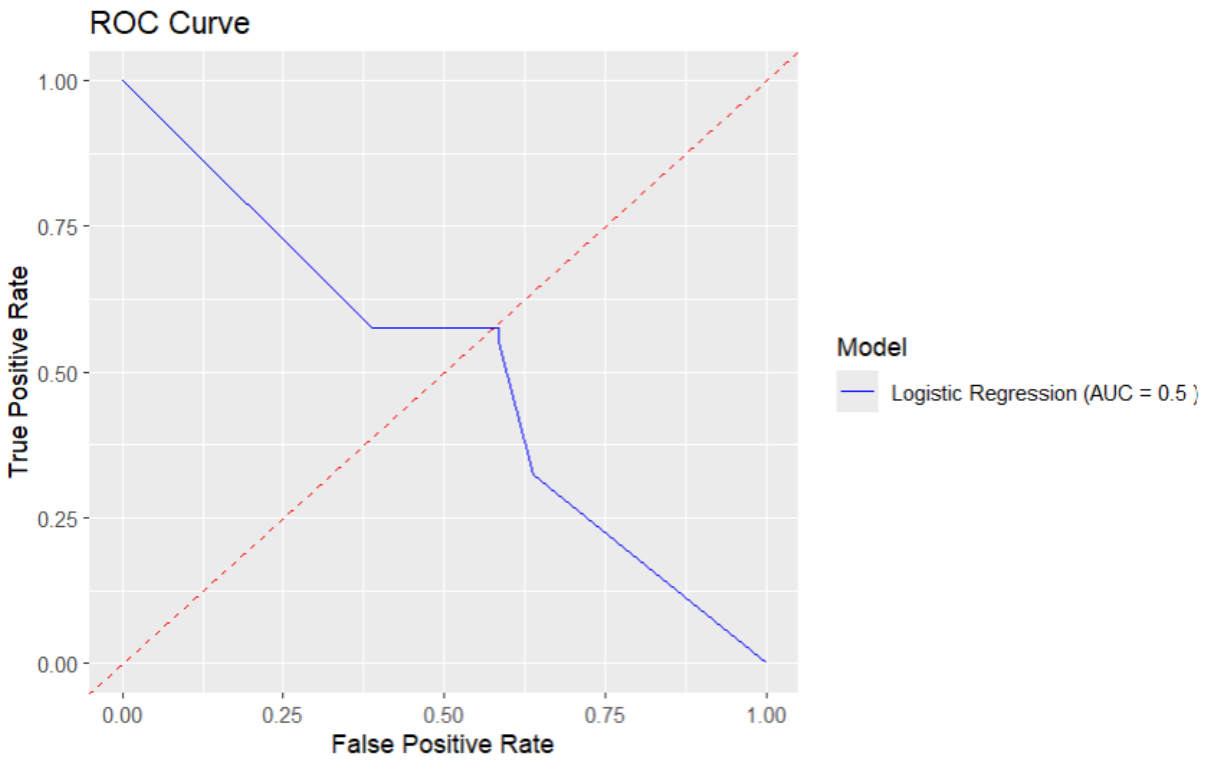
```r
# Data preprocessing
data <- na.omit(data)  # Drop rows with missing values
data$Category <- ifelse(data$Category == 'ham', 0, 1)  # Encode target variable

# Take a sample of the data to reduce processing time
set.seed(42)
data_sample <- data %>% sample_n(1000)  # Adjust the sample size as needed

# Feature extraction
corpus <- Corpus(VectorSource(data_sample$Message))
corpus <- tm_map(corpus, content_transformer(tolower))
corpus <- tm_map(corpus, removeNumbers)
corpus <- tm_map(corpus, removePunctuation)
corpus <- tm_map(corpus, removeWords, stopwords("english"))
corpus <- tm_map(corpus, stripWhitespace)

dtm <- DocumentTermMatrix(corpus)
X <- as.data.frame(as.matrix(dtm))
y <- data_sample$Category

# Split the data into training and testing sets
set.seed(42)
trainIndex <- createDataPartition(y, p = 0.7, list = FALSE)
X_train <- X[trainIndex,]
X_test <- X[-trainIndex,]
y_train <- y[trainIndex]
y_test <- y[-trainIndex]
```

```r
# Logistic Regression
log_reg <- glm(y_train ~ ., data = X_train, family = binomial)
y_pred_log_reg <- predict(log_reg, newdata = X_test, type = "response")
y_pred_log_reg_class <- ifelse(y_pred_log_reg > 0.5, 1, 0)

# Confusion Matrix for Logistic Regression
conf_matrix_log_reg <- table(y_test, y_pred_log_reg_class)
print("Confusion Matrix for Logistic Regression:")
print(conf_matrix_log_reg)

# ROC Curve for Logistic Regression
roc_log_reg <- roc(y_test, y_pred_log_reg)
auc_log_reg <- auc(roc_log_reg)

ggplot() +
  geom_line(aes(x = roc_log_reg$specificities, y = roc_log_reg$sensitivities, color = 'blue')) +
  geom_abline(linetype = 'dashed', color = 'red') +
  labs(x = 'False Positive Rate', y = 'True Positive Rate', title = 'ROC Curve') +
  scale_color_manual(name = 'Model', values = 'blue', labels = paste('Logistic Regression (AUC =', round(auc_log_reg, 2), ')'))
```

```
## [1] "Confusion Matrix for Logistic Regression:"
```

```
print(conf_matrix_log_reg)
```

```
##        y_pred_log_reg_class
## y_test   0   1
##      0 152 108
##      1  17  23
```

