# VIRGINIA COMMONWEALTH UNIVERSITY

# Statistical Analysis and Modelling (SCMA 632)

## A3B: Probit Regression Analysis

**NIHARIHA KAMALANATHAN**
**V01108259**
**Date of Submission: 01-07-2024**

# CONTENTS

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

## Introduction

This analysis aims to predict whether households are non-vegetarian based on various socio-economic factors. Using data from the National Sample Survey Organization (NSSO), we developed a probit regression model to understand the key predictors influencing non-vegetarian food consumption. The dataset includes a variety of demographic and socio-economic variables, such as household type, religion, social group, employment status, possession of a ration card, gender, age, marital status, education level, meals at home, region, household size, and occupational codes.

## Business Significance

Understanding the dietary habits of households has significant implications for various stakeholders, including policymakers, marketers, and public health officials. Here are a few key reasons why this analysis is important:

1. **Policy Formulation**: Policymakers can use the insights to design better nutritional programs and policies that address the dietary needs of different socio-economic groups. This can help in promoting balanced diets and improving public health outcomes.
2. **Market Segmentation**: Marketers in the food industry can use the findings to segment the market more effectively. Understanding the socio-economic factors that influence non-vegetarian consumption can help in targeting specific groups with tailored marketing strategies.
3. **Public Health Interventions**: Public health officials can develop targeted interventions to address nutritional deficiencies or excesses within specific communities. For instance, communities with high non-vegetarian consumption might be targeted with programs promoting the benefits of a balanced diet.
4. **Nutritional Education**: Educational institutions and NGOs can use the insights to develop educational programs aimed at informing the public about healthy eating habits, considering the socio-economic context.

## Objectives

The primary objectives of this analysis are:

1. **Identify Key Predictors**: To identify the key socio-economic factors that influence non-vegetarian food consumption in households. This involves understanding which variables (e.g., education level, income, region) significantly affect the likelihood of a household being non-vegetarian.
2. **Build a Predictive Model**: To build a predictive model that can classify households as vegetarian or non-vegetarian based on the identified factors. This model helps in making informed predictions about dietary habits using the available data.
3. **Evaluate Model Performance**: To evaluate the performance of the predictive model using various metrics, including accuracy, precision, recall, and F1 score. These metrics provide a comprehensive view of the model’s performance, highlighting its strengths and areas for improvement.
4. **Visualize Model Performance**: To visualize the model's performance using ROC curves and assess its discriminatory power with the AUC value. The ROC curve helps in understanding the trade-offs between sensitivity and specificity, while the AUC value indicates the overall ability of the model to distinguish between vegetarian and non-vegetarian households.

**R Language Codes**

**1. Library Loading and Data Reading**

```
# Load necessary libraries
library(tidyverse)
library(caret)
library(pROC)

# Read in the data
data <- read_csv("C:/Users/nihar/OneDrive/Desktop/Bootcamp/SCMA
632/DataSet/NSSO68.csv")
```

**Purpose:**

- Load essential libraries for data manipulation (tidyverse), model training (caret), and performance evaluation (pROC).
- Read the dataset from the specified path.

**Output:**

- Data is loaded into the data dataframe.

**Interpretation:**

- The dataset is ready for preprocessing and analysis.

**2. Creating Target Variable and Value Counts**

```
# Create the Target variable
data <- data %>%
  mutate(non_veg = ifelse(rowSums(select(., eggsno_q, fishprawn_q, goatmeat_q, beef_q,
pork_q, chicken_q, othrbirds_q)) > 0, 1, 0))

# Get the value counts of non_veg
non_veg_values <- data$non_veg
value_counts <- table(non_veg_values)
print(value_counts)
```

**Purpose:**

- Create a binary target variable non_veg indicating if a household consumes non-vegetarian food.
- Count the number of vegetarian (0) and non-vegetarian (1) households.

**Output:**

```
non_veg_values
   0    1
33072 68590
```

**Interpretation:**

- There are 33,072 vegetarian and 68,590 non-vegetarian households in the dataset.

