Predictive Modeling of College Admission Success Using Logistic Regression: An Investigation of Relevant Factors in Preliminary Year Students

A Punchline report – Durga Prasad N

The Issues:

The data contains 33 columns that provide information about college students completing a preliminary year, including factors such as high school GPA, SAT score, federal ethnic group, gender, Pell Grant eligibility, attendance at orientation and experience days, resident status, athletic participation, completion of summer bridge, dropout proneness, predicted academic difficulty, educational stress, and receptivity to institutional help. The data also includes scores on various factors such as receptivity to personal counseling, social engagement, career guidance, and financial guidance. Other columns provide information on completed campus and community service requirements, faculty and peer mentor meetings attended, workshops attended, F17 GPA, S18 GPA, CUM GPA, number of credits earned, completed Connect program, reasons for not completing Connect, retention status, and reasons for not being retained.

We address the questions:

- The report seeks to identify which variables are most useful in predicting success or failure and explore the relationships between relevant factors and college admission success of preliminary year students.
- Evaluate the goodness of fit of the model to the data and calculate the predicted response value for a given set of predictor values, and assess the accuracy of the prediction.

Findings:

Based on our analysis of the data using the provided feature name mapping, we have identified several key factors that are associated with student success at this university.

Our findings indicate that High School GPA , SAT Score , and Federal Ethnic Group are important predictors of student success, with being White (Ethnic Group) associated with higher success rates. Additionally, attending campus events and completing community service requirements were found to be important predictors of success.

We also found that being a male student is associated with lower success rates, while being a Pell Grant recipient or attending an experience day is associated with higher success rates. Dropout proneness, predicted academic difficulty, and receptivity to institutional help, academic assistance, and personal counselling were also significant predictors of success.

Furthermore, our model achieved an 82% accuracy rate, correctly predicting the outcome for 9 out of 11 students from the test data. The F1 score of the model was found to be 0.86.

Discussion:

The findings of our analysis have important implications for universities and their efforts to improve student success. One key takeaway is that a combination of both academic and non-academic factors can significantly impact student outcomes. Therefore, universities should consider addressing these factors in their recruitment, retention, and support programs. And the early identification and support for students who may be at risk of dropping out or experiencing academic difficulty can greatly improve their chances of success. Receptivity to institutional help, academic assistance, and personal counseling were all significant predictors of success.

Appendix A: Methods

Data collection: The data set is a subset of the data from the book An introduction to statistical learning with R.

Variable creation: The data contains 33 columns that provide information about college students completing a preliminary year, including factors such as high school GPA, SAT score, federal ethnic group, gender, Pell Grant eligibility, attendance at orientation and experience days, resident status, athletic participation, completion of summer bridge, dropout proneness, predicted academic difficulty, educational stress, and receptivity to institutional help. The data also includes scores on various factors such as receptivity to personal counseling, social engagement, career guidance, and financial guidance. Other columns provide information on completed campus and community service requirements, faculty and peer mentor meetings attended, workshops attended, F17 GPA, S18 GPA, CUM GPA, number of credits earned, completed Connect program, reasons for not completing Connect, retention status, and reasons for not being retained. Post pre-processing the data is transformed into 36 features.

Analytic methods:

The data was loaded into a pandas dataframe and analysed for outliers and missing values. Boxplots were used to identify the outliers, and rows with outliers were dropped. Rows with more than 25 missing values were also dropped. Column names were replaced with f0, f1, ... for easier access.