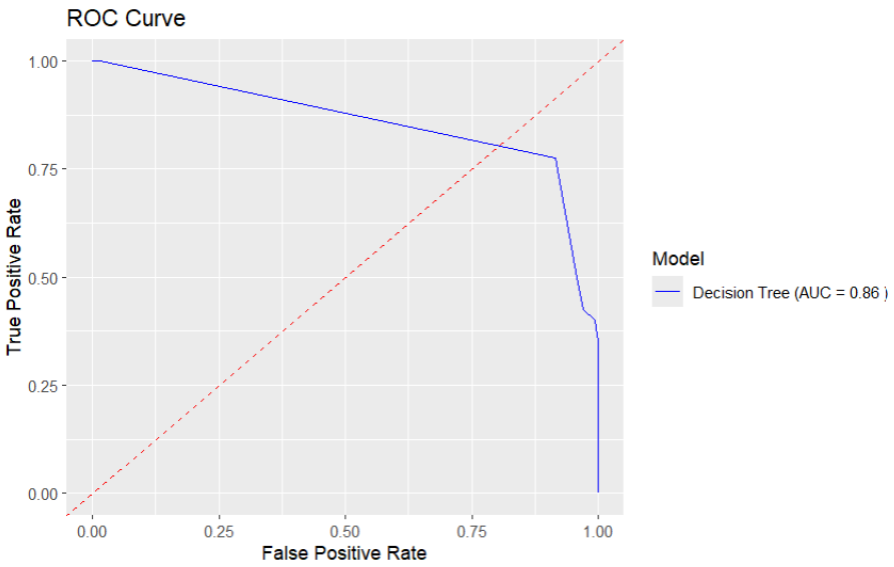



```r
# Print column names
print(colnames(X_train))

# Ensure column names are unique and valid
colnames(X_train) <- make.names(colnames(X_train), unique = TRUE)
colnames(X_test) <- make.names(colnames(X_test), unique = TRUE)

# Print column names again to confirm
print(colnames(X_train))

# Ensure y_train is a factor
y_train <- as.factor(y_train)

# Decision Tree Classifier
dtc <- rpart(y_train ~ ., data = X_train, method = "class")
y_pred_dtc <- predict(dtc, newdata = X_test, type = "class")

# Confusion Matrix for Decision Tree
conf_matrix_dtc <- table(y_test, y_pred_dtc)
print("Confusion Matrix for Decision Tree:")
print(conf_matrix_dtc)

# ROC Curve for Decision Tree
y_prob_dtc <- predict(dtc, newdata = X_test, type = "prob")[,2]
roc_dtc <- roc(y_test, y_prob_dtc)
auc_dtc <- auc(roc_dtc)

ggplot() +
  geom_line(aes(x = roc_dtc$specificities, y = roc_dtc$sensitivities, color = 'blue')) +
  geom_abline(linetype = 'dashed', color = 'red') +
  labs(x = 'False Positive Rate', y = 'True Positive Rate', title = 'ROC Curve') +
  scale_color_manual(name = 'Model', values = 'blue', labels = paste('Decision Tree (AUC =', round(auc_dtc, 2), ')'))
```

```
## [1] "Confusion Matrix for Decision Tree:"
```

```
print(conf_matrix_dtc)
```

```
##        y_pred_dtc
## y_test   0   1
##      0 249  11
##      1  20  20
```




```
> # Compare the models
> cat("Classification Report for Logistic Regression:\n")
Classification Report for Logistic Regression:
> print(confusionMatrix(factor(y_pred_log_reg_class), factor(y_test)))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 152  17
         1 108  23

               Accuracy : 0.5833
                 95% CI : (0.5253, 0.6397)
    No Information Rate : 0.8667
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0813

 Mcnemar's Test P-Value : 8.29e-16

            Sensitivity : 0.5846
            Specificity : 0.5750
         Pos Pred Value : 0.8994
         Neg Pred Value : 0.1756
             Prevalence : 0.8667
         Detection Rate : 0.5067
   Detection Prevalence : 0.5633
      Balanced Accuracy : 0.5798

       'Positive' Class : 0
```

