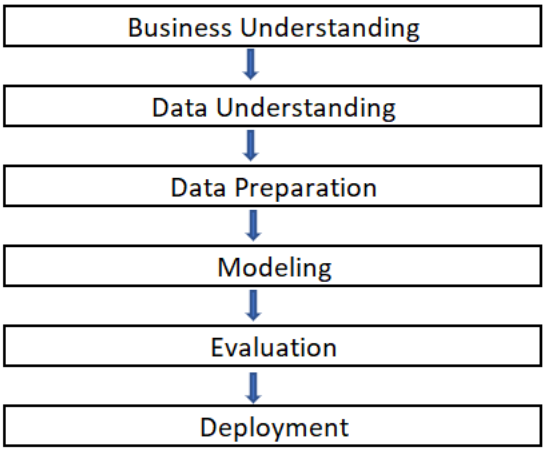
PREDICTING ACCEPTANCE OF PERSONAL LOANS: A COMPARATIVE STUDY OF DISCRIMINANT ANALYSIS AND LOGISTIC REGRESSION MODELS

Praneeth Jajjara

BUSINESS PROBLEM:

Predicting Acceptance of Personal Loans

In the process of sorting the given data set to identify the pattern and relationship that can help us solve the business problem through analysis we follow the below mentioned steps:



Business Understanding:

We would like to predict whether the person is likely to accept the banks offer for a personal loan.

Data Understanding:

We have 13 variables as mentioned below and we need to consider 12 as independent variables “X” and 1 as dependent variable “Y” which is personal loan.

	Age	Experience	Income	ZIP Code	Family	CCAvg	Education	Mortgage	Personal Loan	Securities Account	CD Account	Online	CreditCard
0	46	20	73	93106	1	1.5	2	128	0	0	0	1	0
1	43	19	75	90041	3	0.3	3	0	0	0	0	0	0
2	32	7	55	91301	4	2.0	2	0	0	0	0	1	0
3	28	2	51	94720	4	1.8	3	0	0	0	0	0	1
4	33	6	78	90250	4	2.0	2	119	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	60	35	191	93407	4	5.6	3	0	1	0	0	0	0
996	54	30	194	92056	3	6.0	3	587	1	1	1	1	1
997	30	6	64	94305	4	3.4	1	117	0	0	0	0	0
998	39	14	74	94305	3	3.0	1	0	0	0	0	0	0
999	51	27	92	92121	4	3.0	1	0	1	0	1	1	1

1000 rows × 13 columns

Data Preparation and Modelling:

Step 1:

With respect to the given condition categorise the dependent and independent variables ‘X’ and ‘Y’

```
#Now we use the Personal Loan as the Target Variable.
#Defining X and Y variables according to the condition given

x = data.drop(['Personal Loan'],axis = 1)
y = data['Personal Loan']

x.shape,y.shape
```

```
((1000, 11), (1000,))
```

Step 2:

With help of train-test split procedure we can estimate the performance of model.

First, we need to divide our data into features (X) and labels (y). The dataframe gets divided into X_train,X_test , y_train and y_test. X_train and y_train sets are used for training and fitting the model.

The first subset is used to fit the model and is referred to as the training dataset. The second subset is not used to train the model; instead, the input element of the dataset is provided

to the model, then predictions are made and compared to the expected values. This second dataset is referred to as the test dataset.

```
#using train-test split procedure to estimate the performance of model.


x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state = 0)
x_train.shape,x_test.shape,y_train.shape,y_test.shape
```

Step 3:

Segregate the variables according to the categorical and numerical data

Since, scaling of the data makes it easy for a model to learn and understand the problem.

