FRAUD AND RISK ANALYTICS

ASSIGNMENT 1

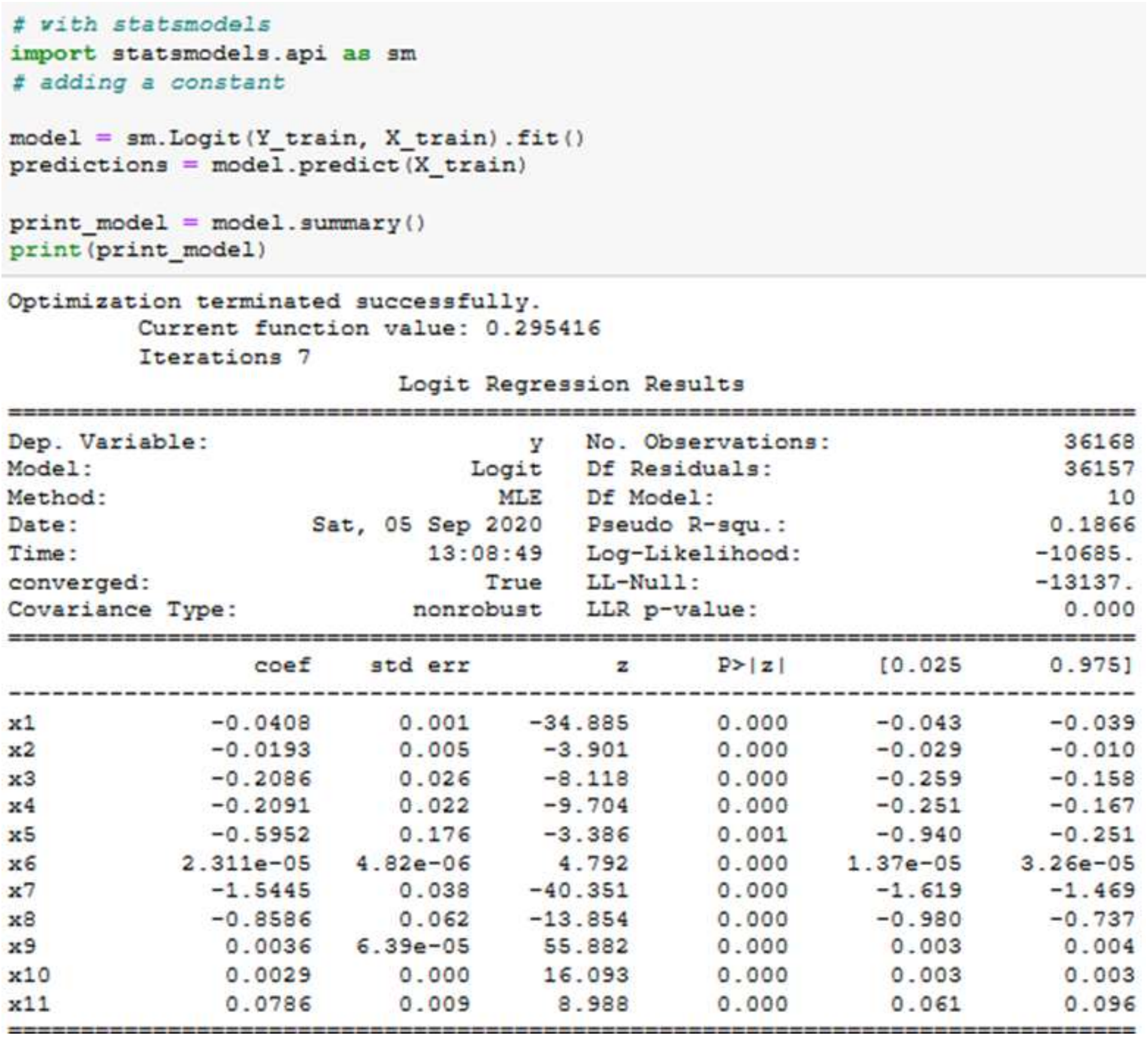
36_Ashana Mehta

Bank Tele Marketing Campaign Case Study

Problem Statement –

To predict whether the customer will take a term loan or not, as a result of the tele-marketing campaign.

1. Logistic Regression Output (with training data only) and Interpretation



```python
# with statsmodels
import statsmodels.api as sm
# adding a constant

model = sm.Logit(Y_train, X_train).fit()
predictions = model.predict(X_train)

print_model = model.summary()
print(print_model)
```

```
Optimization terminated successfully.
         Current function value: 0.295416
         Iterations 7
                           Logit Regression Results
==============================================================================
Dep. Variable:                      y   No. Observations:                36168
Model:                          Logit   Df Residuals:                    36157
Method:                           MLE   Df Model:                           10
Date:                Sat, 05 Sep 2020   Pseudo R-squ.:                  0.1866
Time:                        13:08:49   Log-Likelihood:                -10685.
converged:                       True   LL-Null:                       -13137.
Covariance Type:            nonrobust   LLR p-value:                     0.000
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
x1            -0.0408      0.001    -34.885      0.000      -0.043      -0.039
x2            -0.0193      0.005     -3.901      0.000      -0.029      -0.010
x3            -0.2086      0.026     -8.118      0.000      -0.259      -0.158
x4            -0.2091      0.022     -9.704      0.000      -0.251      -0.167
x5            -0.5952      0.176     -3.386      0.001      -0.940      -0.251
x6          2.311e-05   4.82e-06      4.792      0.000    1.37e-05    3.26e-05
x7            -1.5445      0.038    -40.351      0.000      -1.619      -1.469
x8            -0.8586      0.062    -13.854      0.000      -0.980      -0.737
x9             0.0036   6.39e-05     55.882      0.000       0.003       0.004
x10            0.0029      0.000     16.093      0.000       0.003       0.003
x11            0.0786      0.009      8.988      0.000       0.061       0.096
==============================================================================
```

- **Pseudo R – Square**

The pseudo R-square for the model is 0.1866. Thus, the model improves very less over the null model. However, the pseudo r-square value should only be used to compare different models and one must not judge a model entirely based on it.