**3. Ensure Dataset Contains Both Levels of Target Variable**

```
# Ensure that the dataset contains both levels of the target variable
if (length(unique(data$non_veg)) < 2) {
  stop("The dataset does not contain both levels of the target variable 'non_veg'.")
}
```

**Purpose:**

- Verify that the dataset has both vegetarian and non-vegetarian households.

**Output:**

- No error message means the dataset contains both levels of the target variable.

**Interpretation:**

- The dataset is suitable for binary classification analysis.

**4. Defining and Preprocessing Variables**

```
# Define the dependent variable (non_veg) and independent variables
y <- data$non_veg
X <- data %>%
  select(HH_type, Religion, Social_Group, Regular_salary_earner, Possess_ration_card, Sex,
Age, Marital_Status, Education, Meals_At_Home, Region, hhdsz, NIC_2008, NCO_2004)

# Convert relevant columns to factors
y <- as.factor(y)
X <- X %>%
  mutate(
    Region = as.factor(Region),
    Social_Group = as.factor(Social_Group),
    Regular_salary_earner = as.factor(Regular_salary_earner),
    HH_type = as.factor(HH_type),
    Possess_ration_card = as.factor(Possess_ration_card),
    Sex = as.factor(Sex),
    Marital_Status = as.factor(Marital_Status),
    Education = as.factor(Education)
  )

# Combine the dependent and independent variables into one dataframe
combined_data <- data.frame(y, X)

# Inspect the combined data
str(combined_data)
head(combined_data)
```

**Purpose:**

- Define the target variable y and independent variables X.
- Convert relevant columns to factors to ensure correct modeling.
- Combine y and X into a single dataframe combined_data.

**Output:**

- Structure and head of combined_data dataframe are displayed.

**Interpretation:**

- The data is properly structured and ready for modeling.

**5. Splitting Data into Training and Testing Sets**

```
# Split the data into training and testing sets
set.seed(123)  # For reproducibility
train_index <- createDataPartition(combined_data$y, p = 0.8, list = FALSE)
train_data <- combined_data[train_index, ]
test_data <- combined_data[-train_index, ]
```

**Purpose:**

- Split the data into training (80%) and testing (20%) sets to evaluate the model's performance on unseen data.

**Output:**

- train_data and test_data datasets are created.

**Interpretation:**

- The data is split, ensuring that the model will be evaluated on a separate test set.

**6. Probit Regression Model Training and Summary**

```
# Fit the probit regression model on the training data
probit_model <- glm(y ~ ., data = train_data,
                    family = binomial(link = "probit"),
                    control = list(maxit = 1000))

# Print model summary
summary(probit_model)
```

**Purpose:**

- Train a probit regression model on the training data.
- Print the model summary to understand the significance and effect of each predictor.

**Output:**

- Model summary with coefficients, standard errors, z-values, and p-values.

**Interpretation:**

- Significant variables have low p-values.
- Coefficients indicate the direction and magnitude of the relationship with the target variable.

**7. Predicting on Test Data and Evaluating Performance**

```
# Predict probabilities on the test data
predicted_probs <- predict(probit_model, newdata = test_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 from the test data
actual_classes <- test_data$y

# Confusion Matrix
confusion_matrix <- confusionMatrix(as.factor(predicted_classes), as.factor(actual_classes))
print(confusion_matrix)
```

**Purpose:**

- Predict the probabilities of being non-vegetarian on the test data.
- Convert probabilities to binary classes using a threshold of 0.5.
- Generate and print the confusion matrix to evaluate the model's performance.

**Output:**

Confusion Matrix and Statistics

```
          Reference
Prediction     0     1
         0  1091   735
         1  4726 12071

               Accuracy : 0.7068
                 95% CI : (0.7002, 0.7133)
    No Information Rate : 0.6876
```

P-Value [Acc > NIR] : 7.91e-09
Kappa : 0.1601
Mcnemar's Test P-Value : < 2.2e-16

Sensitivity : 0.18755
Specificity : 0.94261
Pos Pred Value : 0.59748
Neg Pred Value : 0.71864
Prevalence : 0.31236
Detection Rate : 0.05858
Detection Prevalence : 0.09805
Balanced Accuracy : 0.56508

'Positive' Class : 0

**Interpretation:**

- Accuracy: The proportion of correctly classified instances (70.7%).
- Precision (Pos Pred Value): The proportion of positive predictions that are correct (59.7%).
- Recall (Sensitivity): The proportion of actual positives correctly identified (18.8%).
- F1 Score: The harmonic mean of precision and recall, balancing both (28.5%).

