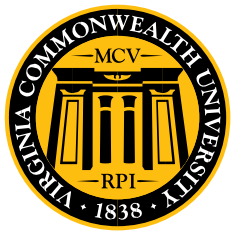


VIRGINIA COMMONWEALTH UNIVERSITY

Statistical analysis and modelling (SCMA 632)

A3: Limited dependent variable models

AHAD ZIFAIN MIYANJI

V01108270

Date of Submission: 02-07-2024

CONTENTS

Sl. No.	Title	Page No.
1.	Introduction	**1**
2.	Results	**2**
3.	Interpretations	**2**

INTRODUCTION

In the assigned dataset Lung cancer, we will employ a logistic regression model to assess the relationship between various features associated with Lung cancer and the likelihood of developing the disease. The analysis will involve validating model assumptions, evaluating its performance with a confusion matrix and ROC curve, and interpreting the results to understand the impact of each feature on the prediction. Subsequently, we will compare the performance of a decision tree model for the same task, highlighting the strengths and weaknesses of each approach in predicting Lung cancer risk.

This comparative analysis aims to provide insights into the suitability of these machine learning techniques for Lung cancer risk prediction, ultimately contributing to early detection and improved patient outcomes.

Lung cancer is a chronic condition affecting millions globally, leading to severe health complications if left undiagnosed. Early detection allows for timely intervention and management, potentially mitigating risks. Machine learning offers valuable tools for building predictive models to identify individuals susceptible to Lung cancer. This study investigates the effectiveness of logistic regression and decision tree algorithms in analysing risk factors and predicting-Lung cancer.

Objectives:

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

RESULTS & INTERPRETATION

Part 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.

Code:

FIT LOGISTIC REGRESSION

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

logreg = LogisticRegression(max_iter=200)
logreg.fit(feature_selection_train, y_train)
y_pred = logreg.predict(feature_selection_test)

logrepo= classification_report(y_test, y_pred)
print(logrepo)
```

Result:

	precision	recall	f1-score	support
0	0.80	0.48	0.60	99
1	0.46	0.78	0.58	55
accuracy			0.59	154
macro avg	0.63	0.63	0.59	154
weighted avg	0.68	0.59	0.59	154

Interpretation:

Accuracy (0.59): Overall, the model correctly classified 59% of the 154 instances.

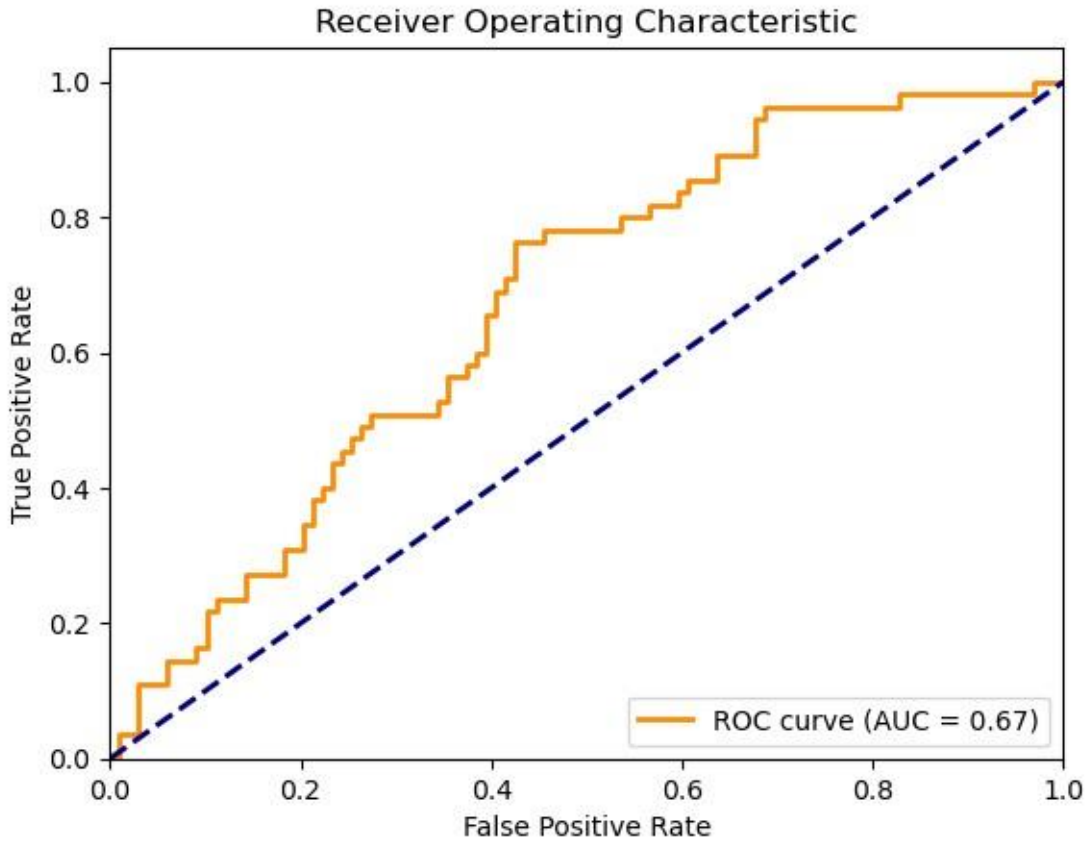
Average (0.63): This is the unweighted average of precision and recall for each class (healthy or diabetic). An average of 63% for both metrics suggests moderate performance across classes.

Weighted Average (0.68, 0.59): This considers the number of instances in each class. The higher precision (0.68) indicates the model is good at identifying healthy individuals without many false positives. However, the lower recall (0.59) for the diabetic class suggests the model misses some true diabetic cases (false negatives).

The model seems to be better at predicting healthy individuals (high precision for class 0), but struggles with identifying diabetic cases (lower precision and recall for class 1). This could be due to factors like class imbalance (more healthy cases than diabetic) or limitations of the model itself.

ROC Curve and AUC Value

