A Short Monograph on Logistic Regression

To serve as a Refresher for PGP-DSBA

Index

Contents

List of Figures

List of Tables

1. Introduction

In the monographs on regression and predictive modelling the problem of prediction is emphasized. Based on the historical data, accurate estimation of future response is an important problem. However, so far only the case of a continuous response has been considered.

In reality response can be continuous or categorical.

When an applicant comes to a bank for loan, the bank asks for a lot of information on the applicant; such as, age, salary, marital status, educational background and qualification, any other existing loan, if yes, then monthly payment, number of children and many other facts. Based on this information the bank takes a decision regarding whether the loan may be approved or not. In this case, approval is the response which may take only two values *Approved* or *Rejected*. Clearly this is a case where the response is binary. The standard linear regression modelling will not work here. Binary response is a very common phenomenon. In financial domain credit card default or loan default is a major source of uncertainty. Credit card companies would like to predict, given a person's earlier behaviour, whether s/he would be able to pay the minimum amount next month. E-commerce companies would like to predict whether a prospective buyer would close a deal with them, given her browsing pattern. Algorithms are developed to classify an email as spam and direct it away from inbox. In all such cases, and in many more, response has only two levels, denoted by Yes (Success) or No (Failure).

In this monograph we deal with binary response only and the technique developed will be logistic regression. Note that, by no means, response needs to be restricted to two levels nor logistic model is the only method of binary prediction. In healthcare domain, given a patient's risk profile, a physician would like to categorize him as low risk, medium risk or high risk. Response here has multiple ordinal levels. Several extensions of logistic regression have been mentioned in Section 4.

Logistic regression may also be taken as a classification algorithm, since it assigns a class label to each observation. The underlying technique of classification is based on regression and a misclassification probability can also be calculated.

Logistic regression is a modelling technique for a binary response. Unlike linear regression, the value of the response is not predicted. If the two response levels are taken as Success and Failure, Probability(Success) is predicted based on the values of the predictors.

2. Linear Regression versus Logistic Regression

Let us recapitulate briefly the basic premises of a multiple linear regression.

Assume *n* (multivariate) sample observations $(x_{1i}, x_{2i}, \dots x_{ki}, y_i)$, *i = 1, 2, …, n,* are available. Y denotes the dependent variable and $X_1, X_2, \dots, X_k$ the independent variables.

The mathematical formulation of multiple linear regression line is:

$$E(Y) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_k X_k$$

where,

Y: is the value of the continuous response (or dependent) variable,

β_0 : is the intercept

X_j: represents the j^{th} independent (predictor) variable continuous in nature. *j = 1, …, k*

β_j: represents the coefficient of the j^{th} independent (predictor) variable.

It is assumed that the expected value of the response is a linear function of all the *k* predictors.

At the observation level, the model may be rewritten as

$$Y_i = \beta_0 + \beta_1 X_{1i} + \beta_2 X_{2i} + \cdots + \beta_k X_{ki} + \varepsilon_i, \quad i = 1, 2, \dots, n$$

ε: represents the unobservable error term and follows N(0, σ^2)

Scatterplots are useful tools in case of multiple regression since scatterplot of Y versus each predictor X_j demonstrates the strength of dependency of the response on that predictor.

The main differences between linear and logistic regression are

I. Correlation between a binary response and a continuous predictor is not defined. Hence scatterplots are not useful in this case

II. The numerical value assigned to the two levels of a binary response is arbitrary. Consider, for example, the case of loan default. Physically the two levels are defaulters and non-defaulters. It is possible to assign values 0 and 1 respectively to the two levels; it is also possible to assign 1 and 0; or 1 and -1; or any other arbitrary set of numbers. Hence it is not possible to regress *Y* on the predictors.

III. Therefore, Prob (*Y* = 1), i.e. probability of *Y* at a given level, is modelled through a regression.

IV. Since probability of an event is a number between 0 and 1, linear relationship between Prob (*Y* = 1) and the set of predictors is not possible. Hence a suitable transformation is necessary.

3. Formal Approach to Logistic Regression

3.1 Odds, Log Odds and Logistic (Logit) Transform

The concept of odds ratio is related to probability. Formally

$$\text{Odds of Success} = \frac{Prob(Success)}{1 - Prob(Success)}$$

If success and failure have equal probability, then the numerical value of odds is 1. As success probability increases, odds increases. While probability is a number between 0 and 1, there is no such restriction on odds. It is a positive number, taking a very small value when success probability is small but may have a high numerical value if success probability is large. Theoretically, odds is a number between (0, +∞).

Another related quantity is called log odds which is defined as $\log_e Odds$. Since odds is a positive quantity, log odds is always defined, but its value may be positive or negative. When value of odds is 1, log odds is 0. When the two outcomes, success and failure, are equiprobable, i.e., each probability is 50%, odds is 1 and log odds is 0. When probability of success is less than 50%, odds is less than 1 and log odds is a negative quantity. When probability of success is more than 50%, the value of odds is more than 1 and log odds is a positive quantity. Theoretically, log odds is a number between (-∞, +∞).

In logistic regression the log odds of success is modelled as a linear function of the predictors. Log odds of success is also known as logit transformation. Mathematically it may be claimed that a logit transform maps a number between 0 and 1 on a real line; logistic transformation is its reverse where a real number is converted into a probability. This is the genesis of the name logistic regression.

Case Study 1:

Physicians would like to identify women with higher risk of turning into diabetic, given their health, family background and lifestyle. Seven hundred and twenty-nine (729) females from different age and ethnicity are screened for their diabetes status. Data on the following attributes was collected.

Age: Chronological age
Pregnancy: Number of pregnancies including those not resulting in live births
Glucose: Blood glucose level
BMI: Body mass index defined as weight/height^2
Diabetes Pedigree Function: Probability of developing type II diabetes, calculation based on family diabetic history
Race: White, Black or Hispanic
Outcome: Currently diagnosed as diabetic or not (1: Diabetic, 0: Non-diabetic)

Statement of the Problem: Develop a logistic regression model to predict whether a woman will develop type II diabetes given her health and genetic information.

Exploratory Analysis (EDA) on Diabetes Data

The Diabetes data is uploaded in Python

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

diabetes = pd.read_csv('diabetes.csv')

diabetes.shape
```

```
(729, 8)
```

```
diabetes.columns
```

```
Index(['Pregnancies', 'Glucose', 'BloodPressure', 'BMI',
       'DiabetesPedigreeFunction', 'Age', 'Outcome', 'Race'],
      dtype='object')
```

```
diabetes.head(6)
```

```
Pregnancies Glucose   BP      BMI     DPF     Age  Outcome    Race

0       6       148     72      33.6    0.627   50      1       White

1       1       85      66      26.6    0.351   31      0       White

2       8       183     64      23.3    0.672   32      1       White

3       1       89      66      28.1    0.167   21      0       White

4       0       137     40      43.1    2.288   33      1       White

5       5       116     74      25.6    0.201   30      0       White
```

**DiabetesPedigreeFunction abbreviated as DPF & BloodPressure as BP*

The data has 729 observations and 8 attributes. The response is Outcome where 1 indicates the woman is diabetic and 0 indicates that she is not. The variable Race is categorical.

The 5-number summary for the continuous variables is given below. For Outcome and Race the proportions are shown.

```
# Summary of all variables

round(diabetes.drop('Outcome',axis=1).describe(),2).T
```

```
               count  mean   std    min    25%    50%    75%    max

Pregnancies    729.0  3.86   3.36   0.00   1.00   3.00   6.00   17.00

Glucose        729.0  121.41 31.18  44.00  99.00  117.00 141.00 199.00

BloodPressure  729.0  72.37  12.38  24.00  64.00  72.00  80.00  122.00

BMI           729.0  32.47  6.89   18.20  27.50  32.40  36.60  67.10

DPF           729.0  0.47   0.33   0.08   0.24   0.38   0.63   2.42

Age           729.0  33.32  11.75  21.00  24.00  29.00  41.00  81.00
```

**DiabetesPedigreeFunction abbreviated as DPF*

```
diabetes.Outcome.value_counts()
0     478
1     251
Name: Outcome, dtype: int64
round(diabetes.Outcome.value_counts(1),2)
0     0.66
1     0.34
Name: Outcome, dtype: float64
diabetes.Race.value_counts()

Hispanic    277
Black       247
White       205
Name: Race, dtype: int64
round(diabetes.Race.value_counts(1),2)
Hispanic    0.38
Black       0.34
White       0.28
Name: Race, dtype: float64
```

Among 729 females, 34% are diabetic. The highest proportion of females are Hispanic (38%), while Black (34%) and Whites (28%) make up the rest.

More visualizations are given below.

```
# Histogram of a few select variables
fig = plt.figure(figsize = (10,8))
ax = fig.gca()
diabetes.iloc[:,1:5].hist(ax = ax)
plt.show()
```

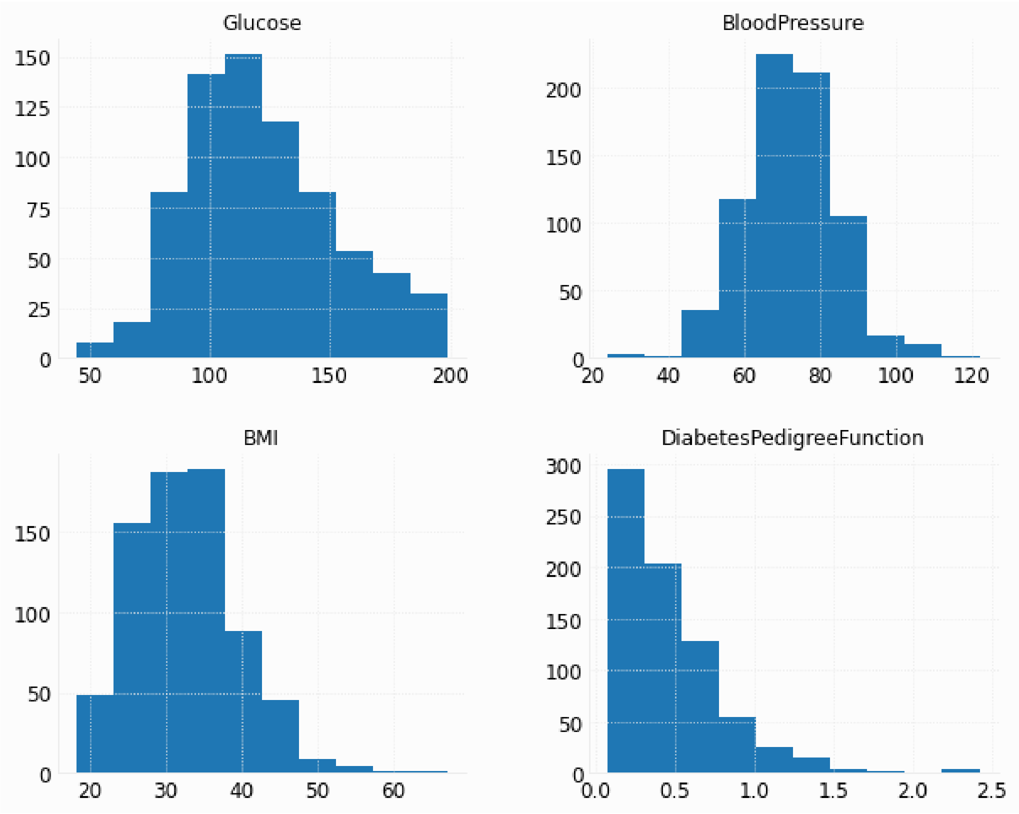


Figure 1- Histograms of continuous predictors: Set 1

```
fig = plt.figure(figsize = (8,5))
ax = fig.gca()
diabetes[['Age','Pregnancies']].hist(ax = ax)
plt.show()
```

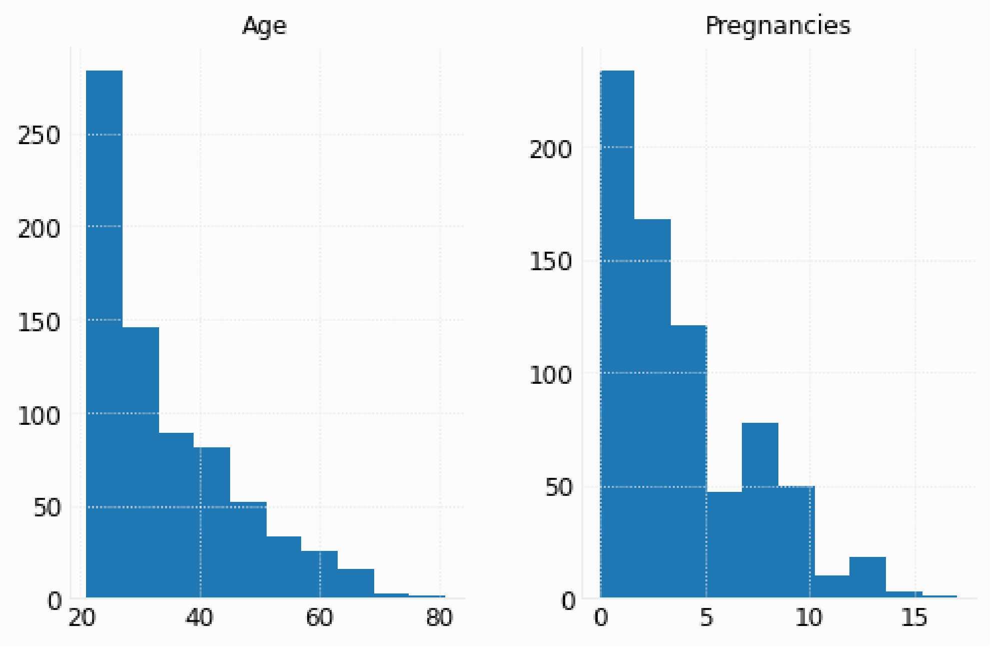


Figure 2- Histograms of continuous predictors: Set 2

```
diabetes.iloc[:,1:5].boxplot(figsize = (20,5))
plt.show()
```

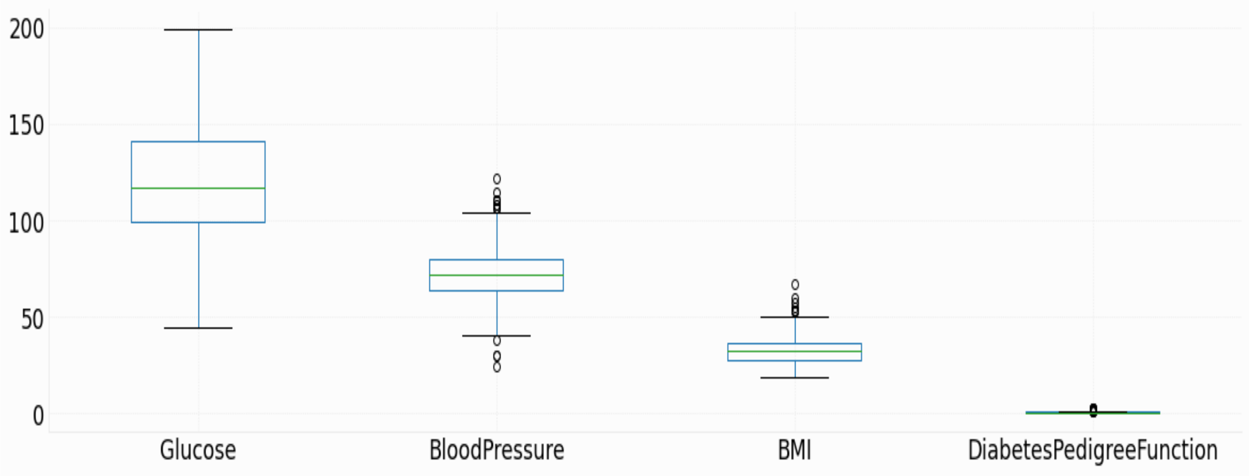


Figure 3- Boxplots of continuous predictors: Set 1