We, scale numerical data using Standard scaler

```
#standardizing numerical variables


from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

#x_train data
x_train_sc = sc.fit_transform(x_train_numr)
x_train_sc = pd.DataFrame(x_train_sc, columns = ['Age','Experience','Income','CCAvg','Mortgage'])


#x_test data

x_test_sc = sc.fit_transform(x_test_numr)
x_test_sc = pd.DataFrame(x_test_sc, columns = ['Age','Experience','Income','CCAvg','Mortgage'])


#checking current shape of array
x_train_sc.shape ,x_test_sc.shape

```

```
((800, 5), (200, 5))
```

We, scale categorical data with using one hot encoding

```
#normalising categorical variables using one hot encoding

#x_train data
x_train_norm = pd.get_dummies(x_train_catg, columns = ['Family','Education'])

#x_test data
x_test_norm = pd.get_dummies(x_test_catg, columns = ['Family','Education'])

x_train_norm.shape,x_test_norm.shape
```

```
((800, 7), (200, 7))
```

Step 4:

Now, we concatenate the data sets into a single data frame and name it “x_train_data_final’ & x_test_data_final’

DATA EVALUATION:

Part 1: Discriminant Analysis

Build a Discriminant Analysis Model to predict whether the person is likely to accept the bank’s offer for a personal loan. If necessary, create new variables to improve the model performance.

SOLUTION:

Discriminant Analysis is used for classification and creates a discriminant function.
Discriminant Analysis results in a single composite “Z” score - Altman’s Z Score or Discriminant Score

Linear Discriminant analysis is used as a dimensionality reduction technique.
It is used to solve classification problems in machine learning

In this case,
Our dependent variable “Y” is Personal Loan

Class is of two types; one is who accepts personal loan and one who rejects

Discriminant Analysis is applied to dependent variable had two or more category/groups and these categories/groups should be mutually exclusive.

As we completed our feature scaling in above step, we will fit our LDA model to the data.

```
In [12]:
lda = Lda(n_components = 1)
x_train_lda = lda.fit_transform(x_train_data_final, y_train)
x_train_lda
```

```
Out[12]: array([[ 3.93210792e-01],
                [-2.83611072e-01],
                [ 6.48268276e-01],
                [ 7.36156105e-01],
                [ 6.10154427e-02],
                [-4.97954119e+00],
                [ 6.13365614e-01],
                [-2.61413399e-01],
                [ 1.25546649e+00],
                [-1.33884321e+00],
                [-6.84793462e-01],
                [ 1.18788584e-01],
                [-3.39994396e-01],
                [-5.07184891e+00],
                [ 1.74917408e-03],
                [-2.99787683e+00],
                [-3.10528123e+00],
                [-9.40632014e-01],
```

Now, Predict the Test Set results:

```
In [13]:
#prediction

y_pred = lda.predict(x_test_data_final)
y_pred
```

```
Out[13]: array([0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0], dtype=int64)
```

Carry out significance tests using Wilk's Lambda

SOLUTION:

Wilks' lambda performs, in the multivariate setting, the same role as the F -test performs in one-way analysis of variance

If a large proportion of the variance is accounted for by the independent variable (grouping Variable) then it suggests that there is an effect from the grouping variable and that the groups have different mean values.

Here, we use MANOVA multivariate analysis of variance because, it determines whether multiple levels of independent variables on their own or in combination with one another have an effect on the dependent variables.

COMMENT:

- A lambda of 1.00 occurs when observed group means are equal (all the variance is explained by factors other than difference between those means), while a small lambda occurs when within-groups variability is small compared to the total variability.
- A small lambda indicates that group means appear to differ
- Wilks' lamda model value being less than 0.05 states that a small lambda occurs when within-groups variability is small compared to the total variability.Hence variables are significant.

CODE :

```
#wilk's lambda can be studied with propoability and wilks value
#wilks lambda ranges from 0-1 , the more complex if the value reached to 1 ,the less complex if the value is near to 'ZERO'


# add the intercept
data['Personal Loan'] = ['a' + str(x) for x in data['Personal_Loan']]

#fit manova #considering all independent variables
manova_result = MANOVA.from_formula('Age+Experience+Income+Family+CCAvg+Education+Mortgage+Securities_Account+CD_Account+Onl
print(manova_result.mv_test())
```

OUTPUT:

```
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value    Num DF  Den DF   F Value   Pr > F
----------------------------------------------------------------
          Wilks' lambda   0.0024 11.0000 988.0000 38112.0582 0.0000
         Pillai's trace   0.9976 11.0000 988.0000 38112.0582 0.0000
 Hotelling-Lawley trace 424.3245 11.0000 988.0000 38112.0582 0.0000
    Roy's greatest root 424.3245 11.0000 988.0000 38112.0582 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
      Personal_Loan       Value    Num DF   Den DF   F Value  Pr > F
----------------------------------------------------------------
           Wilks' lambda  0.5939  11.0000  988.0000  61.4126  0.0000
          Pillai's trace  0.4061  11.0000  988.0000  61.4126  0.0000
  Hotelling-Lawley trace  0.6837  11.0000  988.0000  61.4126  0.0000
     Roy's greatest root  0.6837  11.0000  988.0000  61.4126  0.0000
================================================================
```

Comment on the variables that are significant

SOLUTION:

I have considered “Age” & “Income” as variables to test whether they are significant or not.