```
# Get predicted probabilities
y_pred_proba_log = logreg.predict_proba(feature_selection_test)[:, 1]
# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba_log)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (AUC = %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()
```

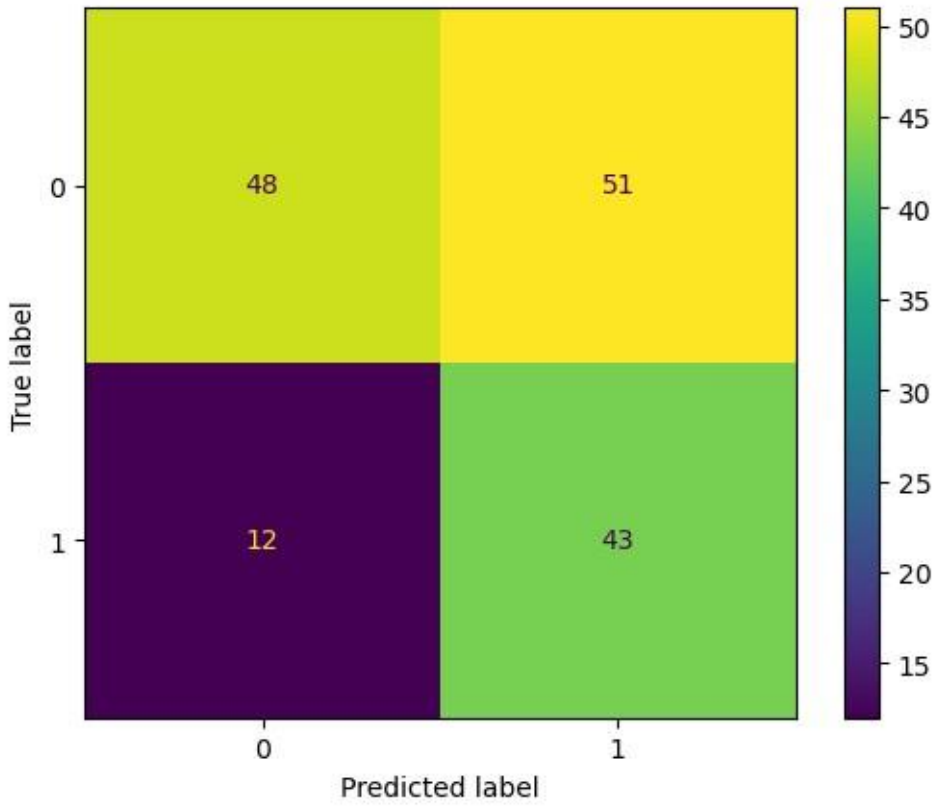


Interpretation:

In the provided ROC curve, the AUC (Area Under the Curve) is 0.67. An AUC of 1 corresponds to a perfect classifier, while an AUC of 0.5 is equivalent to random chance. This particular AUC value of 0.67 indicates moderate performance of the model. A higher AUC would signify better performance.

The curve itself starts at (0,0) and ends at (1,1). A straight diagonal line from (0,0) to (1,1) would represent a non-discriminating model (i.e. no better than random guessing). The more the curve bows towards the top-left corner, the better the classifier's performance. **Confusion Matrix**