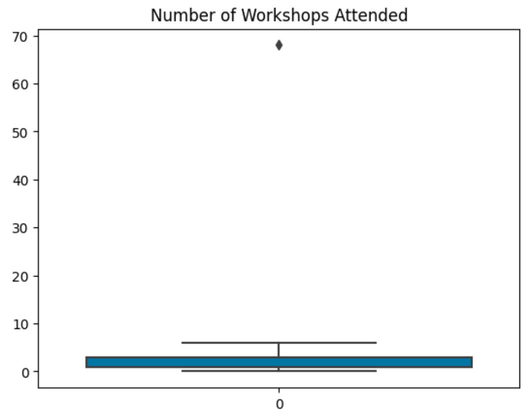
A pre-processing pipeline was created to handle missing values and categorical features. For ethnicity, the simple imputer was used to fill missing values with a constant "Not specified," and the one-hot encoder was used to handle categorical values. For all other binary features, the simple imputer with the "most_frequent" strategy was applied to fill the missing values. For gender, the simple imputer was used to fill missing values with "most_frequent," and then the one-hot encoder was applied. For all other numerical columns, the simple imputer with the "median" strategy was applied. The remaining three text columns were dropped from the dataframe.

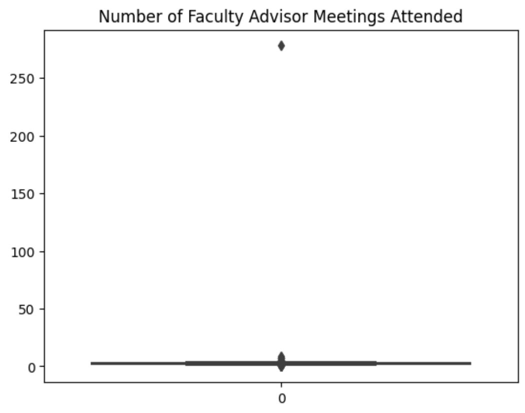
The resulting dataframe had 36 columns and was separated into train and test sets with a 90% train and 10% test split. The logistic regression class from the sklearn library was used to train the model. The model was then tested with the 10% test data, and a confusion matrix, precision-recall curve, and f1-score were computed.

Appendix B: Results

The dataset contains 108 records and 33 columns. And the descriptive stats for numerical features are computed.

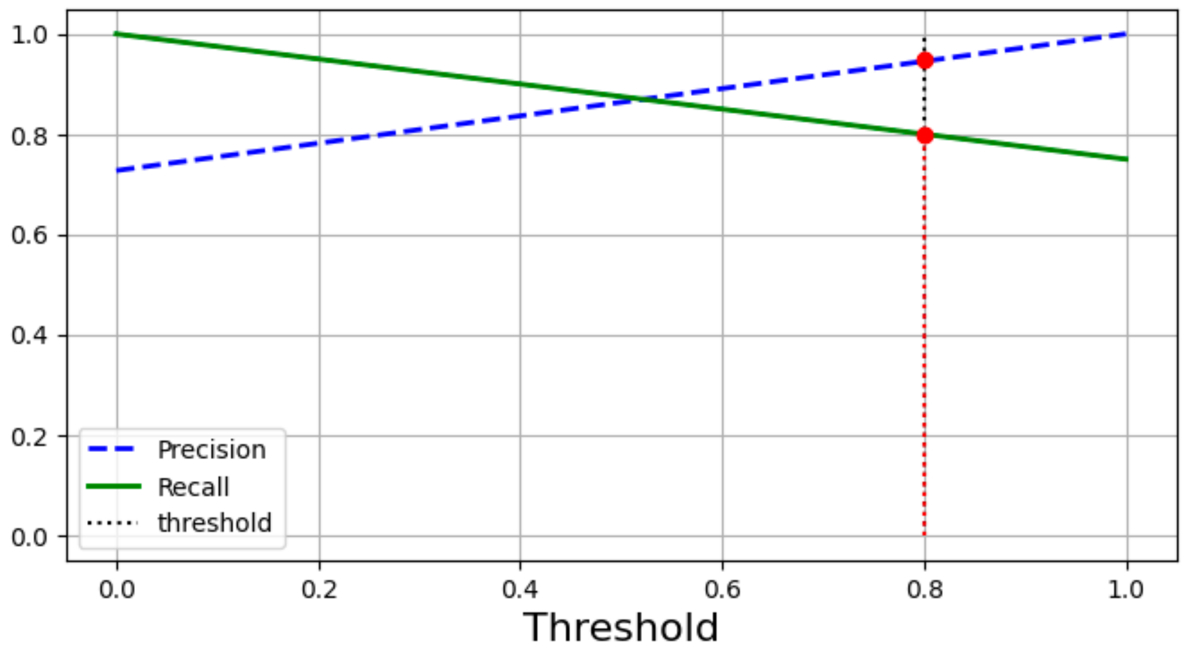
The data was preprocessed using a pipeline, which included filling missing values with appropriate strategies and encoding categorical variables. After preprocessing, the dataset consisted of 36 columns. Descriptive statistics were computed for the dataset. The majority of the students were female (54%), and the mean GPA was 3.12. The mean SAT score was 1844, and the mean high school GPA was 3.29. In terms of ethnicity, the largest group was White (62%), followed by Not specified (19%) and African American (11%).