```
diabetes[['Age','Pregnancies']].boxplot(figsize = (10,5))
plt.show()
```

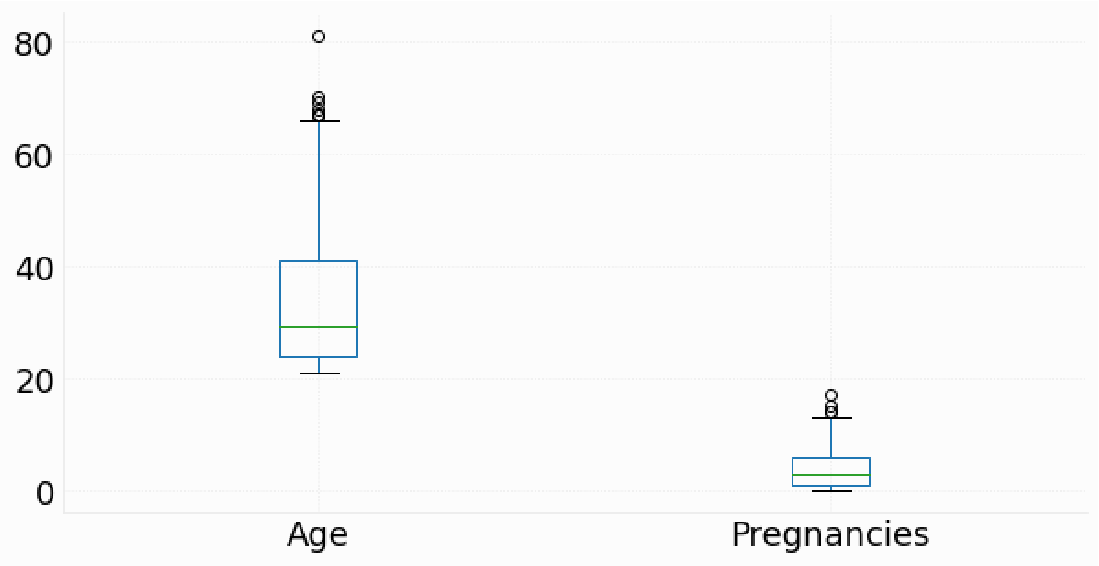


Figure 4- Boxplots of continuous predictors: Set 2

Following Fig. 1 – 4, the variables BP and Glucose may be considered to have symmetric distributions, the rest are clearly positively skewed.

It is also important to investigate the proportion of diabetics among different races.

```
# Cross-table of Race and Outcome

diabetes.groupby(['Race','Outcome'])['Age'].count().unstack()
```

```
Outcome      0     1

Race

Black        156   91

Hispanic     190   87

White        132   73
```

```
round(diabetes.groupby(['Race'])['Outcome'].value_counts(1),2).unstack()
```

```
Outcome      0     1

Race

Black        0.63  0.37

Hispanic     0.69  0.31

White        0.64  0.36
```

It seems that among Hispanics the proportion of diabetic women is lower, compared to the other races. (This observation is contrary to a few other studies.)

3.2 Simple Logistic Regression

Contrary to the simple linear regression, neither correlation coefficient nor scatterplot will be useful to detect dependency of the response on the predictor(s). Recall that correlation coefficient is defined only when both the variables are continuous. It is possible to use rank correlation coefficient between two ordinal variables, but no correlation measure may be defined when at least one variable is binary or even nominal.

In case of logistic regression, the response *Y* is always a nominal variable. Hence no correlation measure can be defined between the response and any of the predictors, be they continuous, nominal or ordinal.

It is possible to create a scatterplot of Y and each of the predictor variables, but the plots will be totally non-informative. One example is shown below.

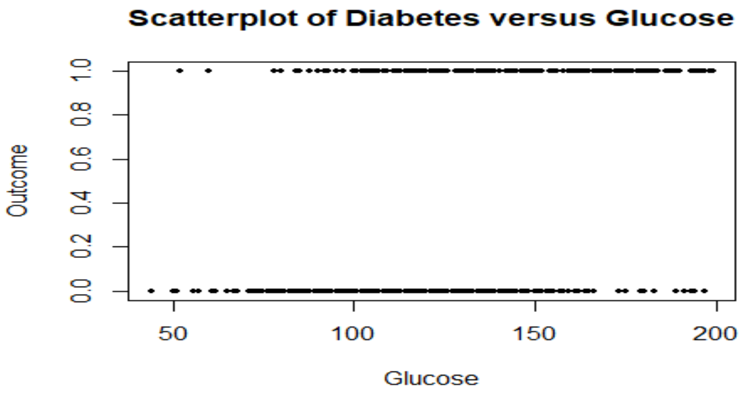


Figure 5- Example of a non-informative scatterplot

It is therefore clear that direct modelling of the binary response is not possible. The log odds ratio is one of the most useful transformations of the response (two other transforms are mentioned in Sec 4) that is modelled as a linear function of the predictors. Log odds transformation is also known as a logit or a logistic transformation.

The form of a simple logistic regression is

$$logit(Y_i|X_i) = \log\frac{\Pr(Y_i=1)}{\Pr(Y_i=0)} = \beta_0 + \beta_1 X_i \quad i = 1, 2, \ldots, n$$

where X represents the independent (predictor) variable, and β_1 represents the associated regression coefficient.

Note that log is taken with base *e*. Hence

$$\frac{\Pr(Y_i=1|X_i)}{\Pr(Y_i=0|X_i)} = e^{\beta_0 + \beta_1 X_i}$$

and probability of success may be expressed as

$$Pr(Y_i = 1|X_i) = \frac{e^{\beta_0 + \beta_1 X_i}}{1+e^{\beta_0 + \beta_1 X_i}}$$

This is known as a logistic function. When there is no possibility of any confusion, often the level X_i of the predictor is omitted.

Y_i can take only two values, success (1) and failure (0). It follows a special type of binomial distribution, known as Bernoulli distribution, where the number of trial is 1. The most important observation is that success probability is a function of the predictor variables. But this is not a linear function.

Probability, being a number between 0 and 1, cannot be modelled as a linear regression function. Linearity is unbounded and for certain values of predictors, estimated probability may either be less than 0 or greater than 1. Logistic function ensures that, for no predictor combination, probability goes outside of [0, 1] boundary. Fig 6 shows the form of logistic function.

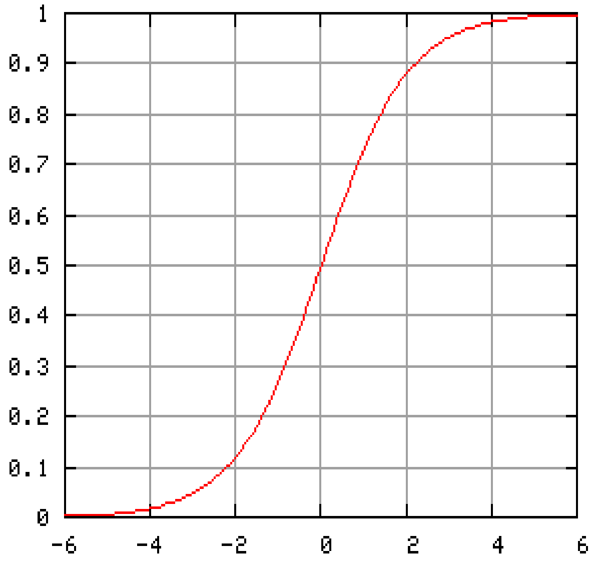


Figure 6- Example of a logistic curve with positive slope

X-axis of Fig 6 contains values of a single predictor variable and Y-axis contains the values of $Pr(Y_i = 1)$. As the value of the predictor increases, probability increases from 0 to 1. The function is defined such that, if the predictor takes smaller values less than a certain threshold (e.g. - 5, in this case) its value is 0; and if the predictor takes values larger than another threshold (e.g. +5, in this case) its value is 1.

If the slope is negative, with increasing value of the predictor, the logistic function goes from 1 to 0.

Case Study continued: Illustration 1

Let us consider a logistic regression of diabetes on race. The observed odds and log odds are given below. In this case Success is defined when a woman is found to be Diabetic and the code for that is assumed to be Y = 1.

Table 1- Diabetes versus Race comparison

Race	Diabetic (Y = 1)	Total	Prop (Success)	Odds (Success)	Log Odds (Success)
Black	**91**	**247**	**0.37**	**0.58**	**-0.54**
Hispanic	**87**	**277**	**0.31**	**0.46**	**-0.78**
White	**73**	**205**	**0.36**	**0.55**	**-0.59**

Need to fit a logistic regression model to estimate log odds of success based on only one predictor, race.

```
import statsmodels.formula.api as sm
```

```
X=pd.get_dummies(diabetes.Race,drop_first=True)
y=np.asarray(diabetes.Outcome)


glm = sm.logit('Outcome~Race',data=diabetes).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.642553
         Iterations 5
```

```
print(glm.summary())
```

```
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  729
Model:                          Logit   Df Residuals:                      726
Method:                           MLE   Df Model:                            2
Date:                Wed, 19 Aug 2020   Pseudo R-squ.:                0.002016
Time:                        14:20:21   Log-Likelihood:                -468.42
converged:                       True   LL-Null:                       -469.37
Covariance Type:            nonrobust   LLR p-value:                    0.3881
==============================================================================
                       coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept           -0.5390      0.132     -4.086      0.000      -0.798      -0.280
Race[T.Hispanic]    -0.2421      0.185     -1.310      0.190      -0.604       0.120
Race[T.White]       -0.0533      0.197     -0.271      0.786      -0.439       0.332
==============================================================================
```

Several important observations need to be mentioned.

Because of the model not being a linear one, there is no closed form solution for the regression coefficients, like linear models. The method of estimation follows an iterative process called Fisher Scoring Algorithm. It is important that the algorithm converges. In the current situation convergence took place in 5 iterations.

If the algorithm does not converge, there will be an explicit message regarding that. If convergence is not achieved, the estimates of regression coefficients are not valid.

The regression parameter estimates for GLM are maximum likelihood estimates, and not least squares estimates.

Note also that significance test for regression coefficients is done through z-test, and not t-test.

For the current example, Race has 3 levels, Hispanic, White and Black. By default, Race = Black has been taken to be the baseline level and the coefficients corresponding to the other levels signify whether they are different from the baseline. Since the p-values for both levels are not significant at 5% level, it may be concluded that Race does not have an impact on the probability of having diabetes.

[*A note on baseline: By default, the categorical attributes (and responses) are coded in such a way that the columns are in lexicographic ordering. E.g., for Race, the first column is Black,* *second Hispanic and third White. If the levels were Asian, Black and Hispanic, Asian would be the first column. Hence when the first column is dropped in this example, naturally Black is the baseline.*

For nominal variables the choice for baseline may not be very important, but for ordinal variable either the lowest rank-ordered or the highest rank-ordered level is typically the choice for baseline. An easy trick is shown with the second case study]

Nevertheless, as an illustration, let us estimate the probabilities of diabetes from the estimated regression coefficients.

The logistic regression equations are

$$\text{Logit}(Y = 1 |\ \text{Race} = \text{Black}) = -0.539$$
$$\text{Logit}(Y = 1 |\text{Race} = \text{Hispanic}) = -0.539 - 0.242 = -0.781$$
$$\text{Logit}(Y = 1 |\ \text{Race} = \text{White}) = -0.539 - 0.053 = -0.592$$

Hence

Probability that a black woman is diabetic $\Pr(Y = 1 |\ \text{Black}) = \frac{\exp(-0.539)}{1+\exp(-0.539)} = 0.37$

Probability that a hispanic woman is diabetic $\Pr(Y = 1 |\ \text{Hispanic}) = \frac{\exp(-0.781)}{1+\exp(-0.781)} = 0.31$

Probability that a white woman is diabetic $\Pr(Y = 1 |\ \text{White}) = \frac{\exp(-0.781)}{1+\exp(-0.781)} = 0.36$

Note that, in this case the model gives exact fit (refer to Table 1), which will not be the case in general. In fact, exact fit, or overfit, is not desirable, because that does not leave any degree of freedom for significance test. *Exact fit provides poor prediction for future data.*

Case Study continued: Illustration 2

Let us now develop a logistic regression model of diabetes on glucose level. Since glucose is a continuous predictor, no contingency table is possible.

```
glm1 = sm.logit('Outcome~Glucose',data=diabetes).fit()
Optimization terminated successfully.
         Current function value: 0.522284
         Iterations 6
print(glm1.summary())
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  729
Model:                          Logit   Df Residuals:                      727
Method:                           MLE   Df Model:                            1
Date:                Wed, 19 Aug 2020   Pseudo R-squ.:                  0.1888
Time:                        14:20:22   Log-Likelihood:                -380.74
converged:                       True   LL-Null:                       -469.37
Covariance Type:            nonrobust   LLR p-value:                 1.936e-40
```

```
======================================================================
                  coef    std err      z     P>|z|      [0.025     0.975]
----------------------------------------------------------------------
Intercept      -5.4043     0.431    -12.538  0.000      -6.249     -4.559
Glucose         0.0380     0.003     11.463  0.000       0.032      0.045
======================================================================
```

The iteration has converged and the logistic regression is

$$\text{Logit}(Y = 1|\ \text{glucose}) = -5.40 + 0.038 * \text{glucose}$$

The regression coefficient is highly significant as the p-value is very small.

In the data, the range of glucose values is [44, 199]. It is always a good practice not to extrapolate too far beyond the given range. Hence

Probability that a woman with glucose level 50 will be diabetic is estimated to be

$$\Pr(Y = 1|\ 50) = \frac{\exp(-5.40 + 0.038 * 50)}{1+\exp(-5.40 + 0.038 * 50)} = 0.029$$

Probability that a woman with glucose level 100 will be diabetic is estimated to be

$$\Pr(Y = 1|\ 100) = \frac{\exp(-5.40 + 0.038 * 100)}{1+\exp(-5.40 + 0.038 * 100)} = 0.168$$

Probability that a woman with glucose level 150 will be diabetic is estimated to be

$$\Pr(Y = 1|\ 150) = \frac{\exp(-5.40 + 0.038 * 150)}{1+\exp(-5.40 + 0.038 * 150)} = 0.57$$

As glucose level increases, probability of being diabetic also increases, but not linearly. The logit function is increasing linearly and in this case with every unit change in glucose level, the increment will be 0.038.

Given a glucose level, odds of a woman being diabetic is $exp(-5.40 + 0.038 * \text{glucose})$. For every unit of increase in glucose level, odds of being diabetic increases by $exp(0.038) = 1.04$.

The following figure shows the relationship between probability of being diabetic with increasing blood glucose level.