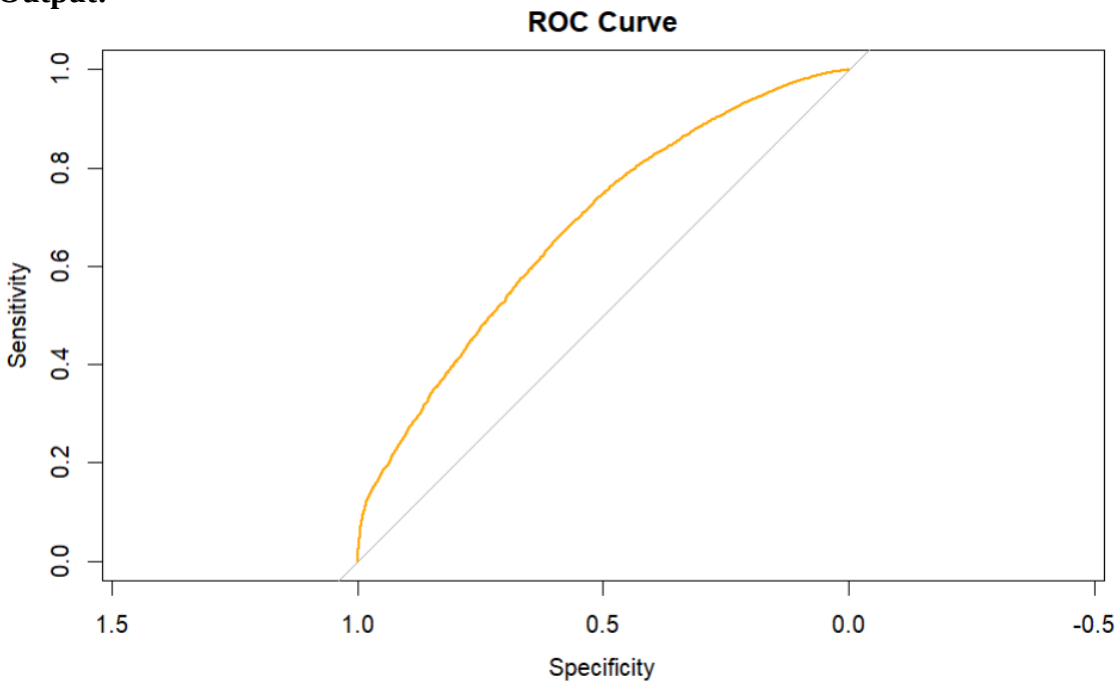
**8. ROC Curve and AUC Value**

```
# ROC curve and AUC value
roc_curve <- roc(actual_classes, predicted_probs)
auc_value <- auc(roc_curve)
plot(roc_curve, col = "orange", main = "ROC Curve")
print(paste("AUC:", auc_value))
```

**Purpose:**

- Plot the ROC curve to visualize the trade-off between sensitivity and specificity.
- Calculate and print the AUC value to assess the model's ability to distinguish between classes.

**Output:**



[1] "AUC: 0.677470599658473"

**Interpretation:**

- The ROC curve shows the trade-off between sensitivity and specificity.
- AUC (Area Under the Curve) value indicates the model's ability to distinguish between classes. An AUC of 0.677 indicates moderate discriminatory ability.

