Assignment A3 (Part-B)

```
import pandas as pd
import numpy as np
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from statsmodels.discrete.discrete_model import Probit
import statsmodels.api as sm
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import matplotlib.pyplot as plt
```

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

```
<ipython-input-26-e985c838466c>:2: DtypeWarning: Columns (1) have mixed types. Specify dtype option on import or set low_memory=Fals
  df = pd.read_csv("NSSO68.csv")
```

	slno	grp	Round_Centre	FSU_number	Round	Schedule_Number	Sample	Sector	state	State_R
0	1	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
1	2	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
2	3	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
3	4	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
4	5	4099999999999999926524952937758 72.0	1	41000	68	10	1	2	24	
...	...	...	...	...	...	...	...	...	...	
101657	101658	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101658	101659	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101659	101660	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101660	101661	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	
101661	101662	7999999999999999708717035972 1984.0	1	79998	68	10	1	1	1	

101662 rows × 384 columns

```
# Create the Target variable
df['non_veg'] = np.where(df[['eggsno_q', 'fishprawn_q', 'goatmeat_q', 'beef_q', 'pork_q', 'chicken_q', 'othrbirds_q']].sum(axis=1) > 0,
df['non_veg']
```

```
0         1
1         1
2         1
3         1
4         1
         ..
101657    0
101658    1
101659    1
101660    0
101661    0
Name: non_veg, Length: 101662, dtype: int64
```

```
# Define dependent variable (y) and independent variables (X)
y = df['non_veg']
X = df[['HH_type', 'Religion', 'Social_Group', 'Regular_salary_earner', 'Possess_ration_card', 'Sex', 'Age', 'Marital_Status', 'Educatio
y.head()
X.head()
```

	HH_type	Religion	Social_Group	Regular_salary_earner	Possess_ration_card	Sex	Age	Marital_Status	Education	Meals_At_Home	I
0	2.0	1.0	3.0	1.0	1.0	1	50	2.0	8.0	59.0	
1	2.0	3.0	9.0	1.0	1.0	2	40	3.0	12.0	56.0	
2	1.0	1.0	9.0	1.0	1.0	1	45	2.0	7.0	60.0	
3	2.0	3.0	9.0	1.0	1.0	1	75	3.0	6.0	60.0	
4	1.0	1.0	9.0	2.0	1.0	1	30	2.0	7.0	59.0	

```
# Impute missing values with mean
imputer = SimpleImputer(strategy='mean')
X_imputed = imputer.fit_transform(X)
X_imputed
```

```
array([[2.0000e+00, 1.0000e+00, 3.0000e+00, ..., 5.0000e+00, 4.7510e+04,
        4.1100e+02],
       [2.0000e+00, 3.0000e+00, 9.0000e+00, ..., 2.0000e+00, 8.5102e+04,
        3.3100e+02],
       [1.0000e+00, 1.0000e+00, 9.0000e+00, ..., 5.0000e+00, 4.9219e+04,
        1.2100e+02],
       ...,
       [5.0000e+00, 1.0000e+00, 9.0000e+00, ..., 7.0000e+00, 4.1001e+04,
        9.2000e+02],
       [2.0000e+00, 1.0000e+00, 9.0000e+00, ..., 5.0000e+00, 4.7211e+04,
        5.2200e+02],
       [1.0000e+00, 1.0000e+00, 9.0000e+00, ..., 7.0000e+00, 1.1130e+03,
        6.1100e+02]])
```

```
# Ensure 'y' is a binary factor
y = y.astype('category')

# Scale numeric variables if needed (optional)
X_imputed_scaled = StandardScaler().fit_transform(X_imputed)
X_imputed_scaled
```

```
array([[-0.33833318, -0.4138791 , -0.47257591, ...,  0.18974726,
         0.28981662, -0.70113625],
       [-0.33833318,  1.34847893,  1.42994352, ..., -1.14520676,
         1.58409885, -1.02440146],
       [-0.80105208, -0.4138791 ,  1.42994352, ...,  0.18974726,
         0.34865702, -1.87297265],
       ...,
       [ 1.04982353, -0.4138791 ,  1.42994352, ...,  1.0797166 ,
         0.06571354,  1.35563868],
       [-0.33833318, -0.4138791 ,  1.42994352, ...,  0.18974726,
         0.27952214, -0.25260576],
       [-0.80105208, -0.4138791 ,  1.42994352, ...,  1.0797166 ,
        -1.30761934,  0.10702679]])
```

```
# Fit the probit regression model
probit_model = Probit(y, X_imputed_scaled)
probit_result = probit_model.fit(maxiter=1000)

# Print model summary
print(probit_result.summary())
```

