6211 Assignment 1

Question 1.)

Table

Description automatically generated

c)

* Variables 'ACADEMIC\_INTEREST\_1', 'ACADEMIC\_INTEREST\_2', 'IRSCHOOL', 'CONTACT\_CODE1', 'CONTACT\_DATE' are dropped as mentioned in the question.
* Column ‘Level\_Year’ is to be dropped as the variable has only 1 unique value ‘FR22’ and has no impact on the predictive models.
* ‘Satscore’ and ‘telecq’ have more than 70% missing values and will not make a good feature. Hence, dropping these two columns.
* Variables Total\_Contacts, Mailq have been for the regression model as they have **VIF** values greater than the threshold value of **10** during model building in the later steps.

d) Variable ‘Enroll’ is the target variable.

Question 2.)

* Categorical variables ‘ETHNICITY’, ‘TERRITORY’ and ‘INSTATE’ are to be dummied.
* Category ‘A’ from variable ETHNICITY is left out and similarly TERRITORY ‘0’ along with INSTATE ‘N’ are left out.

Question 3.)

* Variable **Imputation** is required for model building but **Transformation** isn’t required.
* Variable ‘**Ethnicity’** has been imputed with value **‘C’** as it is the most frequently occurred value.
* Variable ‘**Territory’** has been imputed with value **‘0’** as the missing value percentage is close to 0, and so filling it with ‘0’ won’t make any difference to the model.
* Variable ‘**sex’** has been imputed with value **1** as it is the most frequently occurred value.
* Variable ‘**avg\_income** has been imputed with its **mean value** as the variable is normally distributed**.**
* Variable ‘**distance’** has been imputed with its **mean value** as the variable is right skewed.
* Variables ‘Total\_Contacts’ and ‘SELF\_INIT\_CNTCTS’ have been grouped into bins of 0 and 1. ( if value greater than 0 then assigned value as 1 else 0).

Question 4.)

1. Logit Model Summary

Table

Description automatically generated

1. Tree Plot

A picture containing text, orange

Description automatically generated

Question 5.)

I’ll choose Logistic Regression as its ROC AUC value **(0.685)** is higher than Decision Tree ROC AUC value **(0.636).**

Question 6.)

* Given dataset is highly imbalanced and requires resampling for building better models.
* Based on the regression model, variables **AVG\_INCOME, DISTANCE, ETHNICITY** and **TERRITORY** are the significant variables in the given dataset as their respective **p-values** are less than the threshold of **0.05** whereas the rest variables have higher p-values than 0.05. Keeping the rest variables constant, with every unit increase in either **AVG\_INCOME or DISTANCE or ETHNICITY** or **TERRITORY**, the target value **Enroll** decreases.

Question 7.)

#!/usr/bin/env python

# coding: utf-8

# In[1]:

import pandas as pd

import numpy as np

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from sklearn.model\_selection import train\_test\_split

import statsmodels.api as sm

from sklearn.metrics import confusion\_matrix

from sklearn.metrics import roc\_curve

from sklearn.metrics import roc\_auc\_score

from matplotlib import pyplot as plt

from sklearn import tree

from sklearn.tree import export\_text

# In[2]:

ap\_df = pd.read\_csv("inq2022.csv")

# In[3]:

ap\_df.head()

# In[4]:

ap\_df.info()

# In[5]:

ap\_df.describe(include = 'all')

# In[6]:

ap\_df.isnull().sum()/len(ap\_df)

# In[7]:

ap\_df = ap\_df.drop(columns=['ACADEMIC\_INTEREST\_1','ACADEMIC\_INTEREST\_2','IRSCHOOL','CONTACT\_CODE1','CONTACT\_DATE'])

# In[8]:

ap\_df.describe(include='all')

# In[9]:

ap\_df.info()

# In[10]:

ap\_df.isna().sum()/len(ap\_df)

# In[11]:

ap\_df = ap\_df.drop(columns=['satscore','telecq'])

# In[12]:

ap\_df = ap\_df.drop(columns=['LEVEL\_YEAR'])

# In[13]:

ap\_df.Enroll.value\_counts()

# In[14]:

cat\_columns = ['ETHNICITY','TERRITORY','Instate']

# In[15]:

for col in cat\_columns:

print(ap\_df[col].value\_counts())

print(len(ap\_df))

print(ap\_df.isna().sum())

# In[16]:

ap\_df['ETHNICITY'] = ap\_df['ETHNICITY'].fillna('C') ## Most frequent occurence

ap\_df['TERRITORY'] = ap\_df['TERRITORY'].fillna('0') ## Null % is close to 0, so filling with least occurence value as it doesn't have much impact on the model

ap\_df['sex'] = ap\_df['sex'].fillna(1) ## Highest occurence

ap\_df['avg\_income'] = ap\_df['avg\_income'].fillna(ap\_df['avg\_income'].mean()) ## Right skewed so filling nulls with mean

ap\_df['distance'] = ap\_df['distance'].fillna(ap\_df['distance'].mean()) ## Right skewed so filling nulls with mean

# In[17]:

ap\_df =pd.get\_dummies(ap\_df, drop\_first=True)

ap\_df\_tree = ap\_df.copy() ## Making a replica for Decision Tree Model

ap\_df.describe()

# In[18]:

ap\_df.info()