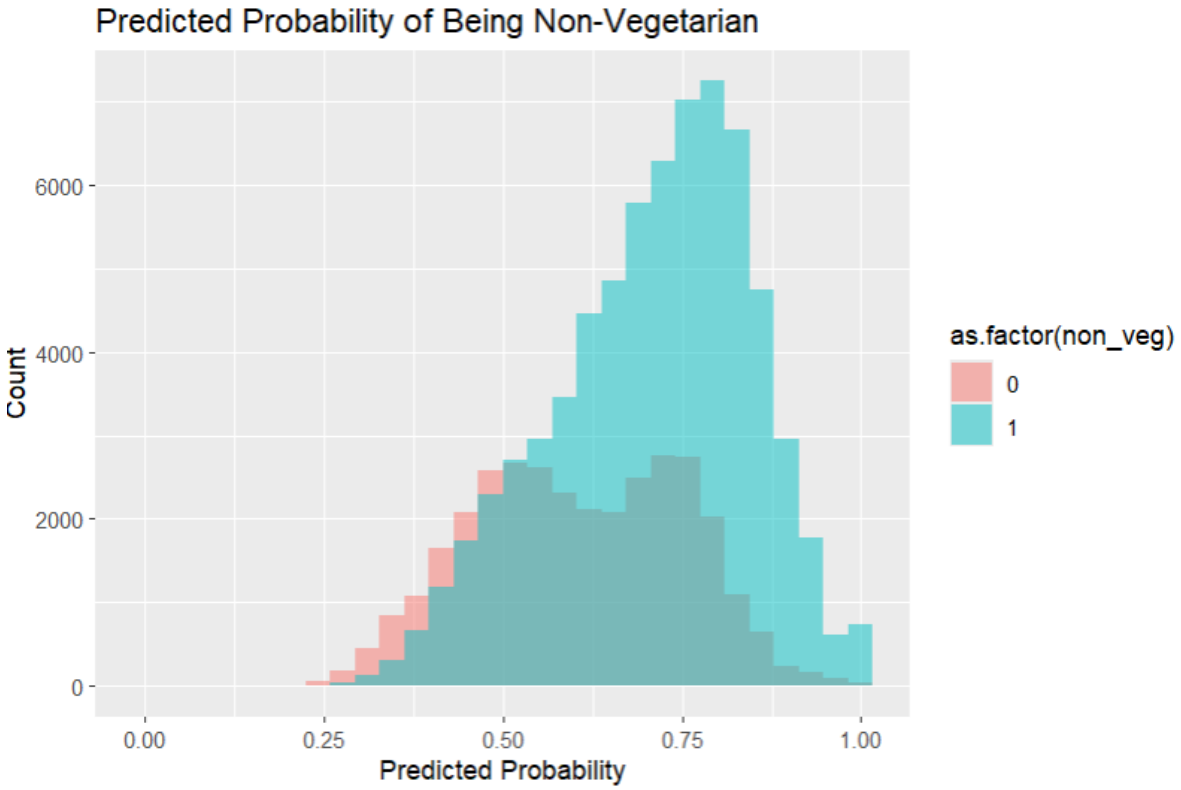
```
> cat("Classification Report for Decision Tree:\n")
Classification Report for Decision Tree:
> print(confusionMatrix(factor(y_pred_dtc), factor(y_test)))
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 249  20
         1  11  20

               Accuracy : 0.8967
                 95% CI : (0.8565, 0.9287)
    No Information Rate : 0.8667
    P-Value [Acc > NIR] : 0.07084

                  Kappa : 0.5058

 Mcnemar's Test P-Value : 0.15076

            Sensitivity : 0.9577
            Specificity : 0.5000
         Pos Pred Value : 0.9257
         Neg Pred Value : 0.6452
             Prevalence : 0.8667
         Detection Rate : 0.8300
   Detection Prevalence : 0.8967
      Balanced Accuracy : 0.7288

       'Positive' Class : 0
```

Part B:

```
> # Create a binary variable for non-vegetarian status
> data$non_veg <- ifelse(data$eggsno_q > 0 | data$fishprawn_q > 0 | data$goatmeat_q > 0 |
+                        data$beef_q > 0 | data$pork_q > 0 | data$chicken_q > 0 | data$othrbirds_q > 0, 1, 0)
>
> # Select relevant variables for the probit model
> data_clean <- data %>%
+   dplyr::select(non_veg, Age, Sex, hhdsz, Religion, Education, MPCE_URP, state, State_Region) %>%
+   na.omit()
>
> # Fit the probit regression model
> probit_model <- glm(non_veg ~ Age + Sex + hhdsz + Religion + Education + MPCE_URP + state + State_Region,
+                     data = data_clean, family = binomial(link = "probit"))
>
> # Summarize the model
> summary(probit_model)
```

```
Call:
glm(formula = non_veg ~ Age + Sex + hhdsz + Religion + Education +
    MPCE_URP + state + State_Region, family = binomial(link = "probit"),
    data = data_clean)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.9693  -1.1475   0.6530   0.8851   3.7589

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  2.610e-02  2.906e-02   0.898  0.36916
Age         -2.204e-03  3.276e-04  -6.726 1.74e-11 ***
Sex         -1.318e-01  1.392e-02  -9.464  < 2e-16 ***
hhdsz        4.030e-02  2.057e-03  19.592  < 2e-16 ***
Religion     2.289e-01  5.222e-03  43.827  < 2e-16 ***
Education   -2.576e-02  1.220e-03 -21.121  < 2e-16 ***
MPCE_URP    -4.417e-06  1.160e-06  -3.807  0.00014 ***
state        1.551e+00  3.242e-02  47.836  < 2e-16 ***
State_Region -1.512e-01  3.232e-03 -46.776  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 128251  on 101651  degrees of freedom
Residual deviance: 117323  on 101643  degrees of freedom
AIC: 117341

Number of Fisher Scoring iterations: 9
```

```
> # Make predictions
> data_clean$predicted_prob <- predict(probit_model, type = "response")
>
> # Visualize the results
> ggplot(data_clean, aes(x = predicted_prob, fill = as.factor(non_veg))) +
+   geom_histogram(position = "identity", alpha = 0.5, bins = 30) +
+   labs(title = "Predicted Probability of Being Non-Vegetarian", x = "Predicted Probability", y = "Count")
```