```
newvalue=np.arange(40,250)
newdata=pd.DataFrame(newvalue,columns=['Glucose'])
predicted_values=glm1.predict(newdata)
sns.lineplot(newdata.Glucose,predicted_values)
plt.title('Estimated Probability Curve')
plt.ylabel('Probability')
plt.show()
```

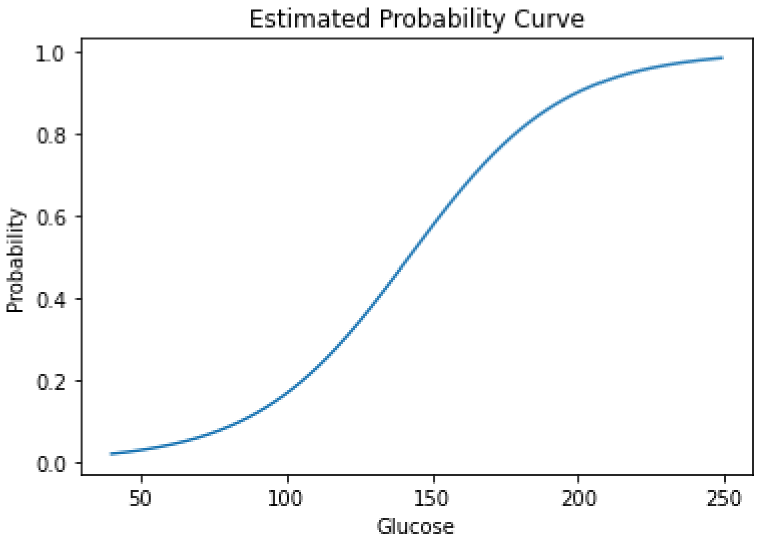


Figure 7- Logistic regression outcome of diabetes on glucose

Fig. 7 clearly shows the sigmoidal curve. Towards the boundaries, the probability curve is parallel to x-axis. That indicates that for smaller or larger values of blood glucose the probability of being diabetic will be 0 or 1 respectively. In practice this means that probability will be very small or almost certain. The boundedness of the probability function is thus established. Between blood glucose levels [100, 200], the probability of being diabetic rises sharply.

From the graph it is also possible to identify at which blood sugar level, say probability of being diabetic is 25% or 50%. Approximately these two thresholds are 120 and 150 respectively.

3.3 Deviance

Logistic regression model is part of a larger class of models, call the Generalized Linear Model (GLM). Linear regression model is also a member of this class of models; but its properties are so simple, inference problems can be handled easily. Because of their simplicity,

i) parameter estimation is done through least square estimates, which are identical to maximum likelihood estimates
ii) significance testing for parameters can be done through a t-test
iii) a goodness-of-fit measure, R^2, is available directly
iv) a residual measure corresponding to every level of the predictor combination is available directly

***None of the above needs to be true for a GLM*.**

For logistic regression, as mentioned before, parameters are estimated through maximum likelihood procedure and no closed form solution is available. Neither a direct residual measure nor R^2 is available. The appropriateness of a logistic regression model is measured through a *deviance* statistic.

Simply put, deviance of a model is the difference between the log-likelihood of a *saturated* model and the model under consideration.

A saturated model is the most general model, containing as many parameters as the number of observations. This model gives a perfect fit to the observations. In other words, for a saturated model, the observed and predicted values are identical for all observations in the data set. A saturated model will have the maximum possible number of parameters, given a set of observations.

Let the value of the likelihood of the observed data, under the saturated model be defined as L_S.

Now consider an alternative model, M_1 and the corresponding likelihood value be L_{M1}.

Deviance for M_1 is defined as $D_{M1} = -2(\log L_{M1} - \log L_S)$

Since L_{M1} must always be less than L_S, deviance is always a positive quantity. Naturally, the smaller the value of deviance, the better is the fit of the model.

To put it differently, the larger the log-likelihood, the better is the model

Deviance statistic follows a χ^2 distribution with appropriate degrees of freedom.

A heuristic explanation of deviance may be given in this manner. A model is a simplification of reality, which provides a simpler but working structure on the data. If the simplification works, the model may be considered appropriate. If a model works, it will be able to provide an adequate prediction for the observations, whereas a saturated model will provide an exact prediction. Hence the difference between the saturated model prediction and alternate model prediction must be small.

Another related concept is the null deviance. A null model is an intercept only model. If such a model is defined as L_N, then null deviance is $D_N = -2(\log L_N - \log L_S)$. The intercept only model is not expected to fit any data. Therefore, it is reasonable to expect that D_{M1} will be smaller than D_N. If the difference is significant, then the model may be considered as significantly different from the null model. This statistic is similar to the overall F test of linear regression.

However, Python does not provide the value of the deviance statistic explicitly, neither does it show the log-likelihood for the saturated model. It gives the log-likelihood for the null model and the log-likelihood for the fitted model. It is important to note that, given a set of observations, saturated log-likelihood and null log-likelihood are constant, just as in case of linear regression the total sum of squares for a given data set is constant.

Let us look at the illustrations above.

Sample size is 730. Hence total degrees of freedom in the model is $n - 1 = 729$. Null model involves only a single parameter, the intercept. Hence degrees of freedom for the null model is $n - 1 - 1 = 728$.

Comparison of residual deviance between the null model and proposed model indicates whether the proposed model fits the data.

D_N - D_{Race} = $-2(\log L_N - \log L_R) = -2(-469.37 - (-468.42))$=938.74 – 936.84 = 1.9 on 2 df. which is not significant.

D_N - $D_{Glucose}$ = $-2(\log L_N - \log L_G) = -2(-469.37 - (-380.74))$=938.74 – 761.49 = 177.25 on 1 df. This is highly significant.

3.4 Multiple Logistic Regression

Finally let us fit a logistic regression model to predict diabetes based on all the predictors in the data.

```
glm2 = sm.logit('Outcome~Pregnancies+Glucose+BloodPressure+BMI+DiabetesPedi
greeFunction+Age+Race',data=diabetes).fit()
Optimization terminated successfully.
         Current function value: 0.466833
         Iterations 6
print(glm2.summary())
                           Logit Regression Results
==============================================================================
Dep. Variable:                Outcome   No. Observations:                  729
Model:                          Logit   Df Residuals:                      720
Method:                           MLE   Df Model:                            8
Date:                Wed, 19 Aug 2020   Pseudo R-squ.:                  0.2749
Time:                        14:20:22   Log-Likelihood:                -340.32
converged:                       True   LL-Null:                       -469.37
Covariance Type:            nonrobust   LLR p-value:                 3.312e-51
==============================================================================
                 coef    std err      z          P>|z|     [0.025       0.975]
------------------------------------------------------------------------------
Intercept       -8.8345 0.824    -10.720       0.000     -10.450       -7.219
Race[T.Hispanic]-0.1950 0.225     -0.868       0.385      -0.635        0.245
Race[T.White]    0.0467 0.243      0.192       0.848      -0.430        0.523
Pregnancies      0.1170 0.033      3.511       0.000       0.052        0.182
Glucose          0.0336 0.004      9.524       0.000       0.027        0.040
BloodPressure   -0.0084 0.009     -0.978       0.328      -0.025        0.008
BMI              0.0933 0.016      5.947       0.000       0.063        0.124
DPF              0.9465 0.306      3.091       0.002       0.346        1.547
Age              0.0184 0.010      1.877       0.060      -0.001        0.038
==============================================================================
*DiabetesPedigreeFunction abbreviated as DPF


diabetes_1=diabetes.copy()
diabetes_1.Race=pd.Categorical(diabetes_1.Race).codes
diabetes_1.head()
```

```
def vif_cal(input_data):
    x_vars=input_data
    xvar_names=input_data.columns
    for i in range(0,xvar_names.shape[0]):
        y=x_vars[xvar_names[i]]
        x=x_vars[xvar_names.drop(xvar_names[i])]
        rsq=sm.ols(formula="y~x", data=x_vars).fit().rsquared
        vif=round(1/(1-rsq),2)
        print (xvar_names[i], " VIF = " , vif)

vif_cal(input_data=diabetes_1.drop(['Outcome'],axis=1))
Pregnancies  VIF =  1.46
Glucose  VIF =  1.16
BloodPressure  VIF =  1.25
BMI  VIF =  1.16
DiabetesPedigreeFunction  VIF =  1.05
Age  VIF =  1.63
Race  VIF =  1.01
```

The model converges after 6 iterations. The model is significant with model deviance (938.74 – 680.64 =) 258.1 on 8 df. Race is not significant, as noted before. Blood Pressure also turns out to be non-significant. The sign for this predictor happens to be negative, indicating that as blood pressure increases, probability of diabetes decreases, which is a medically invalid proposition. However, this is not of concern in this case, since the predictor is not found to be significant.

As in the case of linear regression, predictors must be investigated for multicollinearity. In this case, multicollinearity is not a problem as all VIF values are around 1.

The final recommended model is therefore

```
glm3 = sm.logit('Outcome~Pregnancies+Glucose+BMI+DiabetesPedigreeFuncti
on+Age',data=diabetes).fit()

print(glm3.summary())
Optimization terminated successfully.
         Current function value: 0.468433
         Iterations 6
                            Logit Regression Results
=======================================================================
Dep. Variable:                Outcome   No. Observations:           729
Model:                          Logit   Df Residuals:               723
Method:                           MLE   Df Model:                     5
Date:                Wed, 19 Aug 2020   Pseudo R-squ.:           0.2725
Time:                        15:57:33   Log-Likelihood:         -341.49
converged:                       True   LL-Null:                -469.37
Covariance Type:            nonrobust   LLR p-value:          3.191e-53
=======================================================================
                  coef    std err     z    P>|z|   [0.025      0.975]
-----------------------------------------------------------------------Inte
rcept       -9.2393  0.747   -12.365  0.000    -10.704     -7.775
Pregnancies     0.1162  0.033     3.513   0.000     0.051      0.181
Glucose         0.0332  0.003     9.548   0.000     0.026      0.040
BMI             0.0890  0.015     5.939   0.000     0.060      0.118
DPF             0.9720  0.303     3.205   0.001     0.378      1.566
Age             0.0156  0.009     1.649   0.099    -0.003      0.034
===========================================================================
*DiabetesPedigreeFunction abbreviated as DPF
```

The model converges after 5 iterations. The regression model is significant with deviance value ((938.74 – 682.97 =) 255.77 on 5 df.

Note that the variable Age has been retained in the model, even though it is not significant at 5% level. However, at the model building stage, often it is prudent to retain predictors that are borderline significant. From medical consideration too, Age is an important variable for diabetes onset.

All regression coefficients are positive, indicating that probability of diabetes is positively associated with all the predictors.

Since probability is not linearly related to the predictors, it is customary to provide interpretation of logistic regression through the odds ratios.

For one more pregnancy, odds of being diabetic increases by $e^{0.12}$ = 1.123
For one unit increase in glucose level, odds of being diabetic increases by $e^{0.03}$ = 1.033
For one unit increase in BMI, odds of being diabetic increases by $e^{0.089}$ = 1.09
For one unit increase in diabetic pedigree function, odds of being diabetic increases by $e^{0.97}$ = 2.64
For one more year of age, odds of being diabetic increases by $e^{0.01}$ = 1.01

3.5 Akaike Information Criteria (AIC)

Once a model is tentatively applied, the next question is how good the model is. Often there are multiple working models and a decision needs to be taken regarding which model will be the most suitable, i.e. which model explains the variability in the data the most or which model will have a higher predictive power. In case of linear regression, R^2 or adjusted R^2 are frequently used to determine that. In case of logistic regression (and for more complex models, such as time series models), a decision may be based on the information criteria. It is a function of the likelihood function, adjusted for the number of parameters in the model.

The most commonly used information criterion is proposed by a Japanese Statistician H. Akaike in the early 1970's. This is defined as

$$AIC(M) = -2(\log L_M) + 2p$$

where M is the model under consideration and p is the number of parameters in the model.

Recall that AIC (and Bayesian Information Criteria, BIC) was introduced in Monograph on Regression Model Selection (Section 3). The formula given there is a simplification of the more general formula presented here, the simplification being possible because of linearity.

AIC compares relative quality of several models, but does not say anything about the actual model quality, like R^2 does. This is an estimate of the out-of-sample prediction error, thereby eliminates models that overfit. It can be shown mathematically that AIC is equivalent to a leave-one-out cross-validation, when the sample size is infinitely large. The smallest value of AIC among the competing models is the most preferred.

Case Study continued:

Let us compare the AIC values of the models that have already been applied to Diabetes data.