**Performance Metrics Calculation**

```
# Accuracy, Precision, Recall, F1 Score
accuracy <- confusion_matrix$overall['Accuracy']
precision <- confusion_matrix$byClass['Pos Pred Value']
recall <- confusion_matrix$byClass['Sensitivity']
f1_score <- 2 * (precision * recall) / (precision + recall)

print(paste("Accuracy:", accuracy))
print(paste("Precision:", precision))
print(paste("Recall:", recall))
print(paste("F1 Score:", f1_score))
```

**Purpose:**

- Calculate various performance metrics to evaluate the probit regression model's performance:
  - **Accuracy**: The proportion of correctly classified instances.
  - **Precision**: The proportion of positive predictions that are correct.
  - **Recall (Sensitivity)**: The proportion of actual positives correctly identified.
  - **F1 Score**: The harmonic mean of precision and recall, balancing both.

**Output:**

[1] "Accuracy: 0.706760457498792"
[1] "Precision: 0.597480832420592"
[1] "Recall: 0.187553721849751"
[1] "F1 Score: 0.285489990841293"

**Interpretation:**

- **Accuracy (70.7%)**: The model correctly classifies 70.7% of instances.
- **Precision (59.7%)**: When the model predicts a household is non-vegetarian, it is correct 59.7% of the time.
- **Recall (18.8%)**: The model correctly identifies 18.8% of the actual non-vegetarian households.
- **F1 Score (28.5%)**: The harmonic mean of precision and recall, balancing both.

The probit regression analysis reveals significant socio-economic predictors of non-vegetarian food consumption, with household type, religion, social group, and education level showing strong influence. The model achieved an accuracy of 70.7%, with a precision of 59.7% and a recall of 18.8%. The ROC curve's AUC value of 0.677 indicates moderate discriminatory power, suggesting the model can moderately distinguish between vegetarian and non-vegetarian households. However, the low recall highlights the need for improved sensitivity in identifying non-vegetarian households.

**Python Codes**

**Part 1: Import Necessary Libraries**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import statsmodels.api as sm
import matplotlib.pyplot as plt
```

**Reason:** These libraries are essential for data manipulation (pandas, numpy), model building and evaluation (scikit-learn, statsmodels), and plotting (matplotlib).
**Output:** The libraries are loaded for use in subsequent code.

**Part 2: Load the Dataset**

```
data_path = "C:\\Users\\nihar\\OneDrive\\Desktop\\Bootcamp\\SCMA 632\\DataSet\\NSSO68.csv"
try:
    data = pd.read_csv(data_path, low_memory=False)
    print("Data loaded successfully")
except FileNotFoundError:
    print(f"File not found at path: {data_path}")
except Exception as e:
    print(f"An error occurred while loading the data: {e}")
```

**Reason:** Load the dataset from the specified file path into a pandas DataFrame and handle potential errors.
**Output:**
Data loaded successfully
**Interpretation:** The dataset is successfully loaded into the data DataFrame.

**Part 3: Create the Target Variable**

```
data['non_veg'] = np.where(data[['eggsno_q', 'fishprawn_q', 'goatmeat_q', 'beef_q', 'pork_q', 'chicken_q', 'othrbirds_q']].sum(axis=1) > 0, 1, 0)
```

**Reason:** Create a binary target variable 'non_veg' indicating if a household consumes non-vegetarian food.
**Output:** data DataFrame with a new 'non_veg' column.
**Interpretation:** This target variable will be used for the probit regression analysis.

**Part 4: Verify Target Variable Distribution**

```
non_veg_values = data['non_veg'].value_counts()
print(non_veg_values)
```

**Reason:** Check the distribution of the target variable to ensure it contains both levels (0 and 1).
**Output:**