Code:

```
# add the intercept
data['Personal Loan'] = ['a' + str(x) for x in data['Personal_Loan']]
#fit manova #checking variables (multivarite analysis is performed so we take 2 variables)
manova_result = MANOVA.from_formula('Age+Income~ Personal_Loan',data)
print(manova_result.mv_test())

```

OUTPUT:

```
                  Multivariate linear model
================================================================

----------------------------------------------------------------
       Intercept         Value  Num DF  Den DF   F Value  Pr > F
----------------------------------------------------------------
          Wilks' lambda  0.0543 2.0000 997.0000 8688.6059 0.0000
         Pillai's trace  0.9457 2.0000 997.0000 8688.6059 0.0000
 Hotelling-Lawley trace 17.4295 2.0000 997.0000 8688.6059 0.0000
    Roy's greatest root 17.4295 2.0000 997.0000 8688.6059 0.0000
----------------------------------------------------------------

----------------------------------------------------------------
       Personal_Loan      Value  Num DF  Den DF  F Value  Pr > F
----------------------------------------------------------------
           Wilks' lambda 0.7334 2.0000 997.0000 181.2474 0.0000
          Pillai's trace 0.2666 2.0000 997.0000 181.2474 0.0000
  Hotelling-Lawley trace 0.3636 2.0000 997.0000 181.2474 0.0000
     Roy's greatest root 0.3636 2.0000 997.0000 181.2474 0.0000
================================================================
```

COMMENT:

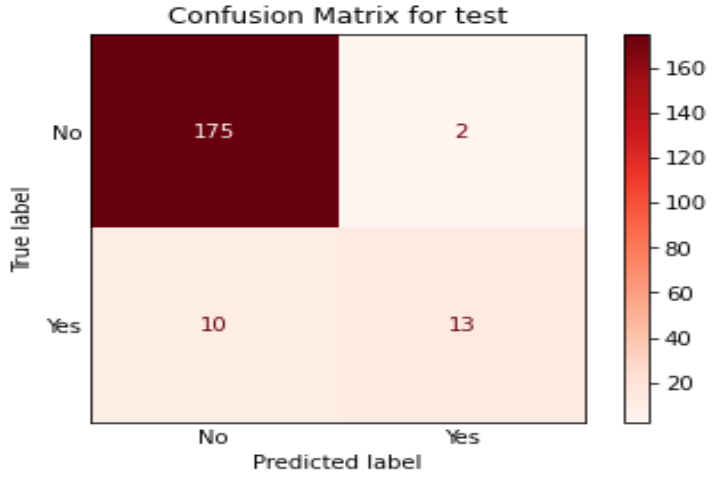
We also carried the test on using Age, Experience, and Income but Age and experience are having more than 0.05 wilk lambda value which states that they have less impact (less significance) on Personal Loan.

For income variable, Since the wilk lambda value is less than 0.05, it is a significant variable and has influence over personal loan

Q4. Create the confusion matrix and comment on the prediction accuracy.

SOLUTION:

Plotting confusion matrix



Measuring Accuracy:

Accuracy = 94%

Code Output:

```
#check for accuracy
from sklearn.metrics import accuracy_score
accuracy_score(y_test, y_pred)
```

```
0.94
```

COMMENT:

This implies are accuracy of identifying loan status is 94% right.

The bank would like to address the top 30 persons with an offer for personal loan based on the probability (propensity). Create a table displaying all the details of the "top" 30 persons who are most likely to accept the bank's offer. Make sure to include the probability of accepting the offer along with all the other details.

SOLUTION:

The probability of customer accepting the offer has been included in the output and highlighted for reference.

	Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online	CreditCard	Probability
52	61	36	184	4	2.3	2	342	0	1	1	1	0.99996354
41	54	30	194	3	6	3	587	1	1	1	1	0.99990666
59	56	32	173	1	4.6	2	88	0	1	1	0	0.9997227
179	37	12	194	4	0.2	3	211	1	1	1	1	0.9997092
22	38	13	169	1	6.8	3	0	0	1	1	1	0.99866224
189	36	12	150	4	5.4	1	0	1	1	1	0	0.97114544
81	38	13	119	1	3.3	2	0	0	1	1	1	0.93456274
34	26	2	171	3	6	2	0	0	0	1	0	0.89933275
141	56	30	111	4	0.3	1	372	1	1	1	0	0.87237409
49	30	5	98	4	1.8	3	129	1	1	1	1	0.81386598
144	42	18	153	3	5.6	1	416	0	0	0	0	0.75212908
172	56	30	158	4	6.1	1	0	0	0	0	0	0.7306394
12	50	24	133	4	1.4	2	342	0	0	0	1	0.72902142
181	51	27	92	4	3	1	0	0	1	1	1	0.69624151
4	33	7	81	2	4.5	3	187	0	1	1	1	0.62961291
117	35	9	164	2	0	1	500	0	0	0	0	0.46049991
169	63	38	134	3	4	2	0	0	0	0	1	0.44644644
77	27	2	170	3	4.7	1	0	0	0	1	0	0.44624174
156	55	30	118	4	5.6	2	0	0	0	1	0	0.44263526
148	28	3	115	4	3.1	2	0	0	0	0	0	0.43971143
108	45	21	102	4	4.7	2	81	0	0	0	0	0.40482085
134	33	8	145	1	2.7	3	0	0	0	1	0	0.39808255
158	31	7	81	2	2	2	0	0	1	1	1	0.38702422
3	31	7	81	2	2	2	0	0	1	1	1	0.38702422
161	62	36	109	4	1.7	3	0	0	0	1	0	0.35766306
96	62	38	99	4	1.7	2	0	0	0	0	0	0.33195391
121	36	6	138	1	7	3	86	0	0	1	0	0.31601393
113	65	40	114	4	3.4	2	0	0	0	0	1	0.30244915
137	48	22	174	1	2.4	1	0	0	0	1	0	0.27865463

Part 2: Logistic Regression

Build a logistic regression equation to predict whether the person is likely to accept the bank's offer for a personal loan. If necessary, create new variables to improve the model performance.

SOLUTION:

A logistic regression model predicts a dependent data variable by analysing the relationship between one or more existing independent variables.

In this case,

Our dependent variable "Y" is Personal Loan

Class is of two types; one is who accepts personal loan and one who rejects

Build a logistic regression equation to predict whether the person is likely to accept the bank's offer for a personal loan

Code and output:

```
# A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing
model = LogisticRegression(random_state=0)
x_train_log = model.fit(x_train_data_final, y_train)
print(' Personal Loan (yes or no / 0 or 1 ) Binary form:', x_train_log.classes_)
print(' Intercept :',x_train_log.intercept_)
print(' Coefficients :',x_train_log.coef_)
```

```
Personal Loan (yes or no / 0 or 1 ) Binary form: [0 1]
Intercept : [-3.19474344]
Coefficients : [[-0.62415589 -1.04097256  0.61535543  1.04811348 -1.38020673  0.52219747
  0.85634972  0.1343337  -0.08230283  2.2497211   0.06813401  0.19052073
 -0.66347851  1.9847788  -0.47907282 -1.12747107]]
```

Logistic Regression is a classification algorithm.

It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables.