```
AIC (Model 1: only Race) = -2(-468.42)+2*3 = 942.84
AIC (Model 2: only Glucose) = -2(-380.74)+2*2 =
765.49

AIC (Model 3: all predictors) = 698.64
AIC (Final Model) = 694.97
```

Model 1 with Race at three levels was a saturated model. The AIC value is the highest for that model. Model 2 has a much lower AIC, hence that model is definitely preferable over Model 1.

Recall that Model 3 had several predictors which turned out to be non-significant. That is why the value of AIC is higher than that of the Final Model.

3.6 Pseudo-R^2 Statistics

There is no direct measure of goodness of fit for a logistic regression. For a linear regression the total sum of squares and the residual sum of squares are two well defined quantities. In case of logistic regression, these are not available. Hence it is difficult to quantify for a proposed logistic model, how much of the total variability in the data, it is able to explain.

A few alternative quantifications similar, but not identical, to R^2 statistic have been proposed for logistic regression model assessment. Two of them are more popular among them, namely McFadden R^2 and Nagelkerke R^2.

McFadden $R^2 = 1 - \frac{log\ L_M}{log\ L_N}$,
where the numerator is the model log-likelihood and the denominator is the log-likelihood of the null (intercept only) model. This quantity measures the improvement over the null model. This statistic does not achieve 1 as the maximum value.

Nagelkerke R^2 is also a function of the two log-likelihoods but has a complex form. This quantity has a range between 0 and 1.

Case Study continued:

```
# Calculate McFadden R-square

print('McFadden Psuedo R Squared (Model 1: only Race) =',round(glm.prsq
uared,2))
print('McFadden Psuedo R Squared (Model 2: only Glucose) =',round(glm1.
prsquared,2))
print('McFadden Psuedo R Squared (Final Model) =',round(glm3.prsquared,
2))
```

```
McFadden Psuedo R Squared (Model 1: only Race) = 0.0
McFadden Psuedo R Squared (Model 2: only Glucose) = 0.19
McFadden Psuedo R Squared (Final Model) = 0.27
```

```
# Calculate Nagelkerke R-square

models=[glm,glm1,glm3]
model_names= {glm: 'Model 1: only Race',glm1:'Model 2: only Glucose',gl
m3: 'Final Model'}
for i in models:
    ll_Intercept=i.llnull
    ll_Model = i.llf
    N= diabetes.shape[0]
    num=(1- np.exp((ll_Model - ll_Intercept)*(-2/N)))
    den=( 1- np.exp((ll_Intercept)*(2/N)))
    nagelkerke_r2 = num/den
    print('Nagelkerke  R Squared for {} ='.format(model_names[i]),round
(nagelkerke_r2,2))
```

```
Nagelkerke  R Squared for Model 1: only Race = 0.0
Nagelkerke  R Squared for Model 2: only Glucose = 0.3
Nagelkerke  R Squared for Final Model = 0.41
```

Since the range of the pseudo-R^2 is 0 to a number less than or equal to 1, the interpretation of the above values is not easy. Instead of taking them as an absolute number, it is better to look at their relative values among the models under consideration. Thus, it is clear that the model proposed as the Final Model has considerable higher R^2 values for both types.

3.7 Accuracy of Logistic Regression as a Classification Algorithm

One major difference between multiple linear regression and logistic regression is that, in the former the value of the response is predicted directly whereas in the latter only the probability of the response being a success is predicted. To actually assign a binary value to the response, a threshold needs to be devised to partition the response space into success and failure.

Typically, the threshold is set at 50% level. If probability of success is 50% or above for a given combination of predictors, the value of response is taken to be 1, otherwise 0. However, this threshold may be set at some other convenient level.

Three measures of accuracy may be defined. Let P be the total number of successes (positives) in the data and N be the total number of failures (negatives). If a success is predicted as success, it is an example of True Positive (TP). If on the other hand a failure is predicted as failure, it is an example of True Negative (TN). In both cases, classification is correct. However, if a success is predicted as a failure, or if a failure is predicted as a success, they are misclassified.

Table 2 - Example of a confusion (misclassification) matrix

Confusion Matrix		Actual	
		Success (Positive)	Failure (Negative)
Predicted	Success (Positive)	TP	FP
	Failure (Negative)	FN	TN
Total		P	N

Probability of misclassification = $\frac{FN+FP}{n}$, where *n* is the sample size.

For the perfect logistic regression, misclassification probability is 0; i.e. no observation would have been misclassified. This indicates overfit of the model and not to be recommended, since such a model will not have good predicting power.

A few other quantities are equally important.

Precision = $\frac{TP}{TP+FP} = \frac{TP}{P}$, i.e. among all the successes (positives) in the data, how many are identified as positive by the logistic regression.

Specificity = $\frac{TN}{TP+FP} = \frac{TN}{N}$, i.e. among all failures (negatives) in the data, how many are actually identified as negative by the logistic regression

Sensitivity or Recall = $\frac{TP}{TP+FN}$, i.e. among all the predicted successes, how many are actually success.

The F-score of the model is defined as $\frac{2(Precision*Recall)}{Precision+Recall}$. F is between 0 and 1, and the closer is it to 1, the better is the model

Among two competing logistic regressions, the one that maximizes all the accuracy measures, is the one of choice. However, it may not be possible to maximize all criteria simultaneously.

Case Study continued:

```
predicted_diabetes=pd.DataFrame(glm3.predict(diabetes.drop('Outcome',axis=1)),columns=['predicted_prob'])

def zero_one(x):
    threshold =0.50
    if x>threshold:
        return 1
    else: return 0

predicted_diabetes['Label']=predicted_diabetes.predicted_prob.apply(zer
o_one)
predicted_diabetes['Outcome'] = diabetes.Outcome
predicted_diabetes.groupby(['Outcome','Label']).count().unstack()
```

```
        predicted_prob

Label   0       1

Outcome

0       423     55

1       108     143
```

If the cut-off threshold is set at 0.5, then misclassification probability is (55 + 108)/729 = 22.3%

Therefore accuracy of the model is 1 – 0.2235 = 77.6%

Precision = 143 / (143 + 108) = 57%

Specificity = 423 / (423 + 55) = 88.5%

Recall = 143 / (143 + 55) = 72.2%

F-score = 2*0.57*0.722/(0.57+0.722) = 0.64 = 64%

It is clear from the above statistics that precision of the model is not high. Among all the diabetics in the data, the model is able to correctly predict only 57% of the cases. On the other hand, specificity 88.5% indicates that, among the non-diabetics, the model is able to correctly identify 88.5%. From a medical perspective, it is always more serious missing a case than incorrectly identify subjects without the condition.

Recall that once the probability of success is estimated through the logistic regression, the partition into two groups, success and failure, is controlled by placing the cut-off threshold. Currently it is set at 0.5. Suppose to improve precision, it is decided to set at 0.35.

```
predicted_diabetes=pd.DataFrame(glm3.predict(diabetes.drop('Outcome',ax
is=1)),columns=['predicted_prob'])

def zero_one(x):
    threshold =0.35
    if x>threshold:
        return 1
    else: return 0

predicted_diabetes['Label']=predicted_diabetes.predicted_prob.apply(zer
o_one)
predicted_diabetes['Outcome'] = diabetes.Outcome
```

```
predicted_diabetes.groupby(['Outcome','Label']).count().unstack()
```

```
        predicted_prob

Label   0       1

Outcome

0       376     102

1       72      179
```

Accuracy = 76%

Precision = 71%

Specificity = 78.7%

Recall = 63.7%

F-score = 67%

Note that, by changing the threshold value, as precision improves significantly, both specificity and recall values decreases by large amount. Accuracy and F-score changes marginally, the former increases and the latter decreases.

Since these measures are a function of the threshold, often the impact of the whole range of thresholds is investigated through the Receiver Operating Characteristic (ROC) curve. The curve is typically obtained by plotting 1 – specificity (False Positive Rate, FPR) on the x-axis and sensitivity (True Positive Rate, TPR) on the y-axis. However, there may be alternative representations of the same.

```
from sklearn import metrics
# # calculate roc curve
# from sklearn.metrics import roc_curve
fpr, tpr, thresholds = metrics.roc_curve(diabetes.Outcome,glm3.predict(
diabetes.drop('Outcome',axis=1)))
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
print('Area Under the Curve', round(metrics.auc(fpr,tpr),4))
plt.plot(fpr, tpr, marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the plot
plt.show()
```

```
Area Under the Curve 0.8401
```

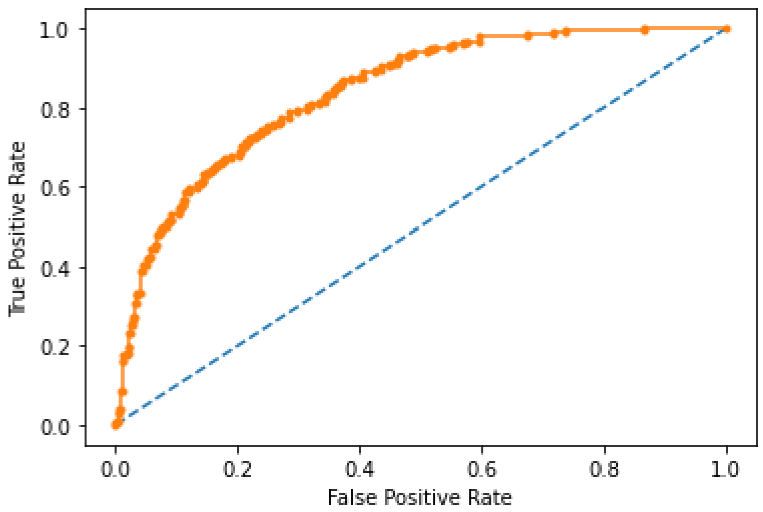


Figure 8- ROC for Diabetes data

Logistic regression may also be treated as a binary classification problem, where the model assigns a class membership to each observation.

Area under ROC curve is a performance measurement for any classification problem (not necessarily for logistic regression only) at various thresholds points. ROC is a probability curve and AUC represents the classification model’s ability to separate the two classes. The higher the AUC, the more powerful is the model to predict true class membership.

The most important observations from Fig. 8 are:

1. There is a trade-off between sensitivity and specificity; if one increases, then the other decreases.
2. The closer the curve comes to the top left corner of the probability space, the more area it covers; hence the better is the model
3. Any curve below the diagonal line is worse than a random allocation mechanism

AUC-ROC is particularly helpful in comparing two or more competing models. The model with higher AUC-ROC is expected to have better discretionary power.

Two alternate representations of Fig 8 are shown below:

```
#precision recall curve
prec,recal,_=metrics.precision_recall_curve(diabetes.Outcome,glm3.predi
ct(diabetes.drop('Outcome',axis=1)))

plt.plot(recal, prec)
plt.xlabel('Recall')
plt.ylabel('Precision')
plt.show()
```

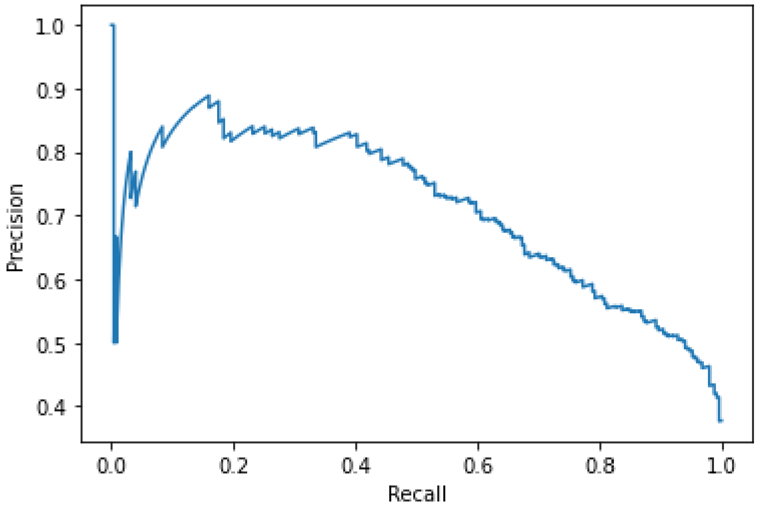


Figure 9- Precision-recall curve for Diabetes data

```
#Sensitivity vs Specificity Curve
Sensitivity = tpr
Specificity = 1 - fpr
plt.plot(Specificity,Sensitivity)
plt.xlabel('Specificity')
plt.ylabel('Sensitivity')
plt.show()
```

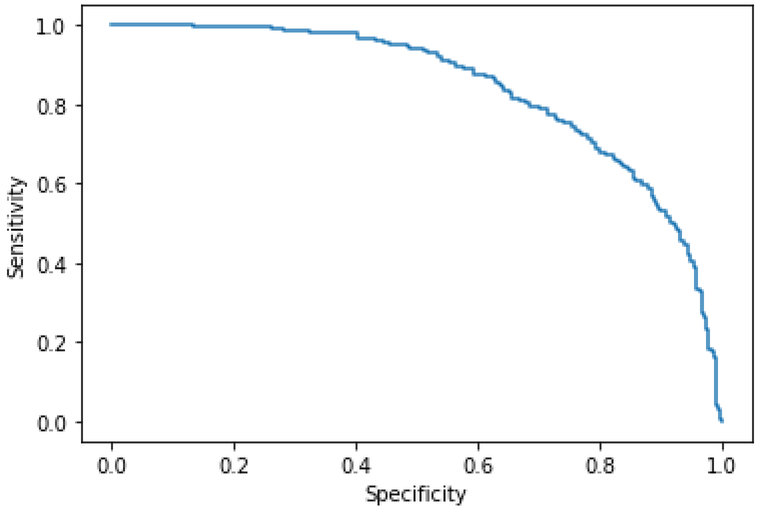


Figure 10- Sensitivity-specificity curve for Diabetes data

3.8 Enhancement of Logistic Regression Predictors: WoE and IV

To improve the performance of a regression model, often one or more predictors are transformed. For linear regression, square root or log transformations are common choice. In case of logistic regression, discretization of continuous variables is possible through a special method called Weight of Evidence (WoE). There are several advantages of computing WoE. If a predictor is non-linear, discretization will help adjusting for that. In fact, WoE is calculated in such a way, that it achieves linear relation with the log odds. Otherwise, with non-linear transformations, such as square root or power transformations, it is not easy to achieve linearity. This technique also eliminates the necessity of handling outliers or missing data separately. WoE provides a guideline for combining levels of discrete predictors too: If consecutive levels of a discrete predictor show very similar values of WoE statistics, then these levels may be combined to avoid data sparseness.

Steps for computation of WoE involves

i) Binning a continuous predictor into *k* bins. The number *k* depends on how smooth or rough the discretized predictor needs to be. The higher the value of *k*, the smoother is the discretized variable. Typically, *k* is between 10 and 20.
ii) Each bin should have at least 5% of the observations
iii) Each bin should have non-zero occurrences of both success and failure
iv) Missing values must be binned separately

The WoE must be distinct in each bin and be monotonically increasing or decreasing. If two consecutive bins have the same value, they must be combined.

Once WoE are calculated, then it replaces the original variable in the logistic regression. If WoE is computed for discrete variable, its use in the regression eliminates introduction of a set of dummy variables corresponding to the original variable.

For each category j, $j = 1, \ldots, k$ $\qquad$ $\text{WoE}(j) = log\,\frac{\Pr(X = x_j \,|Y=1)}{\Pr(X = x_j \,|Y=0)}$

This is a comparison of the distribution of the predictor X, given Y = 1 and Y = 0, separately.

Following is a case study with a much larger data set and categorical predictors where the application of WoE and IV enhancement is demonstrated.

Case Study 2

Income is a notoriously difficult item to have individuals respond to with full accuracy. Often, instead of asking about income directly, it is estimated from various other related questions. The data set Income contains 45223 observations coming from US census where the binary outcome is the annual income of the respondent. Success is defined when the income is above $50,000. The predictors are

Age: Chronological age (continuous)
Workclass: Place of employment
Fnlwgt: A derived variable used to represent sampling weightage in US census (continuous)
Education: Level of education
Education-num: Years of education completed (continuous)
Marital-status
Occupation:
Relationship: Family composition
Race:
Gender
Capital-gain: continuous
Capital-loss: continuous
Hours-per-week: Hours worked per week on an average (continuous)
Native-country: Migrated from
Income: Binary outcome variable (<= 50K : 0; > 50K : 1)

Clearly several of the variables are categorical with possibly many levels. There are no missing data.

Statement of the Problem: Develop a logistic regression model to predict whether a person earns more than $ 50,000 given demographic and other relevant socio-economic and work related information

Exploratory Analysis (EDA) on Income Data

The Income data is uploaded in Python

```
# Upload data into R using CSV file
salary=pd.read_csv('Salary.csv')
salary.shape
(45222, 15)

print(salary.head())
  Age  Workclass  Fnlwgt      Education    Educational-num    Marital-status
0   25    Private  226802           11th                  7       Never-married
1   38    Private   89814        HS-grad                  9  Married-civ-spouse
2   28  Local-gov  336951     Assoc-acdm                 12  Married-civ-spouse
3   44    Private  160323   Some-college                 10  Married-civ-spouse
4   34    Private  198693           10th                  6       Never-married