The above boxplots confirms there are outliers in the dataset. Dropped the records with outliers and maximum missing values and then trained the model using sci-kit learn’s LogisticRegression class.

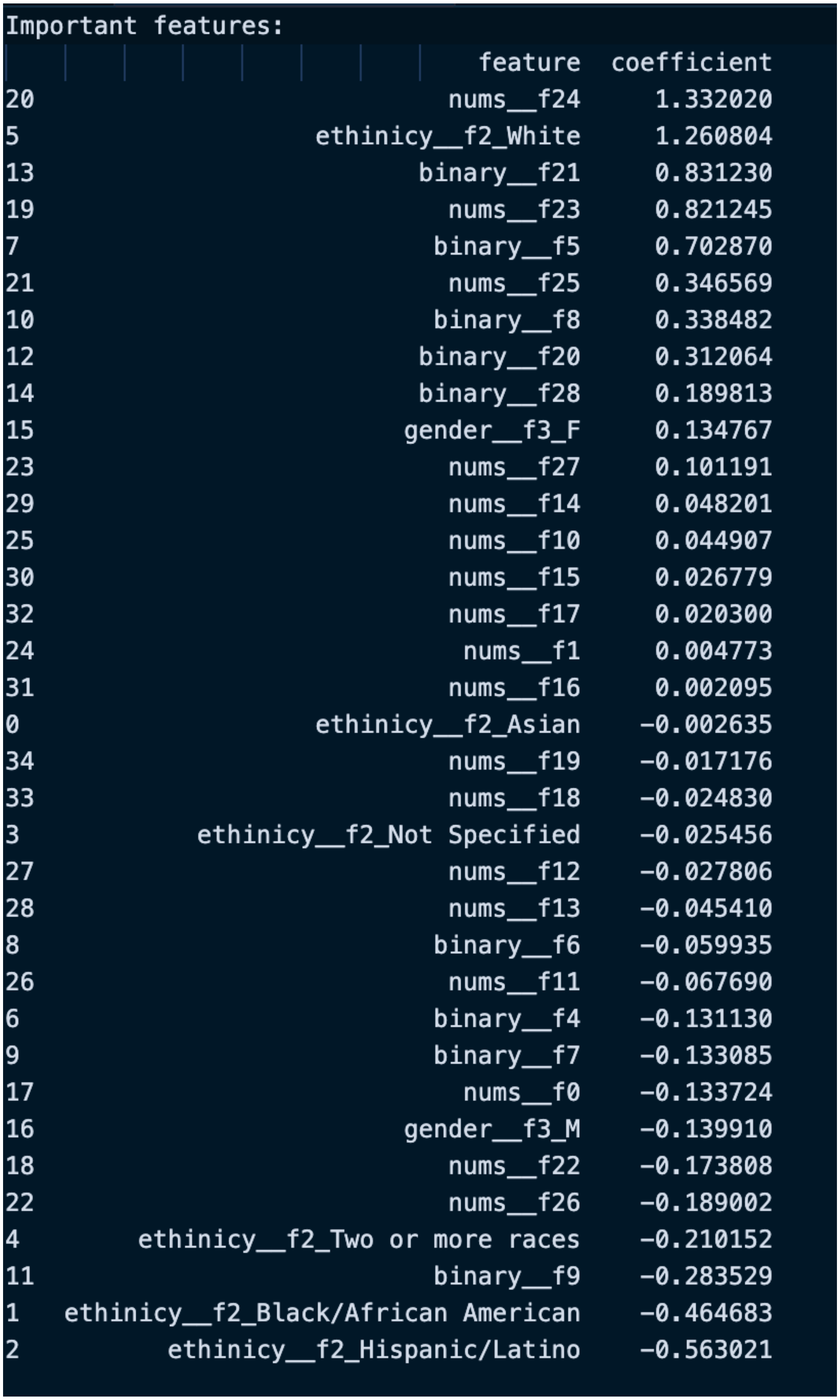
We have achieved an F1 score of 0.86, indicating that our model has a good balance between precision and recall. This suggests that our model is able to correctly identify a high percentage of positive cases while minimizing false positives and false negatives, which is a good indicator of its predictive performance.