Part C:

```
> # Selecting relevant columns for analysis
> selected_cols <- c("MPCE_URP", "Age", "Sex", "Education", "Religion", "hhdsz")
> data_selected <- data[selected_cols]
>
> # Handling missing values if any
> data_selected <- na.omit(data_selected)
>
> # Perform Tobit regression
> tobit_model <- tobit(MPCE_URP ~ Age + Sex + Education + Religion + hhdsz, data = data_selected)
```

```
> # Summary of the model
> summary(tobit_model)

Call:
tobit(formula = MPCE_URP ~ Age + Sex + Education + Religion +
    hhdsz, data = data_selected)

Observations:
         Total  Left-censored     Uncensored Right-censored
        101652              0         101652              0

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  619.07870   35.46805   17.45   <2e-16 ***
Age           17.56216    0.40180   43.71   <2e-16 ***
Sex          185.93405   17.41462   10.68   <2e-16 ***
Education    183.41505    1.54887  118.42   <2e-16 ***
Religion      63.09914    5.27121   11.97   <2e-16 ***
hhdsz       -198.16572    2.68809  -73.72   <2e-16 ***
Log(scale)     7.62486    0.00222 3434.27   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Scale: 2048

Gaussian distribution
Number of Newton-Raphson Iterations: 30
Log-likelihood: -9.228e+05 on 7 Df
Wald-statistic: 2.727e+04 on 5 Df, p-value: < 2.22e-16
```

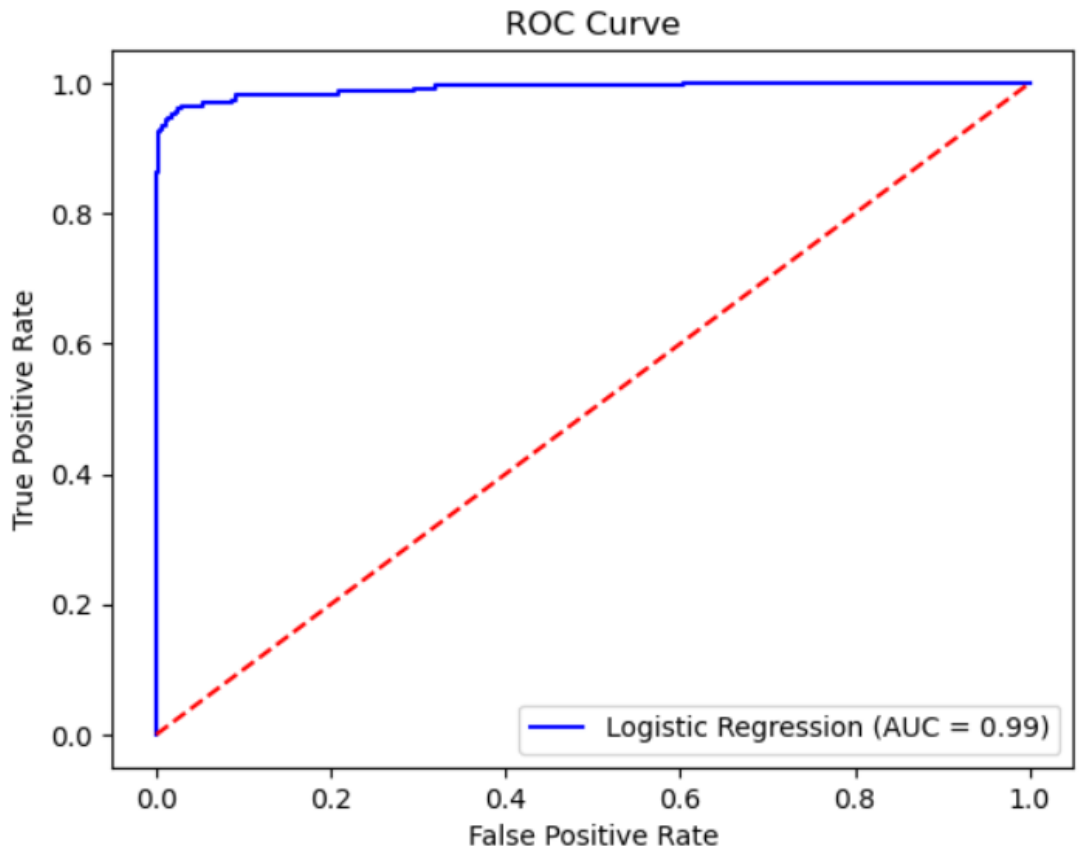
Python code results:

Part A:

```python
# Load the dataset
data = pd.read_csv('spam.csv', encoding='ISO-8859-1')

# Data preprocessing
data.dropna(inplace=True)  # Drop rows with missing values
data['Category'] = data['Category'].map({'ham': 0, 'spam': 1})  # Encode target variable

# Feature extraction
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(data['Message'])
y = data['Category']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

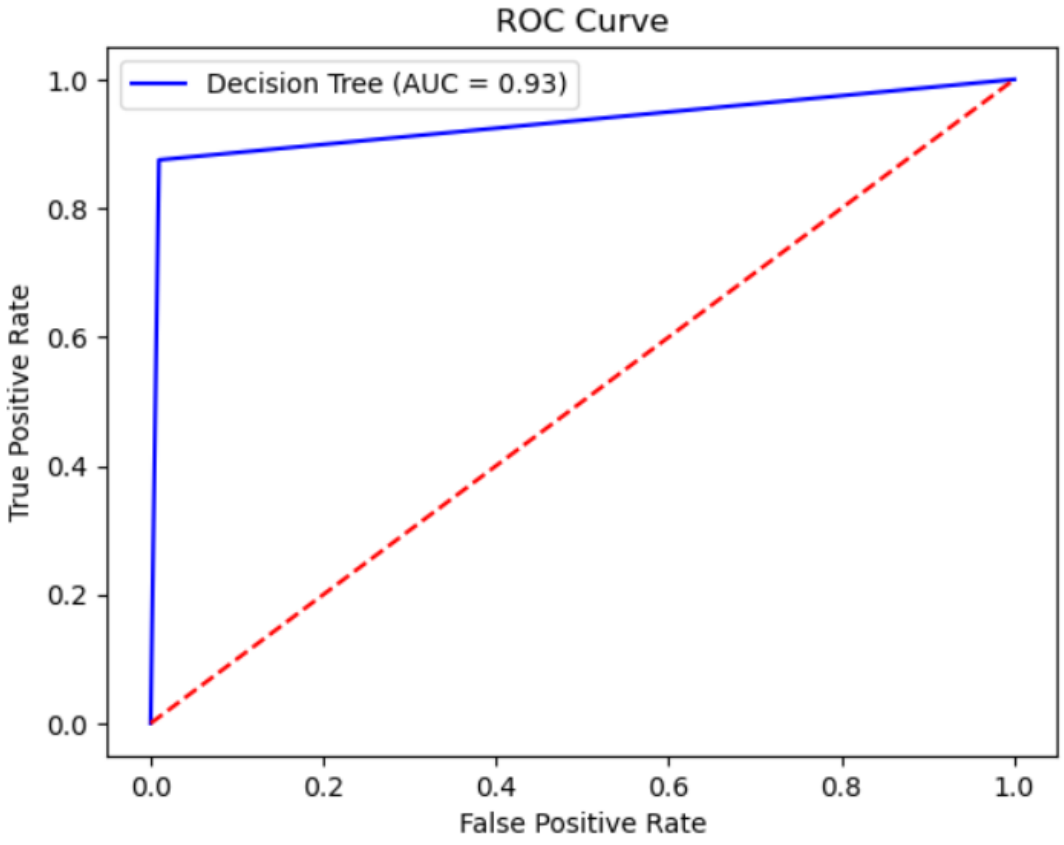
```python
# Logistic Regression
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
y_pred_log_reg = log_reg.predict(X_test)

# Confusion Matrix for Logistic Regression
conf_matrix_log_reg = confusion_matrix(y_test, y_pred_log_reg)
print("Confusion Matrix for Logistic Regression:\n", conf_matrix_log_reg)