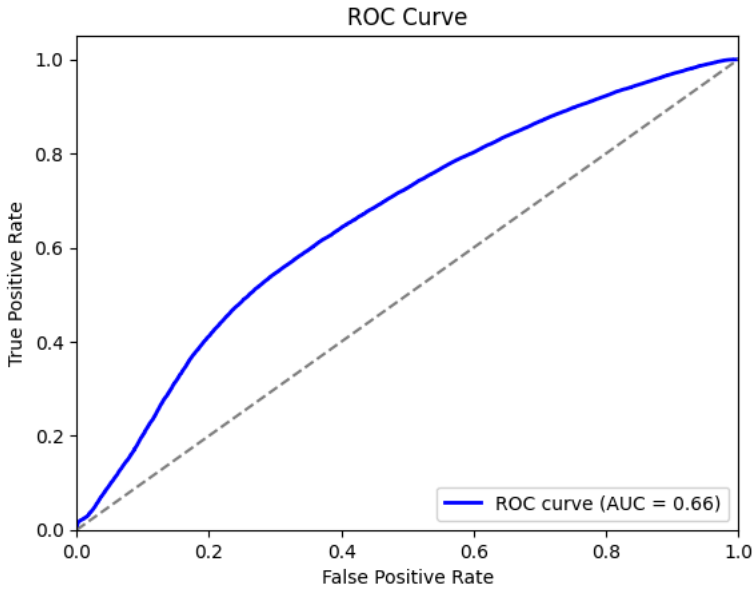
```
Optimization terminated successfully.
         Current function value: 0.663551
         Iterations 5
                          Probit Regression Results
==============================================================================
Dep. Variable:                non_veg   No. Observations:               101662
Model:                         Probit   Df Residuals:                   101648
Method:                           MLE   Df Model:                           13
Date:                Mon, 01 Jul 2024   Pseudo R-squ.:                -0.05190
Time:                        14:18:08   Log-Likelihood:                -67458.
converged:                       True   LL-Null:                       -64129.
Covariance Type:            nonrobust   LLR p-value:                    1.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.0285      0.004     -6.346      0.000      -0.037      -0.020
x2             0.1579      0.005     34.313      0.000       0.149       0.167
x3            -0.1254      0.004    -30.423      0.000      -0.133      -0.117
x4            -0.0135      0.005     -2.880      0.004      -0.023      -0.004
x5            -0.0198      0.004     -4.674      0.000      -0.028      -0.011
x6            -0.0475      0.005     -9.236      0.000      -0.058      -0.037
x7            -0.0176      0.005     -3.839      0.000      -0.027      -0.009
x8             0.0478      0.005      9.059      0.000       0.037       0.058
x9            -0.0322      0.005     -6.551      0.000      -0.042      -0.023
x10            0.1669      0.004     39.700      0.000       0.159       0.175
```

```
x11          -0.0968      0.004    -23.621      0.000      -0.105      -0.089
x12           0.0200      0.004      4.613      0.000       0.011       0.028
x13           0.0756      0.005     15.497      0.000       0.066       0.085
x14           0.0398      0.005      8.239      0.000       0.030       0.049
==============================================================================
```

```
# Predict probabilities
predicted_probs = probit_result.predict(X_imputed_scaled)

# Convert probabilities to binary predictions using a threshold of 0.5
predicted_classes = np.where(predicted_probs > 0.5, 1, 0)

# Confusion Matrix
cm = confusion_matrix(y, predicted_classes)
print("Confusion Matrix:")
print(cm)

# Classification Report
print("Classification Report:")
print(classification_report(y, predicted_classes))
```

```
Confusion Matrix:
[[22233 10839]
 [29282 39308]]
Classification Report:
              precision    recall  f1-score   support

           0       0.43      0.67      0.53     33072
           1       0.78      0.57      0.66     68590

    accuracy                           0.61    101662
   macro avg       0.61      0.62      0.59    101662
weighted avg       0.67      0.61      0.62    101662
```

```
# Calculate ROC curve and AUC
fpr, tpr, thresholds = roc_curve(y, predicted_probs)
roc_auc = auc(fpr, tpr)

# Plot ROC curve
plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (AUC = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='gray', 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:.2f}")
```




```
AUC: 0.66
```

```
Start coding or generate with AI.
```