Here is the confusion matrix values.

```
array([[3, 0], [2, 6]])
```

Here is the list features sorted based on co-efficient scores

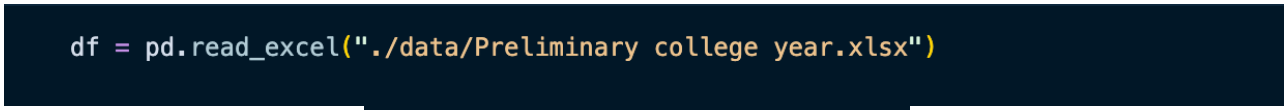


```
Important features:
                                     feature  coefficient
20                                nums__f24     1.332020
5                         ethinicy__f2_White     1.260804
13                              binary__f21     0.831230
19                                nums__f23     0.821245
7                                binary__f5     0.702870
21                                nums__f25     0.346569
10                               binary__f8     0.338482
12                              binary__f20     0.312064
14                              binary__f28     0.189813
15                              gender__f3_F     0.134767
23                                nums__f27     0.101191
29                                nums__f14     0.048201
25                                nums__f10     0.044907
30                                nums__f15     0.026779
32                                nums__f17     0.020300
24                                 nums__f1     0.004773
31                                nums__f16     0.002095
0                         ethinicy__f2_Asian    -0.002635
34                                nums__f19    -0.017176
33                                nums__f18    -0.024830
3                 ethinicy__f2_Not Specified    -0.025456
27                                nums__f12    -0.027806
28                                nums__f13    -0.045410
8                                binary__f6    -0.059935
26                                nums__f11    -0.067690
6                                binary__f4    -0.131130
9                                binary__f7    -0.133085
17                                 nums__f0    -0.133724
16                              gender__f3_M    -0.139910
18                                nums__f22    -0.173808
22                                nums__f26    -0.189002
4             ethinicy__f2_Two or more races    -0.210152
11                               binary__f9    -0.283529
1   ethinicy__f2_Black/African American    -0.464683
2           ethinicy__f2_Hispanic/Latino    -0.563021
```

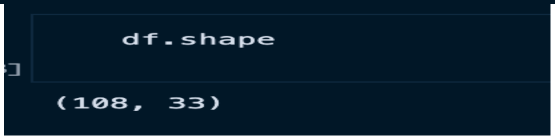
Appendix C: Code

The statistical analysis is performed using following code and a linear model is trained using sklearn package of python.