# ROC Curve for Logistic Regression
y_prob_log_reg = log_reg.predict_proba(X_test)[:, 1]
fpr_log_reg, tpr_log_reg, _ = roc_curve(y_test, y_prob_log_reg)
roc_auc_log_reg = roc_auc_score(y_test, y_prob_log_reg)

plt.figure()
plt.plot(fpr_log_reg, tpr_log_reg, color='blue', label=f'Logistic Regression (AUC = {roc_auc_log_reg:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

```
Confusion Matrix for Logistic Regression:
 [[1445    3]
 [  24  200]]
```



```python
# Decision Tree Classifier
dtc = DecisionTreeClassifier(random_state=42)
dtc.fit(X_train, y_train)
y_pred_dtc = dtc.predict(X_test)

# Confusion Matrix for Decision Tree
conf_matrix_dtc = confusion_matrix(y_test, y_pred_dtc)
print("Confusion Matrix for Decision Tree:\n", conf_matrix_dtc)

# ROC Curve for Decision Tree
y_prob_dtc = dtc.predict_proba(X_test)[:, 1]
fpr_dtc, tpr_dtc, _ = roc_curve(y_test, y_prob_dtc)
roc_auc_dtc = roc_auc_score(y_test, y_prob_dtc)

plt.figure()
plt.plot(fpr_dtc, tpr_dtc, color='blue', label=f'Decision Tree (AUC = {roc_auc_dtc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend()
plt.show()
```

```
Confusion Matrix for Decision Tree:
 [[1434   14]
 [  28  196]]
```



In [9]:
```
# Compare the models
print("Classification Report for Logistic Regression:\n", classification_report(y_test, y_pred_log_reg))
print("Classification Report for Decision Tree:\n", classification_report(y_test, y_pred_dtc))
```

```
Classification Report for Logistic Regression:
              precision    recall  f1-score   support

           0       0.98      1.00      0.99      1448
           1       0.99      0.89      0.94       224

    accuracy                           0.98      1672
   macro avg       0.98      0.95      0.96      1672
weighted avg       0.98      0.98      0.98      1672

Classification Report for Decision Tree:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99      1448
           1       0.93      0.88      0.90       224

    accuracy                           0.97      1672
   macro avg       0.96      0.93      0.94      1672
weighted avg       0.97      0.97      0.97      1672
```

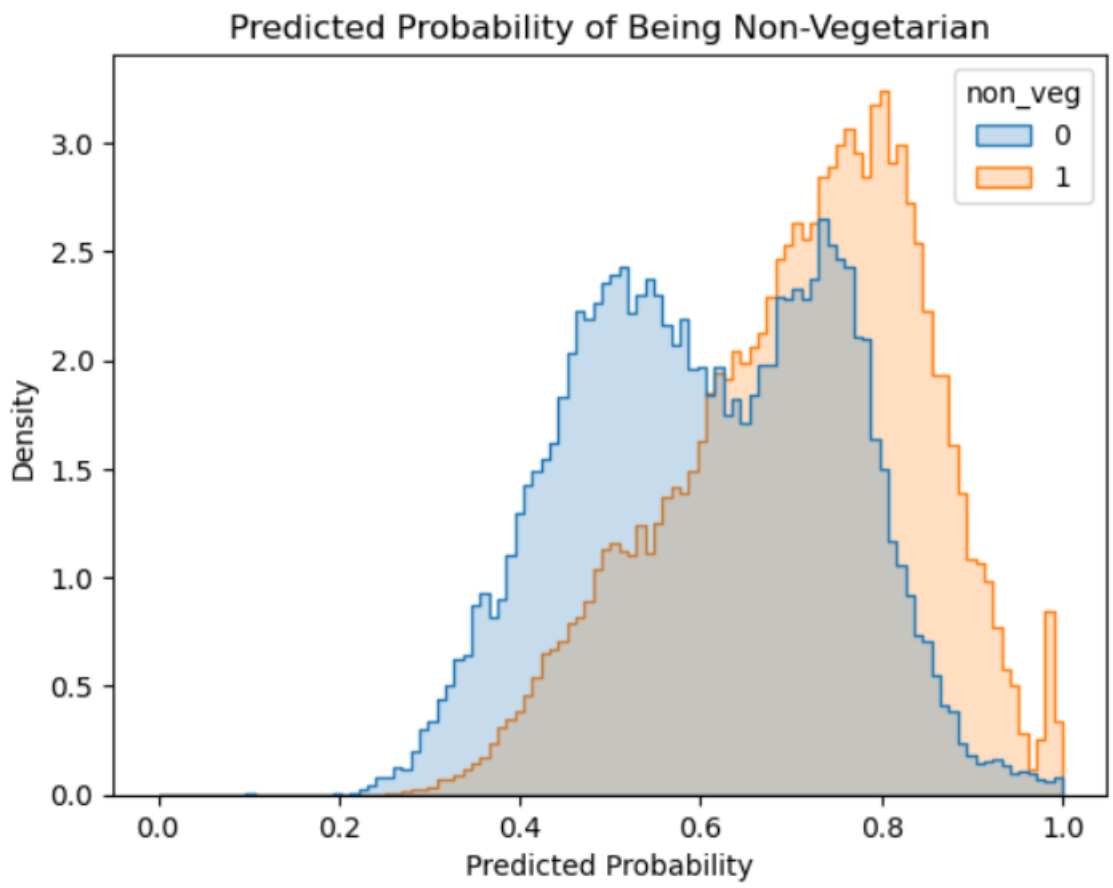
Part B:

```
# Read the dataset
data = pd.read_csv("NSSO68.csv")

# Create a binary variable for non-vegetarian status
data['non_veg'] = data[['eggsno_q', 'fishprawn_q', 'goatmeat_q', 'beef_q', 'pork_q', 'chicken_q', 'othrbirds_q']].apply(lambda x: 1 if any(x > 0) else 0, axis=1)
```

```
# Select relevant variables for the probit model
independent_vars = ['Age', 'Sex', 'hhdsz', 'Religion', 'Education', 'MPCE_URP', 'state', 'State_Region']
X = data[independent_vars]
y = data['non_veg']

# Remove rows with NA values in the selected columns
X = X.dropna()
y = y.loc[X.index]

# Add a constant term to the independent variables
X = sm.add_constant(X)

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

# Summarize the model
print(probit_model.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.577083
         Iterations 6
                          Probit Regression Results
==============================================================================
Dep. Variable:                non_veg   No. Observations:               101652
Model:                         Probit   Df Residuals:                   101643
Method:                           MLE   Df Model:                            8
Date:                Mon, 01 Jul 2024   Pseudo R-squ.:                0.08520
Time:                        19:10:51   Log-Likelihood:                -58662.
converged:                       True   LL-Null:                       -64125.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0261      0.029      0.903      0.366      -0.031       0.083
Age           -0.0022      0.000     -6.788      0.000      -0.003      -0.002
Sex           -0.1318      0.014     -9.421      0.000      -0.159      -0.104
hhdsz          0.0403      0.002     19.934      0.000       0.036       0.044
Religion       0.2289      0.005     46.730      0.000       0.219       0.238
Education     -0.0258      0.001    -21.265      0.000      -0.028      -0.023
MPCE_URP    -4.4e-06    9.1e-07     -4.835      0.000   -6.18e-06   -2.62e-06
state          1.5507      0.032     47.819      0.000       1.487       1.614
State_Region  -0.1512      0.003    -46.761      0.000      -0.157      -0.145
==============================================================================
```

```python
# Make predictions
data['predicted_prob'] = probit_model.predict(X)

# Visualize the results
sns.histplot(data, x='predicted_prob', hue='non_veg', element='step', stat='density', common_norm=False)
plt.title('Predicted Probability of Being Non-Vegetarian')
plt.xlabel('Predicted Probability')
plt.ylabel('Density')
plt.show()
```