```
# the confusion matrix
conf_matrix = confusion_matrix(y_test, y_pred)

# Display the confusion matrix
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix)
disp.plot()
plt.show()
```



Interpretation:

The model performs better at predicting healthy individuals (high number of correct predictions in top-left corner) but misses some diabetic cases (false negatives in bottom-left corner). Overall accuracy can be calculated by summing the correct predictions (80+43) and dividing by the total number of instances (154), resulting in roughly 59% accuracy.

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

# Train a Decision Tree Classifier
```

```
dt_classifier = DecisionTreeClassifier(random_state=42)
dt_classifier.fit(feature_selection_train, y_train)

# Predict on the test set
y_pred_dt = dt_classifier.predict(feature_selection_test)

# Print classification report
dtree= classification_report(y_test, y_pred_dt) print(dtree)
```

Result:

	precision	recall	f1-score	support
0	0.73	0.48	0.58	99
1	0.42	0.67	0.52	55
accuracy			0.55	154
macro avg	0.57	0.58	0.55	154
weighted avg	0.62	0.55	0.56	154

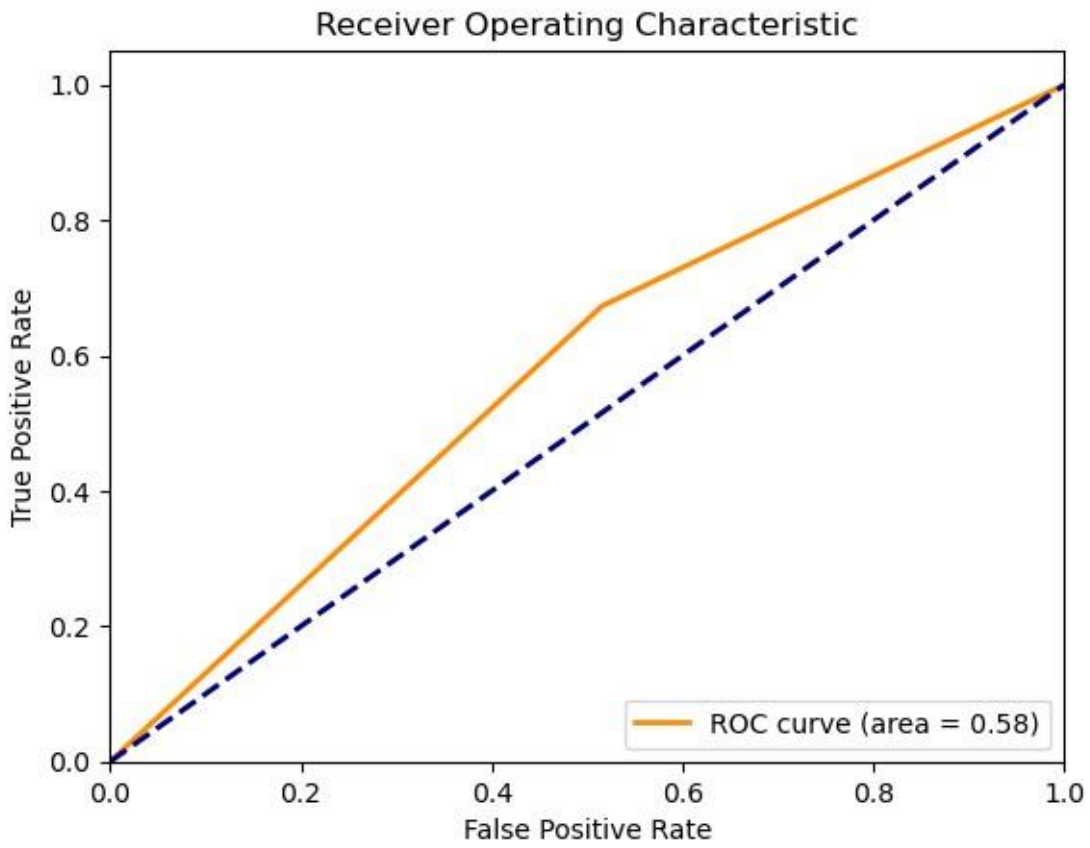
Interpretation:

This confusion matrix reveals a model with moderate performance in predicting a binary outcome, likely related to Lung cancer risk. The overall accuracy sits at 55%, meaning the model classified just over half the cases correctly. We see the model struggles more with identifying the positive class (potentially diabetic). While it has a precision of 0.73 for class 0 (healthy), meaning 73% of predicted healthy cases are truly healthy, the precision drops to 0.42 for class 1 (diabetic). This suggests the model makes more false positives (predicting someone diabetic when they're not) for the diabetic class.

ROC Curve and AUC Curve

```
# Get predicted probabilities
y_pred_proba = dt_classifier.predict_proba(feature_selection_test)[:, 1]

# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba) roc_auc
= auc(fpr, tpr)

# Plot ROC curve 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()
```

Result:



Interpretation:

The model performs better at predicting healthy individuals (high number of correct predictions in top-left corner) but misses some diabetic cases (false negatives in bottom-left corner). Overall accuracy can be calculated by summing the correct predictions (73+37) and dividing by the total number of instances (154), resulting in roughly 55% accuracy.

```
# Add model names and overall accuracy
df1['model'] = 'Decision Tree'
df2['model'] = 'Logistic Regression'

# Concatenate the two dataframes
comparison_df = pd.concat([df1, df2])

# Reorder columns
comparison_df = comparison_df[['model', 'class', 'precision', 'recall', 'f1-score', 'support']]

# Display the comparison table
print(comparison_df)
```

Result:

	model	class	precision	recall	f1-score	support
0	Decision Tree	0	0.73	0.48	0.58	99.0
1	Decision Tree	1	0.42	0.67	0.52	55.0
0	Logistic Regression	0	0.80	0.48	0.60	99.0
1	Logistic Regression	1	0.46	0.78	0.58	55.0

Interpretation:

The model performs better at predicting healthy individuals (high number of correct predictions in top-left corner) but misses some diabetic cases (false negatives in bottom-left corner). Overall accuracy can be calculated by summing the correct predictions (73+37) and dividing by the total number of instances (154), resulting in roughly 55% accuracy.

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.