a. Importing required libraries and loading the dataset into pandas dataframe

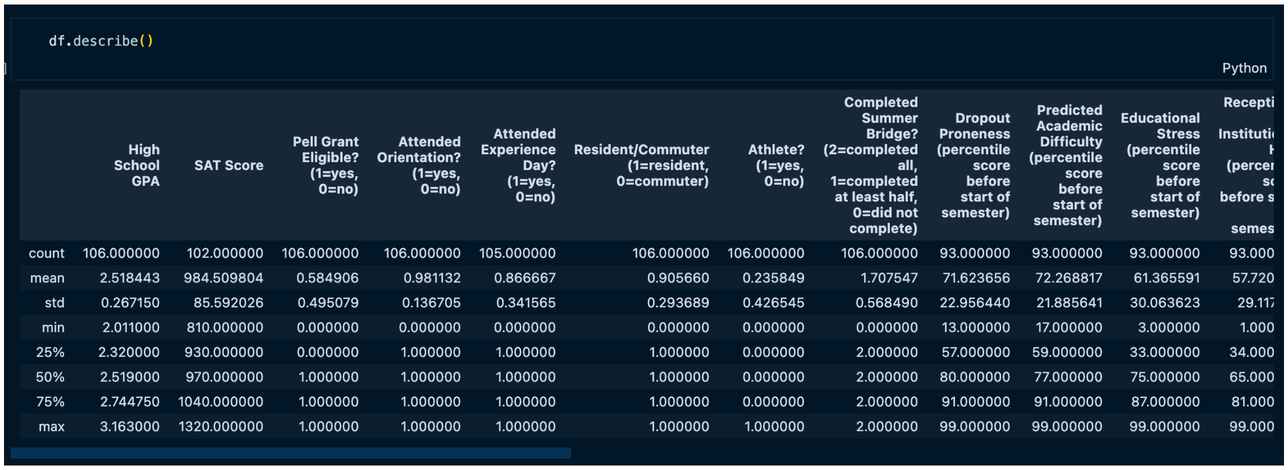


```
df = pd.read_excel("./data/Preliminary college year.xlsx")
```



```
df.shape

(108, 33)
```

b. Dataframe describe method is used to generate descriptive stats of all the numerical columns.