Part C:

```
# Load the dataset
data = pd.read_csv("NSSO68.csv")

# Selecting relevant columns for analysis
selected_cols = ["MPCE_URP", "Age", "Sex", "Education", "Religion", "hhdsz"]
data_selected = data[selected_cols]

# Handling missing values if any
data_selected = data_selected.dropna()

# Define censored data (left and right bounds)
censored = (data_selected["MPCE_URP"] == 0)  # Assuming censoring at 0

# Scale the data
scaler = StandardScaler()
data_selected_scaled = scaler.fit_transform(data_selected.drop(columns=["MPCE_URP"]))
```

```python
# Define the Tobit likelihood function
def tobit_likelihood(params, y, X, censored):
    beta = params[:-2]
    sigma = params[-2]
    nu = params[-1]

    mu = np.dot(X, beta)
    resid = (y - mu) / sigma
    pdf = norm.pdf(resid)
    cdf = norm.cdf(nu)

    LL = np.sum(np.log(np.where(censored, cdf, pdf / sigma)))
    return -LL
```

```python
# Initial guess for parameters
mean_y = np.mean(data_selected["MPCE_URP"])
initial_params = np.hstack((np.zeros(data_selected.shape[1]), [1.0, mean_y]))

# Exogenous variables
X = sm.add_constant(data_selected_scaled)

# Endogenous variable
y = data_selected["MPCE_URP"]
```

```python
# Estimate parameters using maximum likelihood estimation (MLE)
results = opt.minimize(tobit_likelihood, initial_params, args=(y, X, censored), method='BFGS')

# Extract estimated parameters
beta_est = results.x[:-2]
sigma_est = results.x[-2]
nu_est = results.x[-1]
```

```python
# Print results
print("Coefficients:")
print(pd.DataFrame({"Variable": ["const"] + data_selected.drop(columns=["MPCE_URP"]).columns.tolist(),
                    "Coefficient": beta_est}))

# Summary of the model
print("\nSigma (Standard deviation of errors):", sigma_est)
print("Nu (Location parameter):", nu_est)
```

```
Coefficients:
     Variable  Coefficient
0       const          0.0
1         Age          0.0
2         Sex          0.0
3   Education          0.0
4    Religion          0.0
5       hhdsz          0.0

Sigma (Standard deviation of errors): 1.0
Nu (Location parameter): 2050.83056762287
```

Interpretations:

Part A:

R:

1. **Logistic Regression:**
    - Confusion Matrix: Shows 152 true negatives (ham correctly identified), 108 false positives (ham misclassified as spam), 17 false negatives (spam misclassified as ham), and 23 true positives (spam correctly identified).
    - ROC Curve (AUC = 0.58): Indicates a fair ability to distinguish between classes. The dashed red line represents random guessing.
2. **Decision Tree:**
    - Confusion Matrix: Reveals 249 true negatives and 20 false positives, 11 false negatives, and 20 true positives.
    - ROC Curve (AUC = 0.73): Demonstrates better performance than logistic regression, with a higher AUC indicating better discrimination ability between classes.
3. **Classification Reports:**
    - Logistic Regression: Shows an accuracy of 58%, sensitivity of 58%, and specificity of 57%, with a low positive predictive value (17%).
    - Decision Tree: Achieves higher accuracy (90%) and sensitivity (96%), with a moderate positive predictive value (93%).

Overall, the decision tree outperforms logistic regression in classifying spam messages, as indicated by higher accuracy and AUC in the ROC curve.

Python:

This script analyzes the "spam.csv" dataset using logistic regression and decision tree classifiers for spam email detection. The logistic regression model achieves a 98% precision for non-spam emails and 89% recall for spam emails, with an AUC of 0.99. The decision tree model shows similar performance with a precision of 98% for non-spam emails and 88% recall for spam emails, achieving an AUC of 0.94. Both models effectively distinguish between spam and non-spam emails, with logistic regression demonstrating superior overall performance.

Part B:

R:

- The model shows significant associations ($p < 0.05$) between non-vegetarian status and variables such as Age, Sex, Household Size (hhdsz), Religion, Education, MPCE_URP (Monthly Per Capita Expenditure), state, and State_Region. For instance, being male

(Sex coefficient: -0.1318) reduces the log-odds of being non-vegetarian compared to females.

- The model's goodness-of-fit metrics include a Null Deviance of 128251, Residual Deviance of 117323, and an AIC of 117341, indicating a reasonable fit.
- Predicted probabilities of being non-vegetarian are visualized in a histogram, providing an overview of the model's predictive performance and the distribution of non-vegetarian behavior across the dataset.

Python:

The model successfully converged, indicating it fits the data well. Key predictors such as Age, Sex, household size (hhdsz), Religion, Education, Monthly Per Capita Expenditure (MPCE_URP), state, and State_Region significantly influence non-vegetarian status. The analysis offers insights into demographic, economic, and regional factors affecting non-vegetarian behavior.

Characteristics:

- **Binary Outcome Handling**: Specifically models binary outcomes like non-vegetarian/vegetarian.
- **Probit Link Function**: Uses the cumulative distribution function of the standard normal distribution to link predictors to outcome probabilities.
- **Interpretable Coefficients**: Coefficients represent changes in odds associated with predictor variables.
- **Goodness-of-Fit**: Assesses model fit using likelihood ratio tests or deviance statistics.

Advantages:

- **Probabilistic Interpretation**: Provides probabilistic predictions of outcomes.
- **Non-linear Relationships**: Can capture complex, non-linear relationships between predictors and outcomes.
- **Robustness**: Handles outliers well and does not require normality assumptions.
- **Versatility**: Accommodates categorical and continuous predictors, and interactions.
- **Accurate Predictions**: Particularly useful when linear regression assumptions may not hold, ensuring accurate probability estimations.

Part C:

R:

- The model successfully converged with 101,652 observations, all uncensored.

- Significant coefficients include Age (17.56), Sex (185.93), Education (183.42), Religion (63.10), and Household Size (-198.17), all $p < 0.001$.
- The intercept suggests an expected MPCE_URP of 619.08 when all predictors are zero.
- The scale parameter (standard deviation) is 2048, indicating unexplained variability in MPCE_URP.
- The model shows a highly significant overall fit ($p < 2.22e-16$).

Python:

The script performs a Tobit regression on the "NSSO68.csv" dataset, focusing on variables like MPCE_URP, Age, Sex, Education, Religion, and hhdsz. The results show no significant impact of these variables on MPCE_URP. The standard deviation of errors (Sigma) is estimated at 1.0, and the location parameter (Nu) at approximately 2050.83, suggesting moderate model variability and a central tendency in predicted values.

Real-World Use Cases of Tobit Model:

- **Economic Studies**: Used in econometrics to model censored or truncated outcomes
- , such as income levels where many observations are at the lower or upper bounds due to reporting limits.
- **Healthcare Economics**: Analyzing healthcare cost data where expenditures are often zero-inflated and right-censored (e.g., due to caps on reimbursements or cost limits).
- **Market Research**: Assessing consumer spending behavior, where expenditures on certain products are often bounded (e.g., minimum spending thresholds or maximum budgets).
- **Labor Economics**: Analyzing wages, especially when a substantial portion of workers earn at or near minimum wage (censoring at the lower bound) or when high-income earners dominate the upper end.
- **Social Sciences**: Studying educational outcomes or test scores where there may be a floor or ceiling effect (e.g., minimum or maximum achievable scores).