**DSBA 6211 Assignment 1**

Dataset Overview:

Variables used in analysis

|  |  |  |
| --- | --- | --- |
| Variable Name | Data Type | Description |
| ETHNICITY | object | Demographic factor |
| TERRITORY | object | Geographic region |
| TOTAL\_CONTACTS | int64 | Number of total contacts |
| SELF\_INIT\_CNTCTS | int64 | Student-initiated contacts |
| TRAVEL\_INIT\_CNTCTS | int64 | Travel-related contacts |
| SOLICITED\_CNTCTS | int64 | University-initiated contacts |
| REFERRAL\_CNTCTS | int64 | Referral-based contacts |
| CAMPUS\_VISIT | int64 | Indicator of campus visit |
| LEVEL\_YEAR | object | Academic level/year |
| satscore | float64 | Standardized test score |
| sex | float64 | Gender (to be recoded categorical) |
| mailq | int64 | Response to mailed materials |
| telecq | float64 | Response to telephone contacts |
| premiere | int64 | Engagement indicator |
| interest | int64 | Interest measure |
| stucar | int64 | Student characteristics indicator |
| init\_span | int64 | Timing of initial contact |
| int1rat | float64 | Enrollment rate for interest code 1 |
| int2rat | float64 | Enrollment rate for interest code 2 |
| hscrat | float64 | Enrollment rate for high school |
| avg\_income | float64 | Average income (socioeconomic factor) |
| distance | float64 | Distance from university |
| Instate | bool | In-state vs out-of-state |

Rejected Variables:

|  |  |
| --- | --- |
| Variable Name | Reason for Rejection |
| ID | Administrative only, not predictive |
| ACADEMIC\_INTEREST\_1 | Replaced by **INT1RAT** |
| ACADEMIC\_INTEREST\_2 | Replaced by **INT2RAT** |
| IRSCHOOL | Replaced by **HSCRAT** |
| CONTACT\_DATE | Determined irrelevant by Enrollment Mgmt |
| CONTACT\_CODE1 | Determined irrelevant by Enrollment Mgmt |

c. Other Variables in consideration to drop:

LEVEL\_Year – All records are indicating freshmen class. This is the same for each so we can drop since this will have no affect on target variable.

Dropping “satscore”: although this could be a strong predictor on academic ability, 70% of the data is missing. Using the mean would make this model bias on the small number of available test scores

Dropping “telecq” as well since 77% of the data is missing

d. The target variable is “Enroll” (int64). Outcome variable: 1 = enrolled, 0 = not enrolled

2. Dummy Coding

a. Ethnicity, Territory, sex – because these are categorical; we are also going to change “Instate” to Boolean (1 or 0). I also gave Ethnicity a category “Unknown” for NaN values.

b. The only variables that need to be dropped are for columns: [“Ethnicity”, “Territory”].

i. For ethnicity, White needs to be dropped since that is the baseline. However, using drop\_first=True keeps the remaining 4 variables.

ii. Territory is the same using drop\_first =True as the baseline

3. Variable imputation

a. Avg\_income is needed due to at least 20% of the data was missing and is a strong predictor variable. Distance is also needed since 20% was missing as well.

b. Avg\_income filled na values with median because the mean was higher than the median.

4. Variable transformation for regression

a. Distance was log transformed because it was heavily skewed. The students that lived +1000 miles away would affect the model

5. Model results:

Regression

A screenshot of a computer screen

AI-generated content may be incorrect.

Regression results:

Accuracy: 0.968664601931135

AUC: 0.753609615493962

Tree plot model:

A diagram of a company

AI-generated content may be incorrect.

6. The model I would choose is decision tree for interpretability. The AUC performs better however, with the class imbalance of predicting more not enrolled students. The tree shows clear rules such as students in-state and log distance < 4, then enrollment is likely.

7. The decision tree results:

Accuracy: 0.4701038440517398

AUC: 0.7612067917299055

As we can see it fails to predict enrolled students but this could be based on the imbalance class level. Due to the dataset having more not enrolled students vs enrolled.

**Based on the results:**

**Distance**: Students living farther away are significantly less likely to enroll. The log of distance showed a strong negative affect.

**Ethnicity**: Some ethnicity categories (B, H, I, N, O, and Unknown) were associated with lower enrollment rates

**Contact Categories**: Overall contact was drop since this is an enrollment data analyses, self-initiated contacts and referrals were moderately associated with increased likelihood of enrollment.