  Occupation          Relationship  Race  Gender  Capital-gain  Capital-loss
0  Machine-op-inspct Own-child     Black   Male              0             0
1  Farming-fishing   Husband       White   Male            0             0
2  Protective-serv   Husband       White   Male            0             0
3  Machine-op-inspct Husband       Black   Male         7688             0
4  Other-service     Not-in-family White   Male            0             0

   Hours-per-week Native-country Income
0              40  United-States  <=50K
1              50  United-States  <=50K
2              40  United-States   >50K
```

```
3              40  United-States   >50K
4              30  United-States  <=50K
```

```
columns= salary.select_dtypes(include='object').columns
for i in columns:
    print('\n')
    print ("---- {} ----".format(i))
    print(pd.DataFrame([salary[i].value_counts(),
                        round(salary[i].value_counts()/salary.shape[0]*
100,2)],
                       index=['Count','Percent']).T)
    print('\n')
```

```
---- Workclass ----
                    Count  Percent
Private           33307.0    73.65
Self-emp-not-inc   3796.0     8.39
Local-gov          3100.0     6.86
State-gov          1946.0     4.30
Self-emp-inc       1646.0     3.64
Federal-gov        1406.0     3.11
Without-pay          21.0     0.05


---- Education ----
                Count  Percent
HS-grad       14783.0    32.69
Some-college   9899.0    21.89
Bachelors      7570.0    16.74
Masters        2514.0     5.56
Assoc-voc      1959.0     4.33
11th           1619.0     3.58
Assoc-acdm     1507.0     3.33
10th           1223.0     2.70
7th-8th         823.0     1.82
Prof-school     785.0     1.74
9th             676.0     1.49
12th            577.0     1.28
Doctorate       544.0     1.20
5th-6th         449.0     0.99
1st-4th         222.0     0.49
Preschool        72.0     0.16


---- Marital-status ----
                         Count  Percent
Married-civ-spouse     21055.0    46.56
Never-married          14598.0    32.28
Divorced                6297.0    13.92
Separated               1411.0     3.12
Widowed                 1277.0     2.82
Married-spouse-absent    552.0     1.22
Married-AF-spouse         32.0     0.07


---- Occupation ----
                    Count  Percent
Craft-repair       6020.0    13.31
Prof-specialty     6008.0    13.29
```

```
Exec-managerial    5984.0    13.23
Adm-clerical       5540.0    12.25
Sales              5408.0    11.96
Other-service      4808.0    10.63
Machine-op-inspct  2970.0     6.57
Transport-moving   2316.0     5.12
Handlers-cleaners  2046.0     4.52
Farming-fishing    1480.0     3.27
Tech-support       1420.0     3.14
Protective-serv     976.0     2.16
Priv-house-serv     232.0     0.51
Armed-Forces         14.0     0.03


---- Relationship ----
                 Count  Percent
Husband        18666.0    41.28
Not-in-family  11702.0    25.88
Own-child       6626.0    14.65
Unmarried       4788.0    10.59
Wife            2091.0     4.62
Other-relative  1349.0     2.98


---- Race ----
                     Count  Percent
White              38903.0    86.03
Black               4228.0     9.35
Asian-Pac-Islander  1303.0     2.88
Amer-Indian-Eskimo   435.0     0.96
Other                353.0     0.78


---- Gender ----
         Count  Percent
Male   30527.0     67.5
Female 14695.0     32.5


*---- Native-country ----
                             Count  Percent
United-States              41292.0    91.31
Mexico                       903.0     2.00
Philippines                  283.0     0.63
Germany                      193.0     0.43
Puerto-Rico                  175.0     0.39
Canada                       163.0     0.36
El-Salvador                  147.0     0.33
India                        147.0     0.33
Cuba                         133.0     0.29
England                      119.0     0.26
```

**Only a part of the table is included here as there are too many countries with very few observations, except where native country is USA*

The objective of this preliminary analysis with the categorical predictors is to make a decision regarding which levels need to be combined for model building.

Income level is distributed between two classes in the ratio 1:3. The data is not balanced, i.e. class proportions are not equal, but no adjustment is required in the response distribution. There is only one binary predictor, Gender, which contains 68% male. This predictor remains as is.

Let us now consider the other categorical predictors at multiple levels.

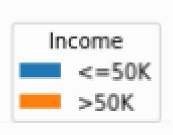
The predictors Workclass and Occupation are similar in nature. Between these two, Occupation is retained. The reason being, in Workclass the proportion is concentrated into only one class, Private. In Occupation, meaningful collapsing of class-levels is possible. The predictors Race and Native Country are reduced to binary; White and Other in the former, USA and Other in the latter.

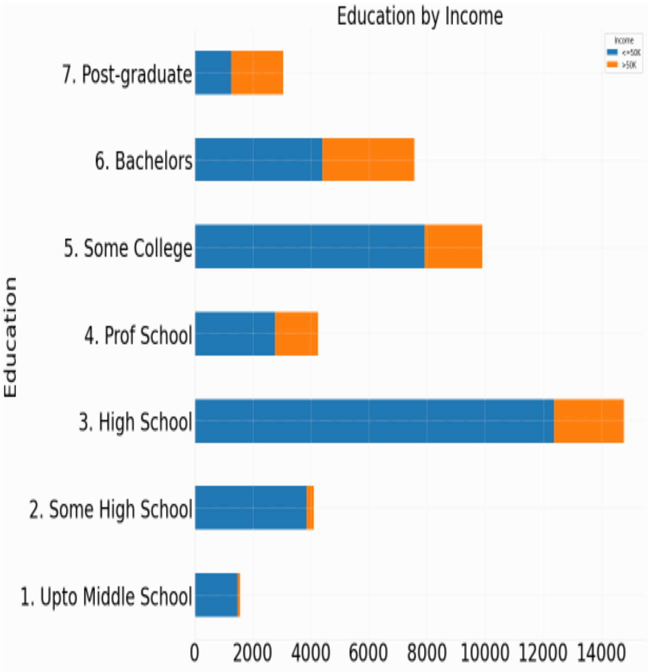
[*Another note on smart handling of hierarchical categorical variables: Levels of categorical variables are arranged in lexicographic order, i.e. A – Z, not necessarily in the order that makes sense. For example, if the data is in months and a bar diagram is to be constructed according to month, the data will be arranged in the order April, August, February, etc., which is not recommended to understand how any variation occur within a year. Similarly, note above that Education categories are arranged not in the order Pre-school to Doctorate, which would help in imparting the information is a more organized manner.*

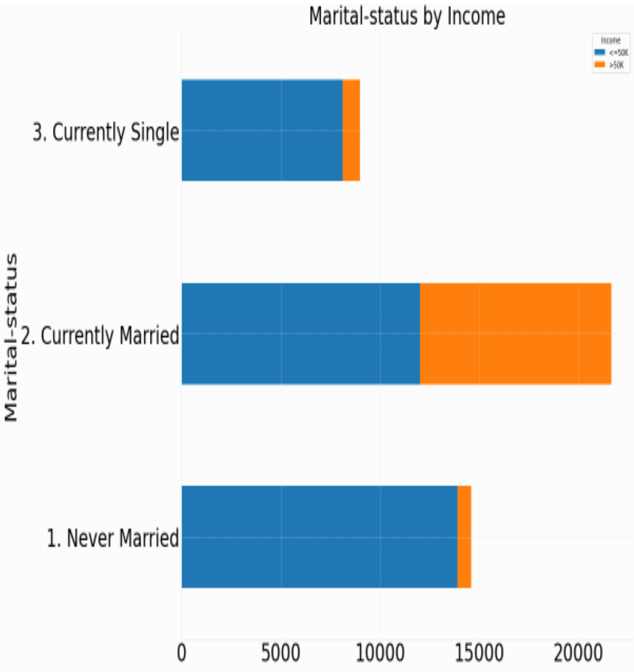
It is possible to give explicit commends to obtain categories in the most suitable order.

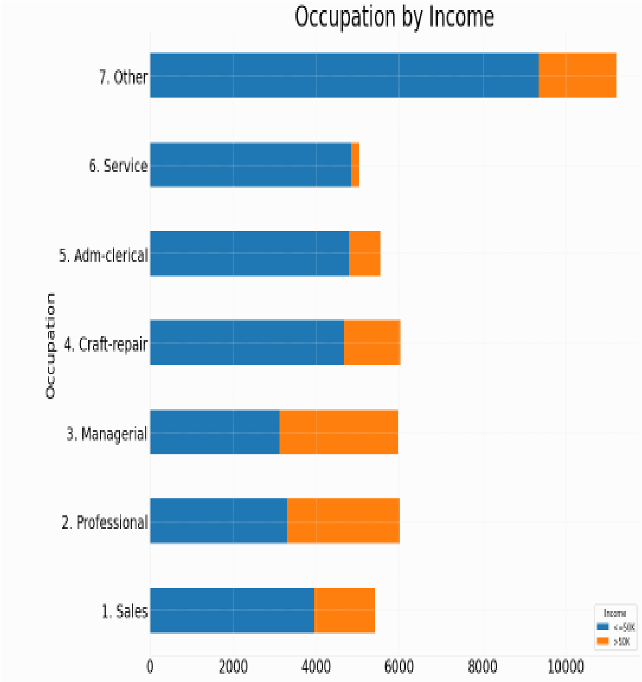
Another trick is to include a numeric in front of the categories in the order we want the categories to be arranged. Note below that when Preschool, 1st-4th, 5th-6th and 7th – 8th are combined to create a category Upto Middle School, it is called '1. Upto Middle School'; the next one is '2. Some High School' and so on. Introduction of the numeric in the name circumvents the problem with lexicographic ordering.]

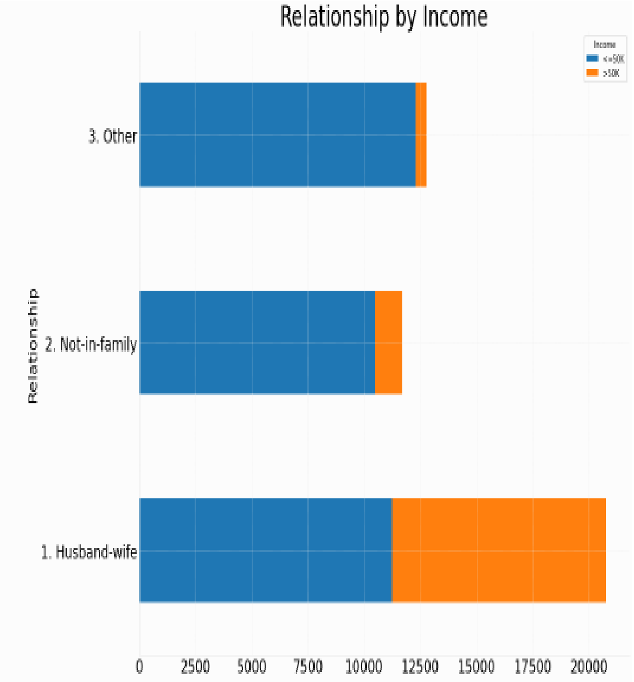
```
# Collapse levels of categorical predictors

dct_edu={'Preschool':'1. Upto Middle School', '1st-4th':'1. Upto Middle
School',
'5th-6th':'1. Upto Middle School', '7th-8th':'1. Upto Middle School',
'9th' : '2. Some High School', '10th' : '2. Some High School',
'11th' : '2. Some High School', '12th' : '2. Some High School',
'HS-grad' : '3. High School',  'Prof-school' : '4. Prof School',
'Assoc-acdm' : '4. Prof School', 'Assoc-voc' : '4. Prof School',
'Some-college' :'5. Some College', 'Bachelors' : '6. Bachelors',
'Masters' : '7. Post-graduate', 'Doctorate' : '7. Post-graduate'}


dct_marr={'Widowed':'3. Currently Single', 'Separated':'3. Currently Si
ngle',
                    'Divorced':'3. Currently Single', 'Never-married':'
1. Never Married',
                    'Married-civ-spouse':'2. Currently Married',
                    'Married-spouse-absent':'2. Currently Married',
                     'Married-AF-spouse' : '2. Currently Married'}


dct_rela={'Husband' :'1. Husband-wife', 'Wife' :'1. Husband-wife',
                                 'Not-in-family' : '2. Not-in-family',
'Unmarried' : '3. Other',
                                   'Own-child' : '3. Other', 'Other-relative' :'3. Other'}



dct_occu={'Sales' :'1. Sales', 'Prof-specialty' : '2. Professional',
'Exec-managerial' : '3. Managerial',
'Craft-repair' : '4. Craft-repair', 'Adm-clerical' : '5. Adm-clerical',
'Priv-house-serv' : '6. Service', 'Other-service' : '6. Service',
'Transport-moving' : '7. Other', 'Tech-support' : '7. Other',
'Protective-serv' : '7. Other', 'Machine-op-inspct' : '7. Other',
'Handlers-cleaners' : '7. Other', 'Farming-fishing' :'7. Other',
'Armed-Forces' : '7. Other'}


salary.Race = salary.Race.apply(lambda x : x if x == 'White' else 'Others')

salary['Native-country'] = salary['Native-country'].apply(lambda x : x if x == 'United-States' else 'Others')

salary.Education=salary.Education.apply(lambda x: dct_edu[x])

salary['Marital-status']=salary['Marital-status'].apply(lambda x: dct_marr[x])

salary.Relationship=salary.Relationship.apply(lambda x: dct_rela[x])


salary.Occupation=salary.Occupation.apply(lambda x: dct_occu[x])



fig=plt.figure(figsize=(20,20))
cols= salary.select_dtypes(include='object').drop(['Income','Workclass','Native-country'],axis=1).columns
for i in range(0, len(cols)):
    ax=fig.add_subplot(3,2,i+1)
    salary.groupby(cols[i])['Income'].value_counts().sort_values().unstack().plot.barh(stacked=True,ax=ax,fontsize=15)
    plt.title('{} by Income'.format(cols[i]),fontsize=20)
    ax.yaxis.label.set_size(20)
plt.tight_layout()
plt.show()
```




Education by Income


Marital-status by Income


Occupation by Income


Relationship by Income

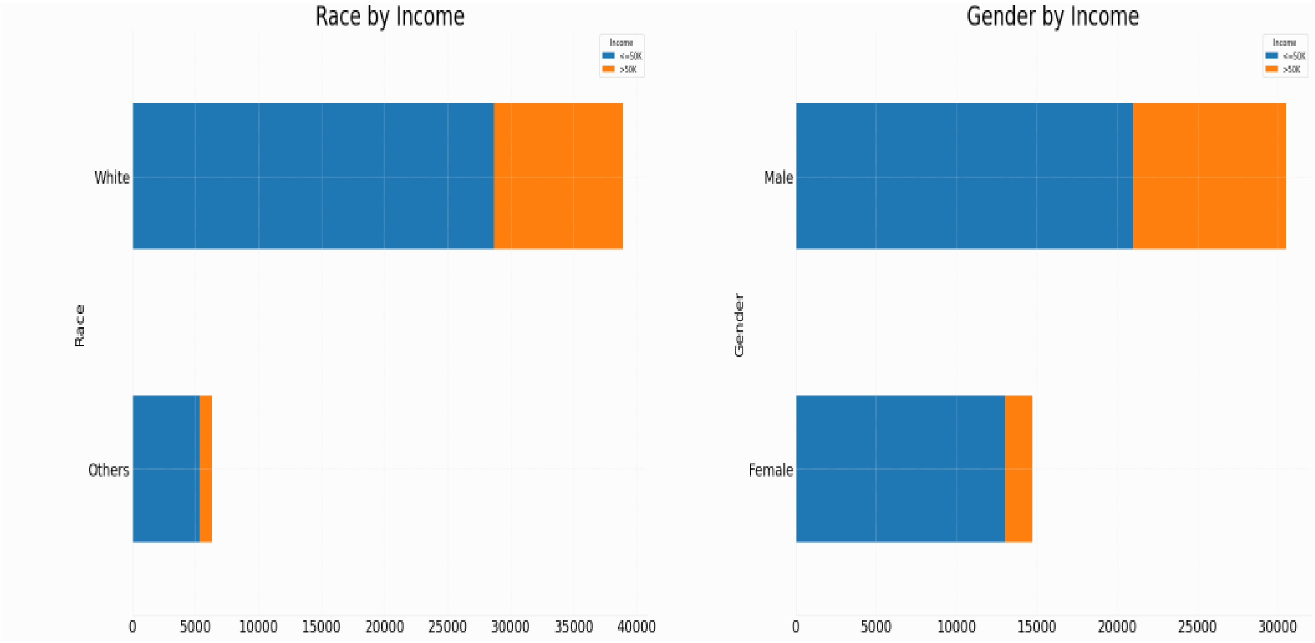


Figure 11- Income distribution across categorical predictors

Next step is to determine whether levels of the categorical predictors need to be combined further for a more informative modelling.