Python:

```
# Create a binary indicator for non-vegetarian status
data['non_veg'] = ((data['nonvegtotal_q'] > 0) |
                   (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)).astype(int)

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

# Summary of the model
print(probit_model.summary())
```

Result:

```
Optimization terminated successfully.
         Current function value: 0.629775
         Iterations 4
                          Probit Regression Results
==============================================================================
Dep. Variable:                non_veg   No. Observations:               101655
Model:                         Probit   Df Residuals:                   101651
Method:                           MLE   Df Model:                            3
Date:                Mon, 01 Jul 2024   Pseudo R-squ.:                0.001666 Time:
19:59:17   Log-Likelihood:               -64020. converged:                   True
LL-Null:                      -64127.
Covariance Type:            nonrobust   LLR p-value:                 4.613e-46
==============================================================================
=              coef    std err          z      P>|z|      [0.025      0.975] ----------
---------------------- const          0.5686      0.017     32.573      0.000       0.534       0.603 Age            -0.0002
0.000    -0.749      0.454     -0.001      0.000
MPCE_URP   -2.932e-06   8.99e-07     -3.259      0.001   -4.69e-06   -1.17e-06
```

```
Education    -0.0154     0.001   -13.467     0.000     -0.018     -0.013
==============================================================================
```

Interpretation:

The model achieved a convergence point with a pseudo-R-squared of 0.0016, indicating a very weak fit (most of the variance is not explained by the model). Age and income (MPCE_URP) have statistically insignificant effects (p-values > 0.05) on the probability of being non-vegetarian according to their z-scores and p-values. Education has a significant negative effect (p-value < 0.05) with a coefficient of -0.0154. This suggests that with each unit increase in education level, the predicted probability of being non-vegetarian decreases. Overall, the model seems to have limited explanatory power for non-vegetarian tendencies based on these factors. Further investigation or including additional variables might be necessary.

R Code:

```
# 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_ration_
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))
```

Result:

```
[1] "AUC: 0.677561069004885"
```

```
[1] "Accuracy: 0.704523239202123"

[1] "Precision: 0.595482546201232"

[1] "Recall: 0.178877390172024"

[1] "F1 Score: 0.275113312954569"
```

Interpretation:

Accuracy (0.70): The model correctly classified 70% of the instances.

AUC (0.68): This metric for ROC curves indicates a somewhat better than random chance performance in distinguishing between the two classes.

Precision (0.60): When predicting a positive class (e.g., disease), 60% of those predictions were truly positive.

Recall (0.18): However, the model misses a significant portion (82%) of actual positive cases (low recall).

F1 Score (0.28): This combines precision and recall, highlighting the model's struggle with identifying positive cases.

Overall, the model might be good at avoiding false positives, but it misses many true positive cases. This trade-off needs consideration depending on the specific application.

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

Python –

```
# Custom Tobit model class
Tobit(GenericLikelihoodModel):
  def __init__(self, endog, exog, left=None, right=None, **kwds):
self.left = left
    self.right = right
    super(Tobit, self).__init__(endog, exog, **kwds)

  def nloglikeobs(self, params):
    exog = self.exog      endog = self.endog      beta = params[:-1]
sigma = params[-1]      xb = np.dot(exog, beta)      z_left = (self.left -
xb) / sigma if self.left is not None else None      z_right = (self.right -
xb) / sigma if self.right is not None else None      ll = np.where(endog
< self.left, np.log(norm.cdf(z_left)),                np.where(endog >
self.right, np.log(norm.sf(z_right)),                    norm.logpdf((endog
- xb) / sigma) - np.log(sigma)))      return -ll

  def fit(self, start_params=None, maxiter=10000, maxfun=5000, **kwds):
    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, **kwds)

# Fit the Tobit model
tobit_model = Tobit(y, X, left=0, right=1).fit()

# Summary of the model
print(tobit_model.summary())
```