**In-State Status**: Being in-state is weakly positive but not a strong driver itself

**Data Management Recommendations**

The balance of data collection is showing imbalance (96% non-enrolled vs 4% enrolled). Future models would perform better if we can enrich data on enrolled students. Maybe oversample or capture details on just enrolled students.

Missing data:

Percentage breakdown:

Enroll

0 96.8%– not enrolled

1 3.14%- enrolled

Income was replaced with median to fill missing records. This weakens predictive value since 22% of the data was missing and a strong variable

Distance was missing a similar amount and had to be transformed using log to reduce outlier effects.

Also take Ethnicity into consideration since 21% was missing and showed on the regression evaluation as a strong predictor

Python Code:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

dfJS = pd.read\_csv('inq2025.csv')

dfJS.head()

dfJS.isnull().sum()/len(dfJS)

# Dropping columns based on assignement documentation on rejected variables

# Dropping LEVEL\_YEAR because all records are freshmen

rejected\_columns = [

"ID",

"ACADEMIC\_INTEREST\_1",

"ACADEMIC\_INTEREST\_2",

"IRSCHOOL",

"CONTACT\_DATE",

"CONTACT\_CODE1",

'LEVEL\_YEAR', # Same for all

'telecq', # missing 77%

'satscore' # missing 70%

]

dfJS = dfJS.drop(columns=rejected\_columns)

# Plot histograms for all numeric columns in dfJS

dfJS.hist(figsize=(16, 12), bins=30, edgecolor="black")

plt.suptitle("Histograms of All Numeric Columns in Raw Data", fontsize=18)

plt.tight\_layout(rect=[0, 0, 1, 0.96])

plt.show()

print(dfJS.describe(include='all'))

dfJS.info()

dfJS.isnull().sum()/len(dfJS)

# Replace Nan values for Avg income since it is a key predictor and only 22% missing

# Calculate mean and median for avg\_income

mean\_income = dfJS["avg\_income"].mean(skipna=True)

median\_income = dfJS["avg\_income"].median(skipna=True)

print("Mean income:",mean\_income)

print("Median income:",median\_income)

# In[240]:

# Using Median income since mean is higher

dfJS["avg\_income"] = dfJS['avg\_income'].fillna(median\_income)

plt.figure(figsize=(8,6))

dfJS["distance"].hist(bins=30, edgecolor="black")

plt.title("Distribution of Distance")

plt.xlabel("Distance (miles)")

plt.ylabel("Frequency")

plt.axvline(dfJS["distance"].mean(), color='red', linestyle='dashed', linewidth=2, label=f"Mean: {dfJS['distance'].mean():.0f}")

plt.axvline(dfJS["distance"].median(), color='green', linestyle='dashed', linewidth=2, label=f"Median: {dfJS['distance'].median():.0f}")

plt.legend()

plt.show()

# Log the distance because it reduces the influence of extreme outliers (students living +1000 miles away)

median\_distance = dfJS['distance'].median(skipna=True)

dfJS['distance'] = dfJS['distance'].fillna(median\_distance)

# Creating log transformation

dfJS['distance\_log'] = np.log1p(dfJS['distance'])

dfJS.isnull().sum()/len(dfJS)

column\_names = list(dfJS.columns)

x = dfJS[column\_names]

print(x)

categorical\_vars = ["ETHNICITY", "TERRITORY", "sex"]

dfJS['ETHNICITY'] = dfJS['ETHNICITY'].fillna("Unknown")

dfJS\_model = pd.get\_dummies(dfJS, columns=categorical\_vars,drop\_first=True)

dfJS\_model["Instate"] = dfJS\_model["Instate"].astype(int)

dfJS\_model.head()

# Heat map Correlaltion check on predictor variables

corr\_matrixJS = dfJS\_model.corr()

plt.figure(figsize=(14,10))

sns.heatmap(corr\_matrixJS, cmap="coolwarm", center=0, annot=False)

plt.title("Correlation Heatmap of Independent Variables", fontsize=16)

plt.show()

# Variance Inflation Factor (VIF) Analysis

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

from statsmodels.tools.tools import add\_constant

XJS = dfJS\_model.drop(columns=["Enroll"])

XJS = XJS.select\_dtypes(include=[np.number]).replace([np.inf, -np.inf], np.nan)

XJS = XJS.fillna(XJS.mean())