```
columns= salary.select_dtypes(include='object').columns

for i in columns:
    sal = salary.groupby([i,'Income']).count()['Educational-num'].unsta
ck()
    sal['Bad%']=round(sal['<=50K']/(sum(sal['<=50K']))*100,5)
    sal['Good%']=round(sal['>50K']/(sum(sal['>50K']))*100,5)
    sal['WOE'] = np.log(sal['Good%']/sal['Bad%'])
    sal['IV'] = (sal['Good%'] -sal['Bad%'])*sal['WOE']/100
    iv = sal['IV'].sum()
    print('\n')
    print('------- {} -------'.format(i))
    print(sal)
    print('\n')
    if iv>=0.1:
        print('IV for {} is {} and is Highly Predictive'.format(i,round
(iv,4)))
    elif iv<0.03:
        print('IV for {} is {} and is Not Predictive'.format(i,round(iv
,4)))
    elif iv<0.1:
        print('IV for {} is {} and is Somewhat Predictive'.format(i,rou
nd(iv,4)))
```

```
------- Education -------
Income                    <=50K   >50K      Bad%     Good%        WOE          IV
Education
1. Upto Middle School      1480     86   4.35115   0.76731 -1.735305    0.062191
2. Some High School        3843    252  11.29829   2.24839 -1.614437    0.146105
3. High School            12367   2416  36.35856  21.55603 -0.522774    0.077384
4. Prof School             2757   1494   8.10549  13.32976  0.497458    0.025989
5. Some College            7909   1990  23.25219  17.75517 -0.269723    0.014827
6. Bachelors               4392   3178  12.91233  28.35475  0.786612    0.121472
```

```
7. Post-graduate        1266  1792   3.72200  15.98858  1.457614  0.178799


IV for Education is 0.6268 and is Highly Predictive

------- Marital-status -------
Income                 <=50K  >50K      Bad%      Good%        WOE        IV
Marital-status
1. Never Married       13897   701  40.85671   6.25446 -1.876776  0.649407
2. Currently Married   12007  9632  35.30017  85.93862  0.889746  0.450553
3. Currently Single     8110   875  23.84312   7.80692 -1.116485  0.179042


IV for Marital-status is 1.279 and is Highly Predictive

------- Occupation -------
Income            <=50K  >50K      Bad%      Good%        WOE        IV
Occupation
1. Sales           3953  1455  11.62169  12.98180  0.110675  0.001505
2. Professional    3304  2704   9.71365  24.12562  0.909742  0.131112
3. Managerial      3117  2867   9.16387  25.57994  1.026540  0.168518
4. Craft-repair    4665  1355  13.71494  12.08958 -0.126142  0.002050
5. Adm-clerical    4784   756  14.06480   6.74518 -0.734847  0.053788
6. Service         4841   199  14.23237   1.77552 -2.081426  0.259280
7. Other           9350  1872  27.48868  16.70236 -0.498224  0.053740


IV for Occupation is 0.67 and is Highly Predictive


------- Relationship -------
Income             <=50K  >50K      Bad%      Good%        WOE        IV
Relationship
1. Husband-wife    11234  9523  33.02758  84.96610  0.944909  0.490772
2. Not-in-family   10474  1228  30.79320  10.95646 -1.033365  0.204986
3. Other           12306   457  36.17922   4.07744 -2.183016  0.700787


IV for Relationship is 1.3965 and is Highly Predictive


------- Race -------
Income  <=50K   >50K      Bad%      Good%        WOE        IV
Race
Others   5318   1001  15.63474   8.93112 -0.559954  0.037537
White   28696  10207  84.36526  91.06888  0.076460  0.005126


IV for Race is 0.0427 and is Somewhat Predictive


------- Gender -------
Income  <=50K  >50K    Bad%     Good%        WOE        IV
Gender
Female  13026  1669  38.296  14.89115 -0.944578  0.221077
Male    20988  9539  61.704  85.10885  0.321582  0.075266


IV for Gender is 0.2963 and is Highly Predictive


------- Native-country -------
Income          <=50K  >50K      Bad%      Good%        WOE        IV
Native-country
Others           3170   760   9.31969   6.78087 -0.318024  0.008074
United-States   30844 10448  90.68031  93.21913  0.027613  0.000701
```

```
IV for Native-country is 0.0088 and is Not Predictive
```

The predictors Ethnicity (Race) and Origin (Native Country) will not be included in the model. These predictors are too imbalanced to be of any practical use.

Note that for both cases the 2x2 contingency tables show significant χ2 values.

```
import scipy.stats as stats

ethnicity=salary.groupby(['Income','Race'])['Educational-num'].count().
unstack()

print (ethnicity)
Chi_Stat,p_value,degF,array=stats.chi2_contingency(ethnicity)
print('For Ethnicity (RACE), the Chi Square Test Statistic is {} with a
p-value of {}'.format(round(Chi_Stat,2),p_value))


origin=salary.groupby(['Income','Native-country'])['Educational-num'].c
ount().unstack()

print (origin)

Chi_Stat1,p_value1,degF1,array1=stats.chi2_contingency(origin)
print('For Origin (Native-Country), the Chi Square Test Statistic is {}
with a p-value of {}'.format(round
(Chi_Stat1,2),p_value1))
```

```
Race     Others  White
Income
<=50K      5318  28696
>50K       1001  10207

For Ethnicity (RACE), the Chi Square Test Statistic is 314.6 with a p-value
of 2.177408309728495e-70

Native-country  Others  United-States
Income
<=50K             3170          30844
>50K               760          10448

For Origin (Native-Country), the Chi Square Test Statistic is 68.16 with a
p-value of 1.5099071570211737e-16
```

This is an important observation. If the sample size is very large, almost always statistical significance is ensured. But that must not be the sole reason to include a predictor in the model.

Next step is to determine whether any of the other categorical variables need to be collapsed over their levels, so that there is enough data in each category and each category is well differentiated from the rest. Colour-coding of categories show which levels are to be combined for building the logistic model.

There are five continuous variables in the data set. Below are the histograms for these.

```
# Histogram of a few select variables
```

```
cont=salary.drop('Educational-num',axis=1).select_dtypes(exclude='objec
t').columns
fig = plt.figure(figsize = (8,8))
ax = fig.gca()
salary[cont].hist(ax = ax)
plt.tight_layout()
plt.show()
```

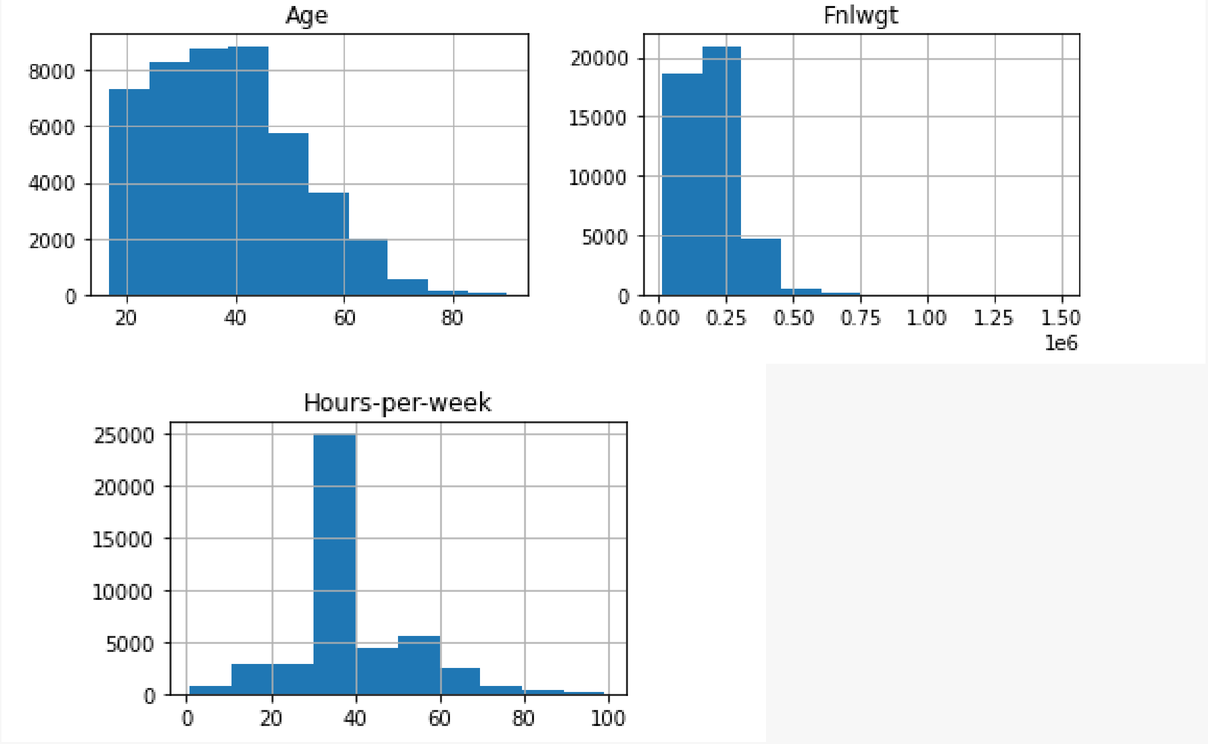


Figure 12- Histogram of continuous predictors: Set 1

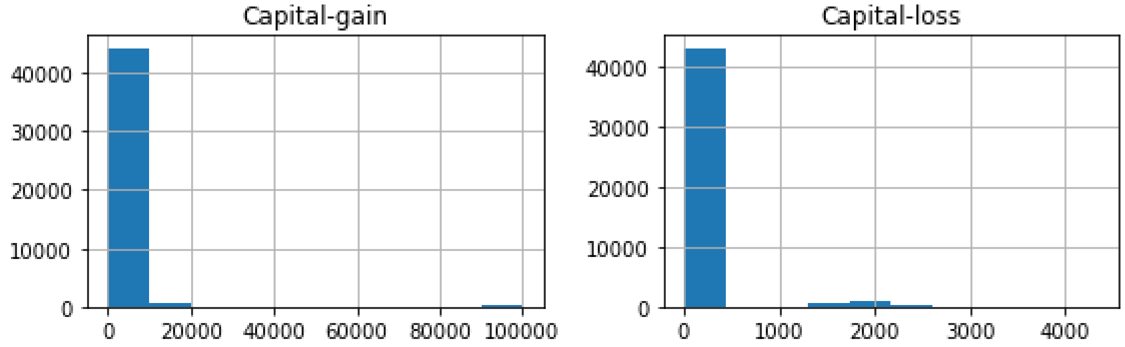


Figure 13- Histogram of continuous predictors: Set 2

Clearly the extreme skewness of both Capital Gain and Capital Loss variables demand adjustment. Since most of the observations are concentrated on 0 value for both variables, one logical transformation is discretization of the continuous variables.

```
salary['Capital-gain'] = salary['Capital-gain'].apply(lambda x : x if x
== 0 else 1)
salary['Capital-loss'] = salary['Capital-loss'].apply(lambda x : x if x
== 0 else 1)
salary['Capital-loss'].value_counts()
```

```
0     43082
1      2140
```

```
Name: Capital-loss, dtype: int64
salary['Capital-gain'].value_counts()
0    41432
1     3790
Name: Capital-gain, dtype: int64
fig=plt.figure(figsize=(12,7))
cols= salary[['Capital-loss','Capital-gain']].columns
for i in range(0, len(cols)):
    ax=fig.add_subplot(2,2,i+1)
    salary.groupby(cols[i])['Income'].value_counts().sort_values().unst
ack().plot.barh(stacked=True,ax=ax)
    plt.legend(['<=50K','>50K'])
    plt.title('{} by Income'.format(cols[i]),fontsize=20)
plt.tight_layout()
plt.show()
```

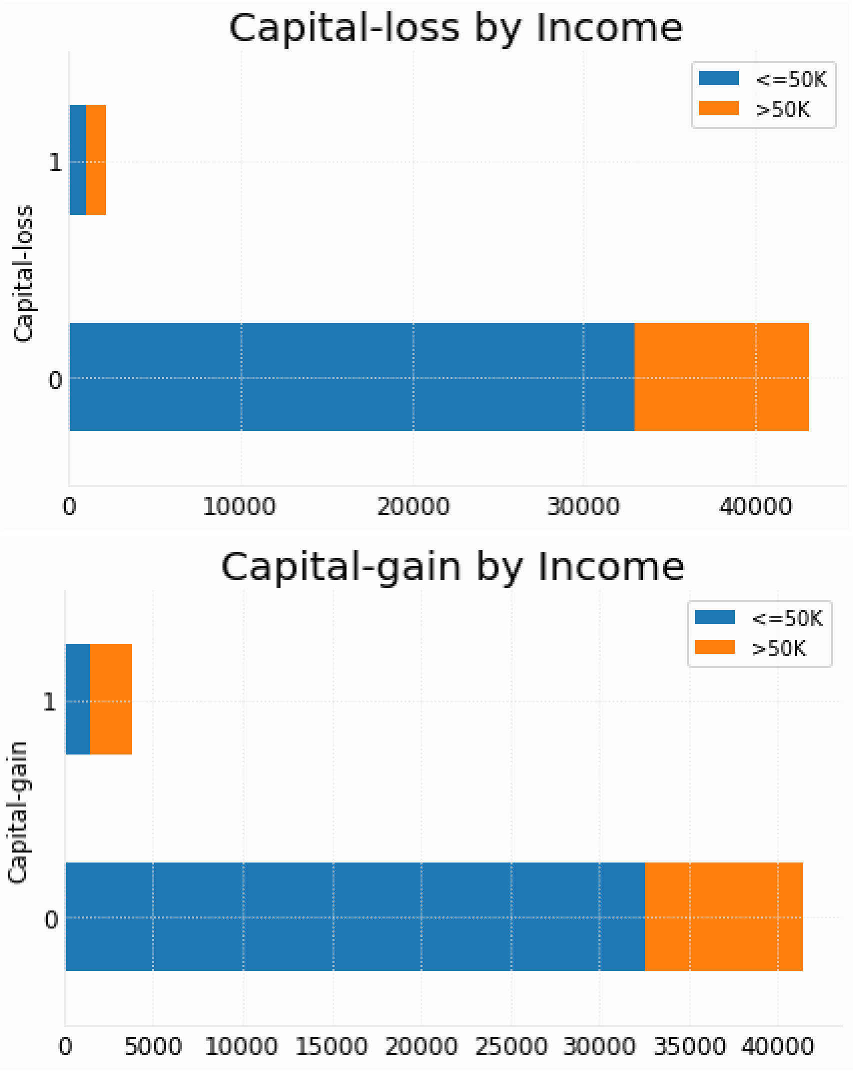


Figure 14-Distribution of Income across Capital Gain and Capital Loss

The proportion of non-zero values is so small, most likely these predictors will be non-informative. These two predictors will not be used in model building.