The pseudo R-square here is calculated as 1 – (LL / LLNull).

Since the general R square model is not applicable over logistic regression models, the pseudo R-square is calculated. Some of the commonly used methods are Cox & Snell and Nagelkerke. Here Nagelkerke method is used to represent Pseudo R-square.

- **Log Likelihood Significance**

Log-Likelihood: -10686

It is the natural logarithm of the Maximum Likelihood Estimation (MLE) function which is the optimization process of finding the set of parameters which result in best fit.

LL-Null: -13137

It is the result of the maximum log-likelihood function when only an intercept is included.

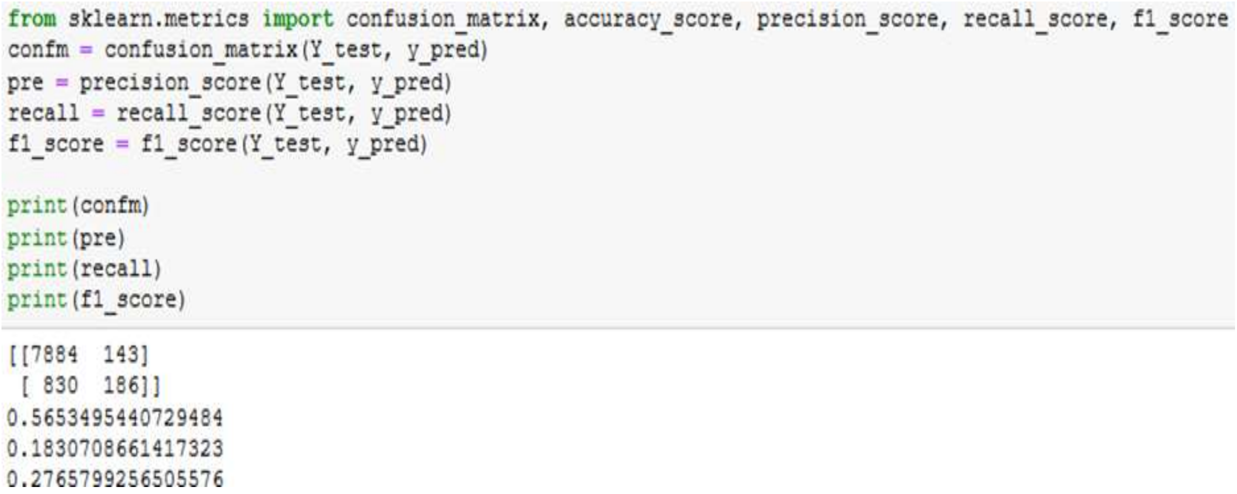
Log p-value – 0.000
This indicates that the values derived for log-likelihood and LL-Null are significant.

- **Significance of all the variables**

The variables x1, x2, x3, …, x11, all have a P > |z| value less than 0.05, thus all the variables used in the logistic regression model are significant.

2. Confusion Matrix output and Interpretation

A confusion matrix, also known as an error matrix, is a summarized table used to assess the performance of a classification model. The number of correct and incorrect predictions are summarized with count values and broken down by each class.



```
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score
confm = confusion_matrix(Y_test, y_pred)
pre = precision_score(Y_test, y_pred)
recall = recall_score(Y_test, y_pred)
f1_score = f1_score(Y_test, y_pred)

print(confm)
print(pre)
print(recall)
print(f1_score)
```

```
[[7884  143]
 [ 830  186]]
0.5653495440729484
0.1830708661417323
0.2765799256505576
```

In our case the components and count values are –

Confusion Matrix	**Predicted 0s**	**Predicted 1s**
Actual 0s	TN (7884)	FP (143)
Actual 1s	FN (830)	TP (186)

Since the values of TNs and TPs are greater than FNs and FPs, the model has correctly predicted the correct values in negative as well as positive cases. (i.e. 0s and 1s)

But at the same time, we can see that in negative case, the values are much greater than that in positive case; thereby indicating an imbalanced dataset.

Hyper tuning factors

- **Accuracy** - This is simply equal to the proportion of predictions that the model classified correctly. **(TP + TN)/ (TP + TN + FP + FN) = 0.89**

- **Precision** - Precision is also known as positive predictive value and is the proportion of relevant instances among the retrieved instances. In other words, it answers the question “What proportion of positive identifications was actually correct?” The higher, the better.
  (TP / (TP + FP)) = 0.565

- **Recall** - Recall, also known as the sensitivity, hit rate, or the true positive rate (TPR), is the proportion of the total amount of relevant instances that were actually retrieved. It answers the question “What proportion of actual positives was identified correctly?” The higher, the better
  (TP / (TP + FN)) = 0.183

- **<u>F1 score</u>** - The F1 score is a measure of a test's accuracy — it is the harmonic mean of precision and recall. It can have a maximum score of 1 (perfect precision and recall) and a minimum of 0. Overall, it is a measure of the preciseness and robustness of your model.
  $\mathbf{2 * (}\boldsymbol{Precision*Recall}\mathbf{)\ /\ (}\boldsymbol{Precision+Recall}\mathbf{) = 0.2765}$