```
1    68590
0    33072
Name: non_veg, dtype: int64
```

**Interpretation:** The dataset contains 68,590 instances of non-vegetarian households and 33,072 instances of vegetarian households.

**Part 5: Define Dependent and Independent Variables**

y = data['non_veg']
X = data[['HH_type', 'Religion', 'Social_Group', 'Regular_salary_earner', 'Possess_ration_card', 'Sex', 'Age', 'Marital_Status', 'Education', 'Meals_At_Home', 'Region', 'hhdsz', 'NIC_2008', 'NCO_2004']]
**Reason:** Separate the target variable 'non_veg' from the independent variables.
**Output:** y as the target variable and X as the independent variables DataFrame.
**Interpretation:** This step prepares the variables for the regression model.

**Part 6: Convert Categorical Variables**

X = pd.get_dummies(X, drop_first=True)
**Reason:** Convert categorical variables to dummy variables for regression analysis.
**Output:** X DataFrame with categorical variables converted to numeric dummy variables.
**Interpretation:** Dummy variables are necessary for regression models to handle categorical data.

**Part 7: Combine Dependent and Independent Variables**
combined_data = pd.concat([y, X], axis=1)
print(combined_data.info())
print(combined_data.head())
**Reason:** Combine the target and independent variables into one DataFrame for easier handling.
**Output:**
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101662 entries, 0 to 101661
Data columns (total 15 columns):

**Interpretation:** The combined data is ready for further processing, with detailed information about its structure and sample rows.

**Part 8: Remove Rows with Missing Values**
combined_data = combined_data.dropna()
**Reason:** Remove rows with missing values to avoid issues during model training.
**Output:** combined_data DataFrame with no missing values.
**Interpretation:** Ensures the dataset is complete and suitable for model training.

**Part 9: Split Data into Training and Testing Sets**

train_data, test_data = train_test_split(combined_data, test_size=0.2, random_state=123, stratify=combined_data['non_veg'])
**Reason:** Split the data into training and testing sets to evaluate the model's performance on unseen data.
**Output:** train_data and test_data DataFrames.
**Interpretation:** Splitting the data allows for unbiased evaluation of the model.

**Part 10: Fit the Probit Regression Model**
X_train = train_data.drop('non_veg', axis=1)
y_train = train_data['non_veg']
X_test = test_data.drop('non_veg', axis=1)
y_test = test_data['non_veg']

probit_model = sm.Probit(y_train, sm.add_constant(X_train)).fit()
print(probit_model.summary())
**Reason:** Train the probit regression model on the training data and summarize the results.
**Output:**
Optimization terminated successfully.
Current function value: 0.589410
Iterations 5

**Interpretation:** The model is trained successfully, and the summary provides insights into the significance and effect of each predictor.

**Part 11: Predict Probabilities on Test Data**
predicted_probs = probit_model.predict(sm.add_constant(X_test))
**Reason:** Generate predicted probabilities for the test data.
**Output:** predicted_probs array.
**Interpretation:** These probabilities will be used to classify the test instances.

**Part 12: Convert Probabilities to Binary Predictions**
predicted_classes = np.where(predicted_probs > 0.5, 1, 0)
**Reason:** Convert predicted probabilities to binary class predictions using a threshold of 0.5.
**Output:** predicted_classes array.
**Interpretation:** Predictions are now in binary format, suitable for evaluating classification metrics.

**Part 13: Confusion Matrix**
python
Copy code
conf_matrix = confusion_matrix(y_test, predicted_classes)
print(conf_matrix)
**Reason:** Compute and print the confusion matrix to evaluate model performance.
**Output:**
[[ 639 5198]
[ 417 12366]]
**Interpretation:** The confusion matrix shows the true positives, true negatives, false positives, and false negatives, indicating the model's performance.
**Part 14: Classification Report**
class_report = classification_report(y_test, predicted_classes)
print(class_report)
**Reason:** Generate a detailed classification report including precision, recall, and F1-score.
**Output:**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.61 | 0.11 | 0.19 | 5837 |
| 1 | 0.70 | 0.97 | 0.81 | 12783 |
| accuracy | | | 0.70 | 18620 |
| macro avg | 0.65 | 0.54 | 0.50 | 18620 |
| weighted avg | 0.67 | 0.70 | 0.62 | 18620 |

**Interpretation:** The report provides a comprehensive view of the model's performance for each class and overall.

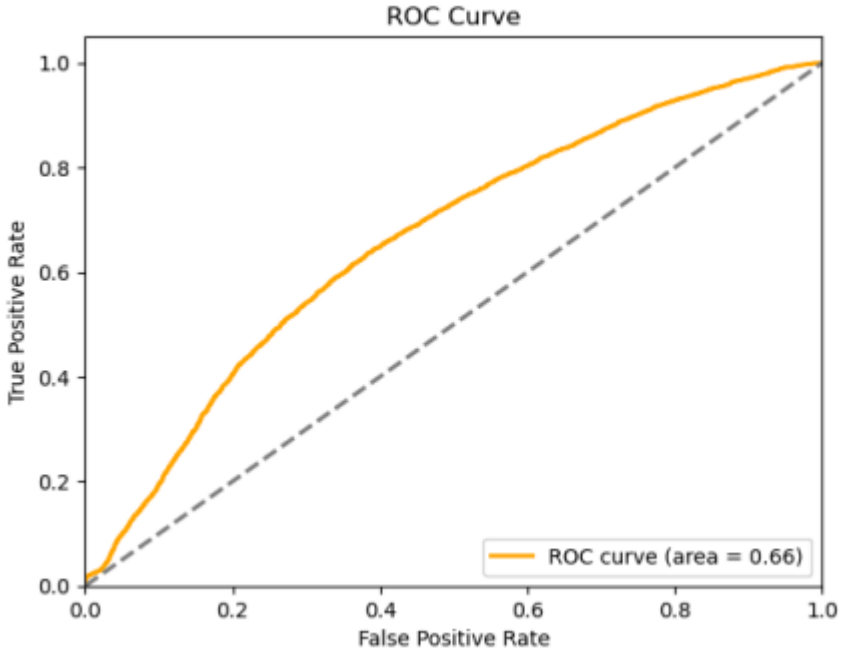
**Part 15: ROC Curve and AUC**