```
columns= salary[['Capital-loss','Capital-gain']].columns

for i in columns:
    sal = salary.groupby([i,'Income']).count()['Educational-num'].unsta
ck()
    sal['Bad%']=round(sal['<=50K']/(sum(sal['<=50K']))*100,5)
    sal['Good%']=round(sal['>50K']/(sum(sal['>50K']))*100,5)
    sal['WOE'] = np.log(sal['Good%']/sal['Bad%'])
    sal['IV'] = (sal['Good%'] -sal['Bad%'])*sal['WOE']/100
    iv = sal['IV'].sum()
    print('\n')
    print('------- {} -------'.format(i))
    print(sal)
    print('\n')
    if iv>=0.1:
        print('IV for {} is {} and is Highly Predictive'.format(i,round
(iv,4)))
    elif iv<0.03:
        print('IV for {} is {} and is Not Predictive'.format(i,round(iv
,4)))
    elif iv<0.1:
        print('IV for {} is {} and is Somewhat Predictive'.format(i,rou
nd(iv,4)))
```

```
------- Capital-loss -------
Income        <=50K   >50K      Bad%      Good%       WOE         IV
Capital-loss
0             32972  10110  96.93656  90.20343 -0.071989   0.004847
1              1042   1098   3.06344   9.79657  1.162494   0.078272


IV for Capital-loss is 0.0831 and is Somewhat Predictive


------- Capital-gain -------
Income        <=50K  >50K      Bad%      Good%       WOE         IV
Capital-gain
0             32599  8833  95.83995  78.80978 -0.195643   0.033318
1              1415  2375   4.16005  21.19022  1.628013   0.277253


IV for Capital-gain is 0.3106 and is Highly Predictive
```

Some more adjustments to a select few Object Columns before Modelling

```
newclass_edu = {'1. Upto Middle School' :'1_Not_HS_pass',
               '2. Some High School' : '1_Not_HS_pass',
                  '3. High School' :'2_HS_Coll_DO',
                  '5. Some College': '2_HS_Coll_DO',
                  '4. Prof School' : '3_Prof_School',
                  '6. Bachelors' : '4_College_Grad',
                  '7. Post-graduate' : '5_Grad_School'}

newclass_occu = { '1. Sales' : '1_Sales_Repair',
                     '4. Craft-repair' : '1_Sales_Repair',
                     '2. Professional' : '2_Prof_Manager',
                     '3. Managerial' : '2_Prof_Manager',
                     '6. Service' : '3_Service',
                     '5. Adm-clerical' : '4_Other',
                         '7. Other' : '4_Other'}

newclass_rela={'1. Husband-wife' :'1_Husband-wife','2. Not-in-family' : '2_Not_in_family', '3. Other' : '3_Other'}


newclass_marr={'3. Currently Single':'3_Currently_Single', '1. Never Married':'1_Never_Married',
                       '2. Currently Married':'2_Currently_Married'}
salary.rename(columns={'Marital-status':'Marital_Status'},inplace=True)

salary['Marital_Status_New']=salary['Marital_Status'].apply(lambda x: newclass_marr[x])

salary['Education_New']=salary['Education'].apply(lambda x: newclass_edu[x])

salary['Occupation_New']=salary['Occupation'].apply(lambda x: newclass_occu[x])

salary['Relationship_New']=salary['Relationship'].apply(lambda x: newclass_rela[x])
```

One Hot Encoding

```
columns=['Gender','Race','Education_New','Marital_Status_New', 'Relationship_New','Occupation_New']

df=pd.get_dummies(salary,columns=columns,drop_first=True)
```

3.9 Training and Validation (Test)

(*For a detail discussion on training and test data sets and the advantage of cross-validation refer to the Monograph on Regression Model Selection*)

After the EDA and all adjustments and transformations were performed on the full data, it was randomly split into training and test sets in 70:30 ratio.

```
from sklearn.model_selection import train_test_split

train,test=train_test_split(df, test_size=0.30,random_state=123,shuffle=False)
train.shape
(31655, 15)
test.shape
(13567, 15)
print('\n')
print ("---- Train ----")
print(pd.DataFrame([train.Income.value_counts(),
                        round(train.Income.value_counts()/train.shape[0
]*100,1)],
                       index=['Count','Percent']).T)
print('\n')

print('\n')
print ("---- Test ----")
print(pd.DataFrame([test.Income.value_counts(),
                        round(test.Income.value_counts()/test.shape[0]*
100,1)],
                       index=['Count','Percent']).T)
print('\n')
```

```
---- Train ----
         Count   Percent
<=50K  23850.0   75.34
>50K    7805.0   24.65

---- Test ----
         Count   Percent
<=50K  10164.0   74.91
>50K    3403.0   25.08
```

It is always a good idea to check that the success proportion of response is similar in both training and test data.

Encoding the Target Variable

```
from sklearn.preprocessing import LabelEncoder
Lr=LabelEncoder()
train.Income=Lr.fit_transform(train.Income)
test.Income=Lr.transform(test.Income)

train.rename(columns={'Hours-per-week':'Hours_per_week'},inplace=True)

test.rename(columns={'Hours-per-week':'Hours_per_week'},inplace=True)

formula = 'Income ~ Age +Fnlwgt + Gender_Male + Hours_per_week + Race_W
hite  +  Education_New_2_HS_Coll_DO +  Education_New_3_Prof_School +  E
ducation_New_4_College_Grad +  Education_New_5_Grad_School +  Marital_S
tatus_New_2_Currently_Married +  Marital_Status_New_3_Currently_Single
+  Relationship_New_2_Not_in_family +  Relationship_New_3_Other +  Occu
pation_New_2_Prof_Manager +  Occupation_New_3_Service +  Occupation_New
_4_Other'
```

Logit Model

```
model=sm.logit(formula,data=train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.371349
         Iterations 8
```

```
print(model.summary())
```

```
                           Logit Regression Results
=======================================================================
===
Dep. Variable:                 Income   No. Observations:         31655
Model:                          Logit   Df Residuals:             31638
Method:                           MLE   Df Model:                    16
Date:                Mon, 05 Oct 2020   Pseudo R-squ.:           0.3348
Time:                        14:32:39   Log-Likelihood:         -11755.
converged:                       True   LL-Null:                -17670.
```

```
Covariance Type:            nonrobust   LLR p-value:              0.000
=======================================================================
                 coef    std err     z     P>|z|      [0.025      0.975]
-----------------------------------------------------------------------
Intercept      -5.6671   0.203   -27.981 0.000      -6.064      -5.270
Age             0.0258   0.001    17.593 0.000       0.023       0.029
Fnlwgt        6.89e-07  1.57e-07   4.392  0.000    3.82e-07    9.96e-07
Gender_Male     0.0808   0.047     1.717  0.086     -0.011       0.173
Hours_per_week  0.0267   0.001    18.289 0.000       0.024       0.030
Race_White      0.1732   0.055     3.151  0.002       0.065       0.281
Edu_HS_Coll_DO  1.2648   0.075    16.809  0.000       1.117       1.412
Edu_Prof_School 1.9614   0.087    22.533  0.000       1.791       2.132
Edu_4_College   2.2950   0.082    27.936  0.000       2.134       2.456
Edu_5_Grad_Sch  2.6225   0.094    28.030  0.000       2.439       2.806
Mar_2_Married   1.2332   0.146     8.461  0.000       0.948       1.519
Mar_3_Single    0.4752   0.071     6.695  0.000       0.336       0.614
Rel_2_No_family-1.1243   0.142    -7.944  0.000      -1.402      -0.847
Rel_3_Other    -1.7723   0.144   -12.304  0.000      -2.055      -1.490
Occ_2_Prof_Mgr  0.6463   0.044    14.576  0.000       0.559       0.733
Occ_3_Service  -1.0768   0.098   -10.983  0.000      -1.269      -0.885
Occ_4_Other    -0.2476   0.042    -5.874  0.000      -0.330      -0.165
===========================================================================
```

**Some Variable names are shortened to maintain the format*

The only predictor which is not significant at 5% level is Gender. This is eliminated in the next stage.

```
formula = 'Income ~ Age +Fnlwgt + Hours_per_week + Race_White  +  Educa
tion_New_2_HS_Coll_DO +  Education_New_3_Prof_School +  Education_New_4
_College_Grad +  Education_New_5_Grad_School +  Marital_Status_New_2_Cu
rrently_Married +  Marital_Status_New_3_Currently_Single +  Relationshi
p_New_2_Not_in_family +  Relationship_New_3_Other +  Occupation_New_2_P
rof_Manager +  Occupation_New_3_Service +  Occupation_New_4_Other'
```

```
model2=sm.logit(formula,data=train).fit()
Optimization terminated successfully.
         Current function value: 0.371395
         Iterations 8
print(model2.summary())
```

```
                           Logit Regression Results
=======================================================================
Dep. Variable:                    Income   No. Observations:      31655
Model:                             Logit   Df Residuals:          31639
Method:                              MLE   Df Model:                 15
Date:                   Mon, 05 Oct 2020   Pseudo R-squ.:        0.3347
Time:                           14:33:47   Log-Likelihood:      -11757.
converged:                          True   LL-Null:             -17670.
Covariance Type:               nonrobust   LLR p-value:           0.000
=======================================================================
                      coef     std err     z    P>|z|    [0.025     0.975]
-----------------------------------------------------------------------
Intercept          -5.6177   0.200   -28.022 0.000   -6.011      -5.225
Age                 0.0259   0.001    17.699 0.000    0.023       0.029
Fnlwgt           7.003e-07 1.57e-07   4.468 0.000 3.93e-07    1.01e-06
Hours_per_week      0.0271   0.001   18.802  0.000    0.024       0.030
Race_White          0.1768   0.055    3.219  0.001    0.069       0.285
```

```
Edu_2_HS_Coll_DO       1.2619   0.075   16.771   0.000    1.114        1.409
Edu_3_Prof_School      1.9581   0.087   22.499   0.000    1.788        2.129
Edu_4_College_Grad     2.2919   0.082   27.900   0.000    2.131        2.453
Edu_5_Grad_School      2.6184   0.094   27.999   0.000    2.435        2.802
Mar_2_Married          1.2371   0.146    8.485   0.000    0.951        1.523
Mar_3_Single           0.4639   0.071    6.563   0.000    0.325        0.603
Rel_2_No_family       -1.1428   0.141   -8.093   0.000   -1.420       -0.866
Rel_3_Other           -1.8013   0.143  -12.588   0.000   -2.082       -1.521
Occ_2_Prof_Mgr         0.6370   0.044   14.475   0.000    0.551        0.723
Occ_3_Service         -1.0924   0.098  -11.191   0.000   -1.284       -0.901
Occ_4_Other           -0.2543   0.042   -6.060   0.000   -0.337       -0.172
==========================================================================
```

To print the Odds Ratio for all variables:

```
params = model2.params
conf = model2.conf_int()
conf['Coef']=np.round(model2.params.values,7)
conf['Odds Ratio'] = params
conf.columns = ['5%', '95%', 'Odds Ratio', 'Coef']
conf[['5%', '95%', 'Odds Ratio']]=np.exp(conf[['5%', '95%', 'Odds Ratio']]
)
conf
```

Age, Fnlwgt and Hours.per.week have positive coefficients, indicating that as values of these predictors increase, probability of yearly income being above $50,000 also increases. For every unit increase in the level of these three predictors, the increase in odds of earning above $50,000 is given in the table below.

Table 3- Calculation of odds ratio from regression coefficients (continuous variables)

PREDICTOR	REGRESSION COEFFICIENT	ODDS RATIO
AGE	0.026	1.03
FNLWGT	0.0000007	1.00
HOURS_PER_WEEK	0.03	1.03

The odds may not look significant but recall that these are for 1 year increase in age, or 1 hour increase in working per week (not per day). However, if difference of 10 years is compared, the odds ratios will look substantial. Similarly, if odds of working 35 hours per week is compared against working 40 hours (i.e. 1 more hour of working per day) is compared, the difference will be significant.

Regression coefficient for ethnicity is 0.18 for whites combined to others, indicating that odds are 1.20 times in favour of whites earning more than $50,000 per year.

Let us now consider the odds ratios corresponding to the multi-category predictors (colour-coded above for better identification)

Table 4- Calculation of odds ratio for different levels of Education

PREDICTOR: EDUCATION (BASELINE = NOT HIGH SCHOOL PASS)	REGRESSION COEFFICIENT	ODDS RATIO
HS/COLLEGE DROP OUT	1.26	3.53
PROFESSIONAL SCHOOL	1.95	7.08
COLLEGE GRADUATE	2.29	9.89
GRADUATE SCHOOL	2.61	13.71

While investigating effect of a categorical predictor, ordinal or nominal, each level is compared against a pre-determined baseline. For an ordinal predictor, usually the lowest or the highest level is assumed to be the baseline. For a nominal variable, any suitable level may be taken as the baseline. Education could be argued to be an ordinal variable. However, in this case, the hierarchy level between college dropout and professional school may be open to debate. To avoid that, the predictor is assumed to be nominal.

From the above table it is clear that compared to a person not finishing high school, everybody else is at a much higher level of advantage. While the odds for a person who has completed high school or have a few years of college education (though not completed) to earn over $50,000 yearly is almost 4 times higher than one who has not completed high school, the odds of a person completing graduate school that value is close to 14 times!

Similarly, for each of the other three predictors, such odds ratios can be calculated against the baseline specified. The positive sign in the regression coefficient indicates that the odds ratio will be more than 1, the negative sign indicates that the value will be less than 1. Let us consider the levels of occupation, the baseline for which is sales/repair.

Table 5- Calculation of odds ratio for different levels of Occupation

PREDICTOR: OCCUPATION (BASELINE = SALES/REPAIR)	REGRESSION COEFFICIENT	ODDS RATIO
PROFESSIONAL/MANAGER	0.63	1.89
SERVICE	-1.09	0.34
OTHER	-0.25	0.77

While persons in professional and managerial level are more likely to earn $50,000 per annum compared to those in sales and repair professions, those in service or other occupations are less likely to earn that amount. The odds ratios in the above table bear support to this claim.

Combining all the regression coefficients and transforming them appropriately on the logit scale comparison of all combinations of the predictors, vis-à-vis a suitable baseline may be done.

3.10 Training and Test Accuracy

Once a satisfactory model is developed on the training data, the same can be applied to estimate accuracy on both training and test data.

3.10.1 Prediction on Training Set