Checking for the accuracy of the model:

Code and output:

```
print('Accuracy:',accuracy_score(y_test,y_pred))
```

```
Accuracy: 0.965
```

Predict the Test Results:

In [39]:
```
#Analysing the Model
y_pred = model.predict(x_test_data_final)
y_pred = pd.DataFrame(y_pred, columns = ['Predicted Personal Loan_LR'])
y_pred
```

Out[39]:

	Predicted Personal Loan_LR
0	0
1	0
2	0
3	0
4	0
...	...
195	0
196	0
197	0
198	0
199	0

200 rows × 1 columns

Observations:

Logit Regression Results

Dep. Variable:	Personal Loan	No. Observations:	800
Model:	Logit	Df Residuals:	788
Method:	MLE	Df Model:	11
Date:	Sun, 13 Nov 2022	Pseudo R-squ.:	0.5742
Time:	18:03:40	Log-Likelihood:	-108.84
converged:	True	LL-Null:	-255.64
Covariance Type:	nonrobust	LLR p-value:	1.947e-56

	coef	std err	z	P>|z|	[0.025	0.975]
const	-13.1263	4.181	-3.140	0.002	-21.321	-4.932
Age	0.0468	0.157	0.298	0.766	-0.261	0.354
Experience	-0.0451	0.157	-0.287	0.774	-0.354	0.263
Income	0.0540	0.007	8.101	0.000	0.041	0.067
Family	0.7976	0.186	4.293	0.000	0.433	1.162
CCAvg	-0.0144	0.095	-0.153	0.879	-0.200	0.171
Education	1.3032	0.244	5.341	0.000	0.825	1.782
Mortgage	0.0011	0.002	0.657	0.511	-0.002	0.004
Securities Account	-1.5107	0.752	-2.008	0.045	-2.985	-0.036
CD Account	3.3043	0.844	3.916	0.000	1.651	4.958
Online	-0.6763	0.393	-1.723	0.085	-1.446	0.093
CreditCard	-1.8696	0.597	-3.132	0.002	-3.039	-0.700

Carry out the omnibus test to test whether the model as a whole is significant. Comment on the result of the omnibus test.

The omnibus test is a likelihood-ratio chi-square test of the current model versus the null model and the significance value of < 0.05 indicates that the current model outperforms the null model.
So, chi square test of independence tests is an omnibus test and is used to test whether the model as a whole is significant or not significant.

Hypothesis:

Ho: The Model is insignificant

H1: The Model is significant

```
## Calculating  the G-stats from Logistic regression Summary Table

G_stats = -2*((-108.84)/(-256.64))
print(G_stats)
import scipy
from scipy.stats import chi2



chi2.pdf(-0.636,12)
```

```
-0.8481920199501247

0.0
```

Since, the p-value < 0.05 we can comment that model is significant.

Hence, we are rejecting the null hypothesis since our model is significant.

Test the hypothesis that βj = 0 for all βj, where βj indicates the coefficient corresponding to jth explanatory variable. Comment on the result of the hypothesis tests.

Wald's test is used to test the significant of each of the individual β.

Thus, our hypothesis

H0: β equal to '0'
H1: β not equal to '0'

The statistic for Wald's test follows Chi-square distribution with one degree of freedom. The statistic for each β_j is calculated as

$$W = \left(\frac{\hat{\beta}_j - 0}{S_e(\hat{\beta}_j)}\right)^2,$$

where $S_e(\hat{\beta}_j)$ is the standard error of the estimate $\hat{\beta}_j$

Building the model:

```
#building the model

logistic_reg = sm.Logit(y_train, x_train).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.141853
         Iterations 9
```

Conduct a Wald test for equality of multiple coefficients:

```
# Conduct a Wald test for equality of multiple coefficients

x_vars = logistic_reg.summary2().tables[1].index
wald_str = ' = '.join(list(x_vars[6:-1]))
print(wald_str)
wald_test = logistic_reg.wald_test(wald_str) # joint test
print(wald_test)
```

```
Mortgage = Securities Account = CD Account = Online
<Wald test (chi2): statistic=[[19.24524365]], p-value=0.00024326189943249874, df_denom=3>
```

Comment:

Since, the p-value > 0.05 we can comment that model is significant.

Hence, we are rejecting the null hypothesis since our model is significant.

Carry out the hypothesis test that the model fits the data. Comment on the results.