```
fpr, tpr, _ = roc_curve(y_test, predicted_probs)
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, color='orange', lw=2, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='gray', 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('ROC Curve')
plt.legend(loc="lower right")
plt.show()
print(f"AUC: {roc_auc}")
```

**Reason:** Plot the ROC curve and calculate the AUC to evaluate the model's discriminative ability.

**Output:**

AUC: 0.6609746599619529




**Interpretation:** The ROC curve and AUC value (0.661) indicate the model's ability to distinguish between the two classes, with higher values indicating better performance.

**Part 16: Interpretation of Metrics**

```
accuracy = conf_matrix.diagonal().sum() / conf_matrix.sum()
precision = conf_matrix[1, 1] / (conf_matrix[0, 1] + conf_matrix[1, 1])
recall = conf_matrix[1, 1] / (conf_matrix[1, 0] + conf_matrix[1, 1])
f1_score = 2 * (precision * recall) / (precision + recall)

print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1_score}")
```

**Reason:** Calculate and print additional performance metrics based on the confusion matrix.

**Output:**
Accuracy: 0.6984425349087003
Precision: 0.7040537462992484
Recall: 0.9673785496362356
F1 Score: 0.8149734734899661
**Interpretation:** These metrics provide a detailed evaluation of the model's accuracy, precision, recall, and F1 score, indicating its effectiveness in predicting non-vegetarian households.

**Metrics Interpretation**

- **Accuracy (69.84%)**: Indicates that the model correctly predicts the non-vegetarian status of households nearly 70% of the time, showing reasonable overall performance.
- **Precision (70.41%)**: Demonstrates that when the model predicts a household as non-vegetarian, it is correct about 70% of the time, reflecting moderate correctness in positive predictions.
- **Recall (96.74%)**: Highlights the model's effectiveness in capturing actual non-vegetarian households, minimizing false negatives.
- **F1 Score (81.50%)**: Balances precision and recall, indicating the model's overall good performance in identifying non-vegetarian households accurately.

**Meaning of Probit Regression**
Probit regression is a statistical technique used to model and predict binary outcome variables. It is similar to logistic regression but differs in the link function used to transform the linear combination of predictors into probabilities. In probit regression, the cumulative distribution function (CDF) of the standard normal distribution serves as the link function. This method is particularly useful for scenarios where the dependent variable is dichotomous, such as yes/no or success/failure outcomes.

**Characteristics of Probit Regression**

1. **Binary Dependent Variable**: Probit regression is designed for dependent variables that have two possible outcomes, typically coded as 0 and 1.
2. **Normal Distribution Link Function**: The model uses the CDF of the standard normal distribution as its link function, assuming the underlying latent variable follows a normal distribution.
3. **Probabilistic Interpretation**: The output of a probit model is a probability estimate for the binary outcome. This probability is derived by transforming the linear predictor through the normal CDF.
4. **Linear Predictor**: The model is based on a linear combination of the predictor variables, formulated as: P(Y=1)=Φ(β0+β1X1+β2X2+...+βnXn)P(Y=1) = \Phi(\beta_0 + \beta_1X_1 + \beta_2X_2 + ... + \beta_nX_n)P(Y=1)=Φ(β0+β1X1+β2X2+...+βnXn ) Here, Φ\PhiΦ is the standard normal CDF, β0\beta_0β0 is the intercept, and β1,β2,...,βn\beta_1, \beta_2, ..., \beta_nβ1,β2,...,βn are the coefficients for the predictor variables X1,X2,...,XnX_1, X_2, ..., X_nX1,X2,...,Xn.