```
predicted_on_Train=pd.DataFrame(model2.predict(train.drop('Income',axis
=1)),columns=['predicted_prob'])

def zero_one(x):
    threshold =0.35
    if x>threshold:
        return 1
    else: return 0

print('Confusion Matrix  on Train Set\n')


predicted_on_Train['Label']=predicted_on_Train.predicted_prob.apply(zer
o_one)

predicted_on_Train['Income'] = train.Income

print(predicted_on_Train.groupby(['Income','Label']).count().unstack())
```

```
Confusion Matrix  on Train Set

       predicted_prob
Label               0     1
Income
0               20000  3859
1                2328  5468
```

```
print('Classification Report on Train Set\n')

print(metrics.classification_report(predicted_on_Train.Income,predicted
_on_Train.Label))
```

```
Classification Report on Train Set

              precision    recall  f1-score   support

           0       0.90      0.84      0.87     23859
           1       0.59      0.70      0.64      7796

    accuracy                           0.80     31655
   macro avg       0.74      0.77      0.75     31655
weighted avg       0.82      0.80      0.81     31655
```

```
# # calculate roc curve
# from sklearn.metrics import roc_curve
fpr, tpr, thresholds = metrics.roc_curve(train.Income,model2.predict(tr
ain.drop('Income',axis=1)))
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
print('For Training Set')
print('Area Under the Curve', round(metrics.auc(fpr,tpr),4))
plt.plot(fpr, tpr, marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the plot
plt.show()
```

```
Area Under the Curve 0.8732
```

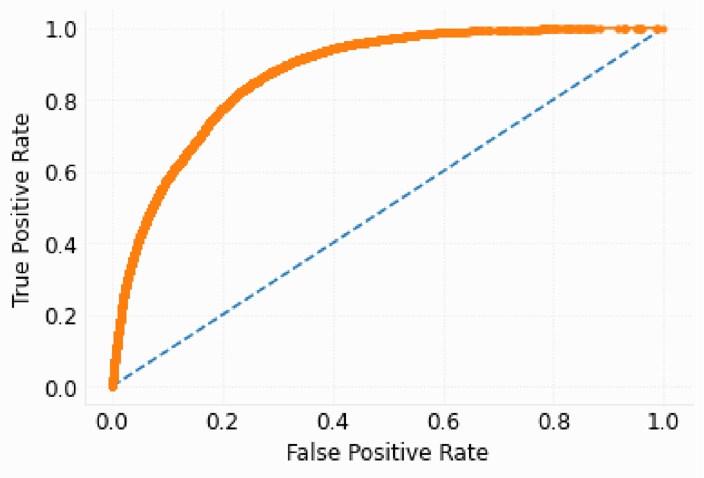


Figure 15-ROC for Training Data

A number of accuracy measures are available. These are all discussed in detail for Case Study 1.

3.10.2 Prediction on Test Set

Once a model is decided upon, the same is applied to Test data. Here the model is NOT developed independently, but the same parameter estimates found from Training data are used. The goal is pure prediction accuracy; i.e., when new observation vectors are available, how accurately the model is expected to predict the probability of having income more than 50k.

No model estimate is done for Test data.

```
predicted_on_Test=pd.DataFrame(model2.predict(test.drop('Income',axis=1
)),columns=['predicted_prob'])

def zero_one(x):
    threshold =0.35
    if x>threshold:
        return 1
    else: return 0
```

```
print('Confusion Matrix on Test Set\n')

predicted_on_Test['Label']=predicted_on_Test.predicted_prob.apply(zero_
one)

predicted_on_Test['Income'] = test.Income

print(predicted_on_Test.groupby(['Income','Label']).count().unstack())
```

```
Confusion Matrix on Test Set

       predicted_prob
Label               0     1
Income
0                8572  1583
1                 988  2424
```

```
print('Classification Report on Test Set\n')
print(metrics.classification_report(predicted_on_Test.Income,predicted_
on_Test.Label))
```

```
Classification Report on Test Set

              precision    recall  f1-score   support

           0       0.90      0.84      0.87     10155
           1       0.60      0.71      0.65      3412

    accuracy                           0.81     13567
   macro avg       0.75      0.78      0.76     13567
weighted avg       0.82      0.81      0.82     13567
```

```
# # calculate roc curve
# from sklearn.metrics import roc_curve
fpr, tpr, thresholds = metrics.roc_curve(test.Income,model2.predict(tes
t.drop('Income',axis=1)))
plt.plot([0, 1], [0, 1], linestyle='--')
# plot the roc curve for the model
print('For Test Set')
print('Area Under the Curve', round(metrics.auc(fpr,tpr),4))
plt.plot(fpr, tpr, marker='.')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the plot
plt.show()
```

```
Area Under the Curve 0.8749
```

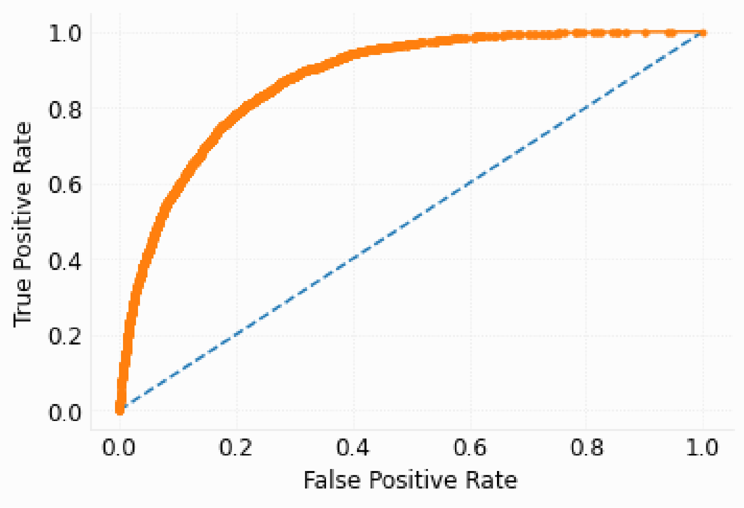


Figure 16-ROC for Test Data

Note that in this case prediction accuracy in Training and Test data are very close. That is a support for consistency of the model building procedure. Most often though training accuracy is considerably more than the test accuracy. If that happens, it is advisable to go back to the drawing board and scrutinize the predictors and their behaviour.

4. Further Discussions and Considerations

Logistic regression falls under the class of Generalized Regression Models (GLM), where a function of the response is regressed on the predictors. Logistic transformation of probability of success is the best known member of GLM, but by no means it is the only one. Because of the particular structure of logistic regression, residuals are defined and treated in a different manner.

4.1 Residuals of Logistic Model

Logistic model formulation does not require any explicit inclusion of error term. However, a residual may always be defined as the difference between the observed and the expected under the specified model. Two types of residuals associated with logistic regression are the Pearson residual and the deviance residual. Recall that in case of logistic regression modelling, deviance statistic is used to determine goodness of fit, which is based on the deviance residuals. The deviance statistic is the sum of squares of the deviance residuals and it follows a $\chi2$ distribution with $n - p$ degrees of freedom, where p is the number of parameters in the logistic regression model including the intercept. For an acceptable model, the value of the deviance statistic is expected to be equal to the residual degrees of freedom, because expected value of a $\chi2$ distribution is equal to its df.

The deviance residuals, or the standardized deviance residuals, may be plotted to investigate model adequacy.

4.2 Overdispersion

When a linear logistic model has a satisfactory fit to the binomial data, the mean deviance should be close to 1. If the fit is not satisfactory, i.e., if the fitted model is not adequate to describe the observed proportions, the mean deviance is expected to be more than 1. On the other hand, if the model is adequate, but still the residual mean deviance is greater than 1, the data is said to exhibit overdispersion or extra binomial variation. Note that for a binomial distribution, both mean and variance depends on the unknown success probability.

There may be various explanations of overdispersion. It may be due to an inadequacy in the model, e.g. non-linear terms are required instead of only linear terms. Or overdispersion may be due to insufficient number of interaction terms in the model. Overdispersion may be caused by presence of outliers in the data. It is important that all the possible causes of apparent overdispersion are eliminated before it is inferred that the data is overdispersed.

If overdispersion still persists, then along with the estimation of logistic regression parameters, an overdispersion parameter φ needs to be estimated. The estimate depends on the logistic regression parameters too. It is recommended that the model with maximum number of parameters is utilized to estimate φ by taking the ratio of the average of the residuals and 1. Once the parameter is estimated, the observations needs to be properly weighted by an appropriate function of φ and alternative logistic models may be fitted.

4.3 Goodness-of-Fit for Logistic Model

A goodness of fit statistic compares the observed and the expected values. The statistic tests the null hypothesis H_0: The model under consideration fits the data against the alternative Ha: The model does not fit the data. If the null hypothesis is rejected, the alternative does not specify which model fits the data.

The deviance statistic associated with a logistic regression model is also a goodness of fit statistic. An alternative statistic often considered is the Hosmer-Lemeshow statistic. The data is grouped into deciles and average probability of success and average probability of failure in each decile are computed. Based on the actual and expected number of success, and actual and expected number of failures in each decile, a Pearson goodness of fit statistic is computed.

4.4 Probit and Log-Log models

Consider again a binary outcome, where success probability is denoted by π. Note that the logit (or logistic) transformation is $\log\frac{\pi}{1-\pi}$. The probit transformation depends on the inverse cumulative probability of a standard normal distribution. One common application area of probit model is to estimate the effective dose of a drug trial. If the cumulative probability distribution of a standard normal distribution is denoted by Φ, the probit transformation is defined by $\Phi^{-1}(\pi)$. This is used as the response and a linear regression involving the predictors is developed.

The complementary log-log transformation is defined as $\log[-\log(1 - \pi)]$.

Both probit and complementary log – log function maps a probability in the range (0, 1) to the real line $(-\infty, \infty)$, just as logistic model does.

4.5 Response having 3 or more classes

Logistic regression deals with a binary response. However, there are extensions of logistic regression available where multi-class responses can also be handled.

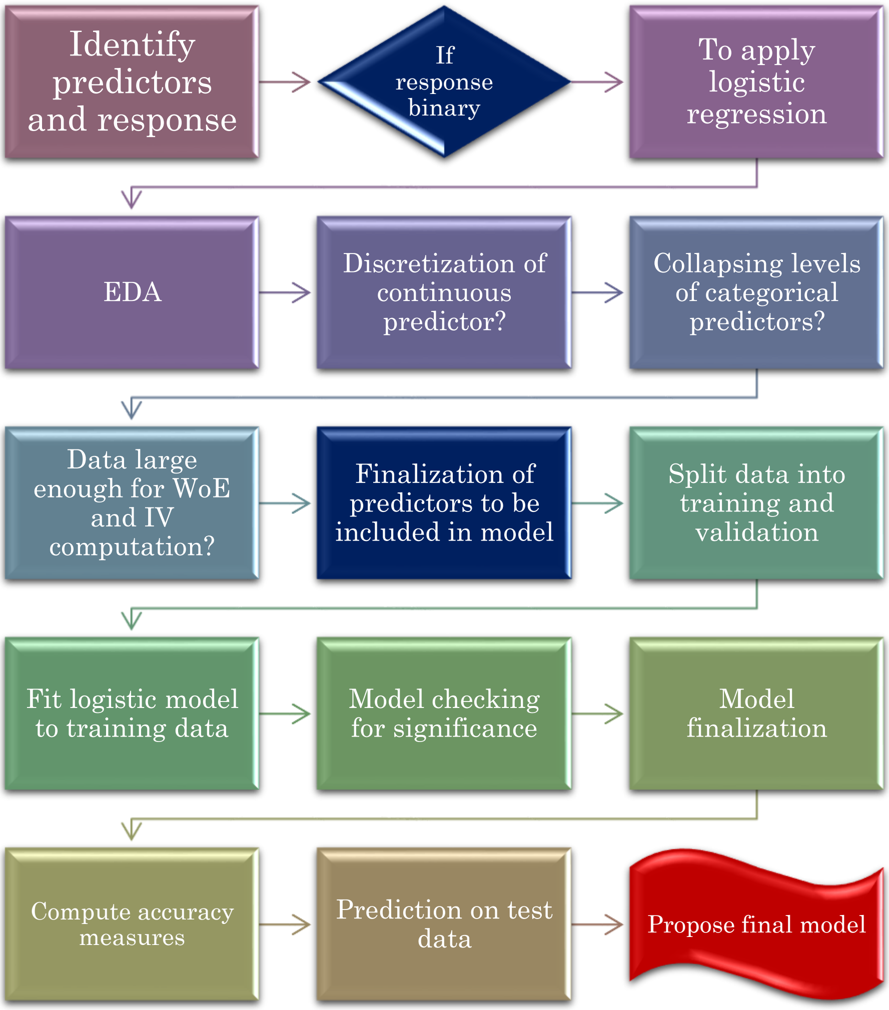
4.5.1 Nominal Logit Model

Consider a discrete response with more than two levels. Examples of such cases arise when a brand choice is considered. If there is no hierarchical relationship among the levels of the response, a nominal logit model may be applied to determine the dependency of the brand choice on the predictors. Among the levels of the brand, one is considered baseline and a series of logit functions are determined comparing the base level with the other levels. If there are K levels of the response, then a sequence of $K – 1$ logits are determined. Multinomial logistic regression is performed taking into account all the $K – 1$ logits simultaneously.

4.5.2 Ordinal Logit Model

If there is inherent hierarchy among the levels of the response, then an ordinal logit model is fitted. An example of such a case arises when the response blood pressure levels are categorized into normal, high-normal, moderate and very high. Depending on the situation either the lowest level or the highest level of the response is assumed to be the baseline. A sequence of models is developed exploiting the natural ordering.

Flowchart of Logistic Regression



Identify predictors and response

If response binary

To apply logistic regression

EDA

Discretization of continuous predictor?

Collapsing levels of categorical predictors?

Data large enough for WoE and IV computation?

Finalization of predictors to be included in model

Split data into training and validation

Fit logistic model to training data

Model checking for significance

Model finalization

Compute accuracy measures

Prediction on test data

Propose final model