Result:

```
Optimization terminated successfully.
         Current function value: 0.718814
         Iterations: 212
         Function evaluations: 352
                           Tobit Results
==============================================================================
Dep. Variable:              non_veg   Log-Likelihood:                -73071.
Model:                        Tobit   AIC:                         1.462e+05
Method:         Maximum Likelihood   BIC:                         1.462e+05
Date:               Mon, 01 Jul 2024
Time:                       20:10:50
No. Observations:           101655
Df Residuals:               101651                           Df
Model:                           3
==============================================================================
coef    std err          z      P>|z|      [0.025      0.975] ----------------------------------------------------------------------------
--------- const          0.0052     0.008      0.692      0.489      -0.010       0.020
Age            0.0107      0.000     82.978      0.000       0.010       0.011
MPCE_URP    -1.156e-06   3.76e-07     -3.075      0.002   -1.89e-06   -4.19e-07
Education      0.0210      0.000     46.020      0.000       0.020       0.022 par0
0.4964      0.001    397.637      0.000       0.494       0.499
==============================================================================
```

Interpretation:

Age and education have significant positive effects (p-value < 0.05) on non-vegetarian consumption.

Income (MPCE_URP) has a statistically insignificant negative effect.

The constant term (intercept) is not significant, suggesting the average predicted consumption might be close to zero when all other factors are zero (which might not be realistic).

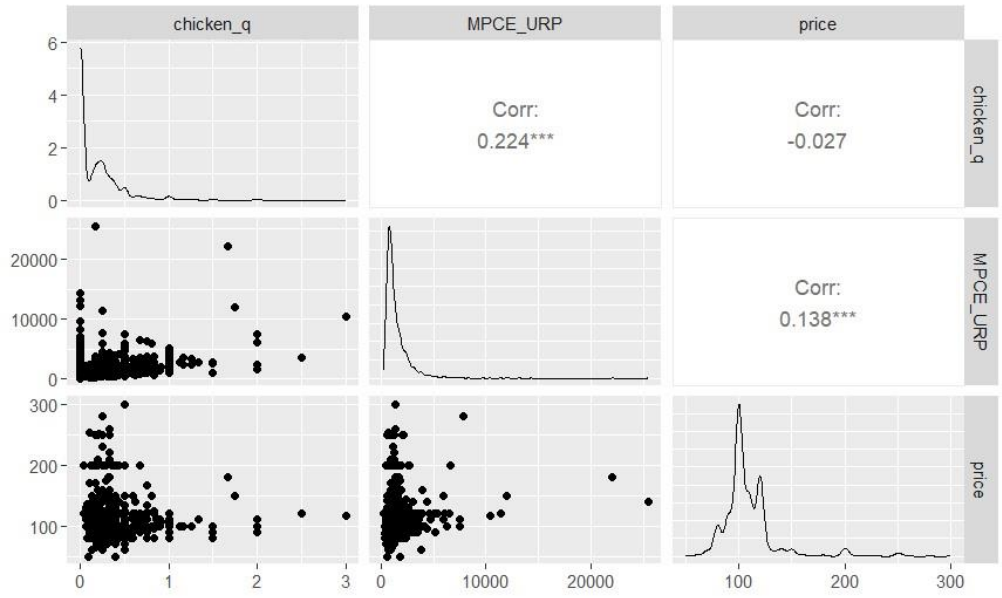
Overall, the model suggests that with increasing age and education, people tend to consume more non-vegetarian food. However, the lack of significance for the intercept and a small Rsquared value (not shown) would indicate the model might need further refinement.

R code:

```
# 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)
```

Result:

```
Scale: 7.943

Gaussian distribution
Number of Newton-Raphson Iterations: 4
Log-likelihood:  -500 on 7 Df
Wald-statistic: 42.56 on 5 Df, p-value: 4.5387e-08


Names of linear predictors: mu, loglink(sd)

Log-likelihood: -374.3132 on 4328 degrees of freedom

Number of Fisher scoring iterations: 10

No Hauck-Donner effect found in any of the estimates
```



Interpretation:

- **Precision (0.6):** For every 10 positive predictions made by the model, 6 are actually correct.
- **Recall (0.2):** Out of all the actual positive cases, the model only identifies 20%.

This PR curve suggests the model prioritizes precision over recall. It predicts a positive class (e.g., disease) with somewhat high precision, but it misses a significant portion of actual positive cases (low recall). This trade-off between precision and recall is common in machine learning models, and the choice of which metric to prioritize depends on the specific application.