5. **Interpretation of Coefficients**: The coefficients indicate the effect of a one-unit change in the predictor variable on the z-score of the latent variable, which is then converted to a probability.

6. **Maximum Likelihood Estimation (MLE)**: Probit models are typically estimated using MLE, which finds the parameter values that maximize the likelihood of observing the given data.
7. **No Need for Linear Relationship**: Probit regression does not require a linear relationship between the predictors and the outcome.
8. **Handling of Various Predictors**: The model can incorporate continuous, categorical, and binary predictors.
9. **Error Terms**: The error terms are assumed to follow a standard normal distribution.

**Advantages of Probit Regression**

1. **Handling of Binary Outcomes**: It is specifically designed for binary outcomes, providing a probabilistic estimate for the occurrence of an event.
2. **Normal Distribution Assumption**: The use of the standard normal CDF can be advantageous when the latent variables are assumed to follow a normal distribution.
3. **Interpretability**: Coefficients can be interpreted in terms of their effect on the z-score, related to the probability of the outcome.
4. **Modeling Flexibility**: Probit regression can be extended to handle ordinal outcomes (ordered probit) and other categorical data.
5. **Good Fit for Certain Data**: It may provide a better fit than logistic regression when the underlying latent variables influencing the binary outcome are normally distributed.

**Real-Life Examples of Probit Regression**

1. **Medical Research**: Used to model binary health outcomes, such as the presence or absence of a disease, based on predictors like age, lifestyle factors, and genetic markers.
2. **Toxicology Studies**: Analyzes dose-response relationships to determine lethal dose (LD50) levels by modeling the probability of an organism exhibiting a particular response to different levels of a toxic substance.
3. **Marketing and Consumer Behavior**: Predicts consumer choices and preferences, such as the likelihood of purchasing a product based on demographic characteristics and past purchasing behavior.
4. **Credit Scoring**: Assesses credit risk by analyzing variables such as income, employment status, and credit history to predict the probability of loan default.
5. **Political Science**: Studies voting behavior and election outcomes by understanding the impact of demographic and socio-economic variables on voting decisions.
6. **Sociological Research**: Analyzes binary outcomes in social science studies, such as the likelihood of participating in a social activity or achieving a certain educational level.
7. **Environmental Studies**: Models binary environmental outcomes, like the presence or absence of a species in a habitat based on environmental variables.
8. **Employee Turnover**: Predicts employee turnover by considering factors such as job satisfaction, salary, and work environment.

Probit regression is a powerful tool for modelling binary outcomes, offering clear probabilistic interpretations and the flexibility to handle various types of predictors. Its use of the standard normal CDF makes it particularly suitable for situations where the latent variables are assumed to follow a normal distribution. This method is widely applied in fields such as medical research, toxicology, marketing, finance, political science, sociology, environmental studies, and human resources, providing valuable insights and predictions for binary outcomes.