Hypothesis test:

In [38]:
```
red_default_ll = red_result_sm_lg.llf
print(f"Reduced model Log likelihood : {red_default_ll}")
```

```
Reduced model Log likelihood : -258.74235696838724
```

In [39]:
```
#calculate likelihood ratio Chi-Squared test statistic
LR_statistic = -2*(red_default_ll)



print(f"Chi-Squared test-statistic : {LR_statistic}")
```

```
Chi-Squared test-statistic : 517.4847139367745
```

In [40]:
```
#calculate p-value of test statistic using 2 degrees of freedom
p_val = stats.chi2.sf(LR_statistic, 2)



print(f"P value : {p_val}")
```

```
P value : 4.262085267547921e-113
```

Comment:

This means that the full model and nested model fit the data differently. Thus, we should use full data because additional predictor variable offers significant improvement in fit.

The bank would like to address the top 30 persons with an offer for personal loan based on the probability (propensity). Create a table displaying all the details of the "top" 30 persons who are most likely to accept the bank's offer. Make sure to include the probability of accepting the offer along with all the other details.

SOLUTION:

The probability of customer accepting the offer has been included in the output and highlighted for reference.

Age	Experience	Income	Family	CCAvg	Education	Mortgage	Securities Account	CD Account	Online	CreditCard	probability
54	30	194	3	6	3	587	1	1	1	1	0.995526444
61	36	184	4	2.3	2	342	0	1	1	1	0.993890176
37	12	194	4	0.2	3	211	1	1	1	1	0.992227094
56	32	173	1	4.6	2	88	0	1	1	0	0.973189103
26	2	171	3	6	2	0	0	0	1	0	0.924935954
38	13	169	1	6.8	3	0	0	1	1	1	0.920881897
36	12	150	4	5.4	1	0	1	1	1	0	0.784953069
56	30	158	4	6.1	1	0	0	0	0	0	0.731015842
42	18	153	3	5.6	1	416	0	0	0	0	0.722018479
50	24	133	4	1.4	2	342	0	0	0	1	0.703702958
27	2	170	3	4.7	1	0	0	0	1	0	0.625570602
28	3	115	4	3.1	2	0	0	0	0	0	0.602541304
55	30	118	4	5.6	2	0	0	0	1	0	0.579151168
36	6	138	1	7	3	86	0	0	1	0	0.544980526
33	8	145	1	2.7	3	0	0	0	1	0	0.539276568
62	36	109	4	1.7	3	0	0	0	1	0	0.514866437
63	38	134	3	4	2	0	0	0	0	1	0.508515128
45	21	102	4	4.7	2	81	0	0	0	0	0.506217222
56	30	111	4	0.3	1	372	1	1	1	0	0.456796647
30	5	98	4	1.8	3	129	1	1	1	1	0.442515098
35	9	164	2	0	1	500	0	0	0	0	0.440802942
62	38	99	4	1.7	2	0	0	0	0	0	0.41888537
48	22	174	1	2.4	1	0	0	0	1	0	0.373674475
65	40	114	4	3.4	2	0	0	0	0	1	0.357701151
38	13	119	1	3.3	2	0	0	1	1	1	0.350296926
45	20	94	3	0.5	3	0	0	0	0	0	0.308997472
38	12	89	4	1.4	2	0	0	0	0	0	0.27869055
54	28	79	4	2.6	3	0	0	0	0	0	0.266573984
41	17	140	1	3.5	1	342	0	0	0	0	0.234258227

Compare the above list of 30 persons against the 30 persons obtained from Discriminant Analysis (part 1). Comment on the similarities and dissimilarities.

SOLUTION:

Comparing 30 persons of Model 1 (Lda) vs 30 persons of Model 2 (Logistic Regression)

```
In [54]: top_30_customer_comparision = pd.DataFrame(np.array([final_model_2.index, final_model_1.index]).T, columns = ["Logistic regr
         top_30_customer_comparision
         #top_30_customer_comparision.to_clipboard()
```

Out[54]:

	Logistic regression	Lda
0	41	52
1	52	41
2	179	59
3	59	179
4	34	22
...	...	...
195	35	100
196	123	7
197	7	32
198	0	0
199	58	58

The above image consists of the summary of attributes of both the datasets. First being the top 30 with LDA and the second one is the top 30 with logistic regression.

Observations:

23 records are same in both the tables which implies 76% of the same customers are observed for both the models

Categories:

Range of age seems to be almost the same and mean in both the cases.

Range of experience is also almost same and mean also varies which is very negligible.

The minimum income in both is mildly different, so is the mean.