# In[19]:

ap\_df[['TOTAL\_CONTACTS','SELF\_INIT\_CNTCTS','TRAVEL\_INIT\_CNTCTS',

'SOLICITED\_CNTCTS','REFERRAL\_CNTCTS','CAMPUS\_VISIT',

'sex','mailq','premiere','interest','stucar','init\_span','int1rat',

'int2rat','hscrat','avg\_income','distance']].skew(axis = 0, skipna = True)

# In[20]:

ap\_df[['TOTAL\_CONTACTS','SELF\_INIT\_CNTCTS','TRAVEL\_INIT\_CNTCTS',

'SOLICITED\_CNTCTS','REFERRAL\_CNTCTS','CAMPUS\_VISIT',

'sex','mailq','premiere','interest','stucar','init\_span','int1rat',

'int2rat','hscrat','avg\_income','distance']].hist(bins=30, figsize=(20, 15))

# In[21]:

ap\_df['TOTAL\_CONTACTS'] = np.where(ap\_df['TOTAL\_CONTACTS']>0,1,0)

ap\_df['SELF\_INIT\_CNTCTS'] = np.where(ap\_df['SELF\_INIT\_CNTCTS']>0,1,0)

# In[22]:

ap\_df.drop(columns=['Enroll']).hist(bins=30, figsize=(20, 15))

# Considering VIF threshold as 10

# In[23]:

ap\_vif = pd.DataFrame()

ap\_vif["feature"] = ap\_df.drop(columns=['Enroll']).columns

ap\_vif["VIF"] = [variance\_inflation\_factor(ap\_df.drop(columns=['Enroll']).values, i)

for i in range(len(ap\_df.drop(columns=['Enroll']).columns))]

print(ap\_vif)

# In[24]:

ap\_df.corr()

# In[25]:

ap\_df = ap\_df.drop(columns=['TOTAL\_CONTACTS']) ### Highest VIF value

# In[26]:

ap\_vif = pd.DataFrame()

ap\_vif["feature"] = ap\_df.drop(columns=['Enroll']).columns

ap\_vif["VIF"] = [variance\_inflation\_factor(ap\_df.drop(columns=['Enroll']).values, i)

for i in range(len(ap\_df.drop(columns=['Enroll']).columns))]

print(ap\_vif)

# In[27]:

ap\_df.corr()

# In[28]:

ap\_df = ap\_df.drop(columns=['mailq']) ### Highest VIF value

# In[29]:

ap\_vif = pd.DataFrame()

ap\_vif["feature"] = ap\_df.drop(columns=['Enroll']).columns

ap\_vif["VIF"] = [variance\_inflation\_factor(ap\_df.drop(columns=['Enroll']).values, i)

for i in range(len(ap\_df.drop(columns=['Enroll']).columns))]

print(ap\_vif)

# ## Regression Model

# In[30]:

ap\_X\_train, ap\_X\_val, ap\_y\_train, ap\_y\_val = train\_test\_split(ap\_df.drop(columns=['Enroll']),ap\_df['Enroll'], test\_size=0.3,random\_state=0)

ap\_log\_reg = sm.Logit(ap\_y\_train, ap\_X\_train).fit()

print(ap\_log\_reg.summary())

# In[31]:

ap\_prediction\_prob = ap\_log\_reg.predict(ap\_X\_val)

ap\_prediction = list(map(round, ap\_prediction\_prob))

confusion\_matrix(ap\_y\_val,ap\_prediction)

# In[32]:

lr\_auc = roc\_auc\_score(ap\_y\_val, ap\_prediction\_prob)

print('Logistic: ROC AUC=%.3f' % (lr\_auc))

# In[33]:

lr\_fpr, lr\_tpr, \_ = roc\_curve(ap\_y\_val, ap\_prediction\_prob)

plt.plot(lr\_fpr, lr\_tpr, marker='.')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()

# ## Tree Model

# In[34]:

ap\_X\_train, ap\_X\_val, ap\_y\_train, ap\_y\_val = train\_test\_split(ap\_df\_tree.drop(columns=['Enroll']),ap\_df\_tree['Enroll'], test\_size=0.3,random\_state=0)

ap\_dtree = tree.DecisionTreeClassifier(max\_depth=4,min\_samples\_split=30)

ap\_dtree = ap\_dtree.fit(ap\_X\_train, ap\_y\_train)

# In[35]:

ap\_r = export\_text(ap\_dtree, feature\_names=list(ap\_X\_train.columns.values))

print(ap\_r)

# In[36]:

plt.figure(figsize=[25,20])

tree.plot\_tree(ap\_dtree,

feature\_names=list(ap\_X\_train.columns.values),

class\_names=True,

filled=True)

plt.show()

# In[37]:

ap\_prediction =ap\_dtree.predict(ap\_X\_val)

confusion\_matrix(ap\_y\_val,ap\_prediction)

# In[38]:

ap\_dtree.score(ap\_X\_val,ap\_y\_val)

# In[39]:

ap\_prediction\_prob = ap\_dtree.predict\_proba(ap\_X\_val)

ap\_tree\_auc = roc\_auc\_score(ap\_y\_val, ap\_prediction\_prob[:,1])

print('Decision Tree: ROC AUC=%.3f' % (ap\_tree\_auc))

# In[40]:

tree\_fpr, tree\_tpr, \_ = roc\_curve(ap\_y\_val, ap\_prediction\_prob[:,1])

plt.plot(tree\_fpr, tree\_tpr, marker='.')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.show()