XJS\_const = add\_constant(XJS)

vif\_dataJS = pd.DataFrame()

vif\_dataJS["Variable"] = XJS\_const.columns

vif\_dataJS["VIF"] = [variance\_inflation\_factor(XJS\_const.values, i) for i in range(XJS\_const.shape[1])]

print(vif\_dataJS.sort\_values(by="VIF", ascending=False))

# Make sure the target is numeric

dfJS\_model["Enroll"] = dfJS\_model["Enroll"].astype(int)

# Convert Instate to numeric (0/1)

dfJS\_model["Instate"] = dfJS\_model["Instate"].astype(int)

# Force all columns to numeric

dfJS\_model = dfJS\_model.apply(pd.to\_numeric, errors="coerce")

# Check again

print(dfJS\_model.dtypes.value\_counts())

# Convert boolean columns to int (0/1)

for col in dfJS\_model.select\_dtypes(include=["bool"]).columns:

dfJS\_model[col] = dfJS\_model[col].astype(int)

# Double check after conversion

print(dfJS\_model.dtypes.value\_counts())

import statsmodels.api as sm

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

from sklearn.metrics import accuracy\_score, roc\_auc\_score, confusion\_matrix, classification\_report

# Ensure everything is numeric

dfJS\_model = dfJS\_model.apply(pd.to\_numeric, errors="coerce")

# Split data

yJS = dfJS\_model["Enroll"]

XJS = dfJS\_model.drop(columns=["Enroll", "TOTAL\_CONTACTS", "distance"])

XJS\_train, XJS\_test, yJS\_train, yJS\_test = train\_test\_split(

XJS, yJS, test\_size=0.3, random\_state=0, stratify=yJS

)

# Reduce predictors

important\_varsJS = [

"distance\_log", "SELF\_INIT\_CNTCTS", "TRAVEL\_INIT\_CNTCTS",

"SOLICITED\_CNTCTS", "REFERRAL\_CNTCTS", "CAMPUS\_VISIT",

"avg\_income", "Instate"

] + [col for col in XJS\_train.columns if col.startswith("ETHNICITY\_")]

XJS\_train\_reduced = sm.add\_constant(XJS\_train[important\_varsJS])

XJS\_test\_reduced = sm.add\_constant(XJS\_test[important\_varsJS])

regressionJS = sm.Logit(yJS\_train, XJS\_train\_reduced).fit(maxiter=200)

print(regressionJS.summary())

# Evaluate on test set

yJS\_pred\_prob = regressionJS.predict(XJS\_test\_reduced)

yJS\_pred = (yJS\_pred\_prob >= 0.2).astype(int)

print("\nAccuracy:", accuracy\_score(yJS\_test, yJS\_pred))

print("AUC:", roc\_auc\_score(yJS\_test, yJS\_pred\_prob))

print("Confusion Matrix:\n", confusion\_matrix(yJS\_test, yJS\_pred))

from sklearn.metrics import roc\_curve, auc

fprJS, tprJS, \_ = roc\_curve(yJS\_test, yJS\_pred\_prob)

roc\_aucJS = auc(fprJS, tprJS)

plt.figure(figsize=(6,6))

plt.plot(fprJS, tprJS, label=f"ROC Curve (AUC = {roc\_aucJS:.3f})")

plt.plot([0,1], [0,1], linestyle="--", color="orange")

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve - Logistic Regression")

plt.legend(loc="lower right")

plt.show()

from sklearn.tree import DecisionTreeClassifier, plot\_tree

treeJS = DecisionTreeClassifier(

criterion="gini",

max\_depth=3,

class\_weight="balanced",

random\_state=0

)

treeJS.fit(XJS\_train, yJS\_train)

yJS\_tree\_pred = treeJS.predict(XJS\_test)

yJS\_tree\_pred\_prob = treeJS.predict\_proba(XJS\_test)[:, 1]

print("Accuracy:", accuracy\_score(yJS\_test, yJS\_tree\_pred))

print("AUC:", roc\_auc\_score(yJS\_test, yJS\_tree\_pred\_prob))

print("Confusion Matrix:\n", confusion\_matrix(yJS\_test, yJS\_tree\_pred))

plt.figure(figsize=(20,10))

plot\_tree(treeJS, feature\_names=XJS.columns, class\_names=["Not Enrolled", "Enrolled"], filled=True, fontsize=8)

plt.show()

# Count enrollments vs non-enrollments

enroll\_countsJS = dfJS\_model["Enroll"].value\_counts()

print(enroll\_countsJS)

print("\nPercentage breakdown:")

print(enroll\_countsJS / len(dfJS\_model) \* 100)