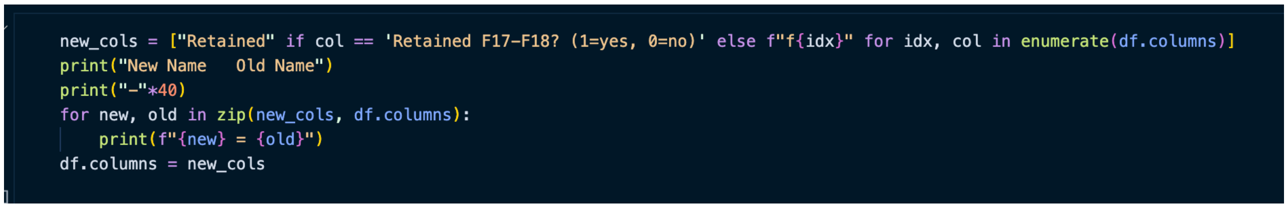


```
df.describe()
```

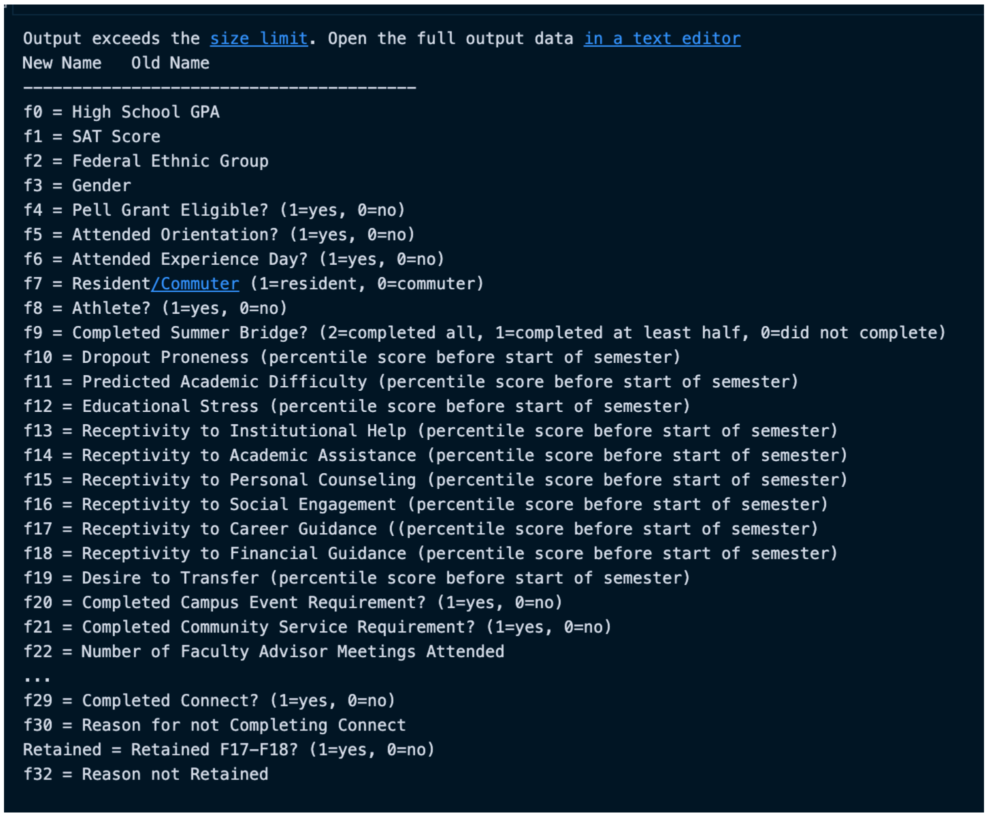
Python

	High School GPA	SAT Score	Pell Grant Eligible? (1=yes, 0=no)	Attended Orientation? (1=yes, 0=no)	Attended Experience Day? (1=yes, 0=no)	Resident/Commuter (1=resident, 0=commuter)	Athlete? (1=yes, 0=no)	Completed Summer Bridge? (2=completed all, 1=completed at least half, 0=did not complete)	Dropout Proneness (percentile score before start of semester)	Predicted Academic Difficulty (percentile score before start of semester)	Educational Stress (percentile score before start of semester)	Recepti Institutio H (percer sc before s semes
count	106.000000	102.000000	106.000000	106.000000	105.000000	106.000000	106.000000	106.000000	93.000000	93.000000	93.000000	93.000
mean	2.518443	984.509804	0.584906	0.981132	0.866667	0.905660	0.235849	1.707547	71.623656	72.268817	61.365591	57.720
std	0.267150	85.592026	0.495079	0.136705	0.341565	0.293689	0.426545	0.568490	22.956440	21.885641	30.063623	29.117
min	2.011000	810.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	13.000000	17.000000	3.000000	1.000
25%	2.320000	930.000000	0.000000	1.000000	1.000000	1.000000	0.000000	2.000000	57.000000	59.000000	33.000000	34.000
50%	2.519000	970.000000	1.000000	1.000000	1.000000	1.000000	0.000000	2.000000	80.000000	77.000000	75.000000	65.000
75%	2.744750	1040.000000	1.000000	1.000000	1.000000	1.000000	0.000000	2.000000	91.000000	91.000000	87.000000	81.000
max	3.163000	1320.000000	1.000000	1.000000	1.000000	1.000000	1.000000	2.000000	99.000000	99.000000	99.000000	99.000

c. Renamed the columns in the dataframe using following code.

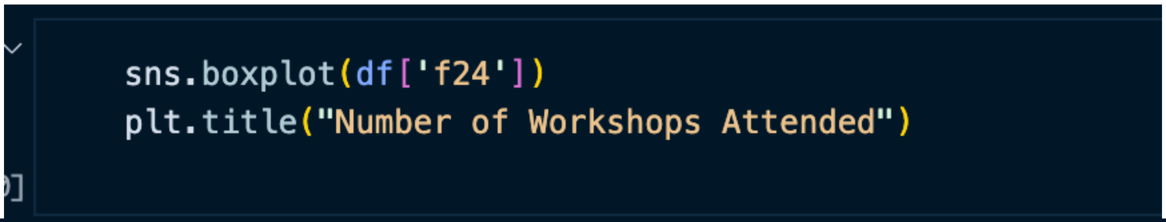


```
new_cols = ["Retained" if col == 'Retained F17-F18? (1=yes, 0=no)' else f"f{idx}" for idx, col in enumerate(df.columns)]
print("New Name   Old Name")
print("-"*40)
for new, old in zip(new_cols, df.columns):
    print(f"{new} = {old}")
df.columns = new_cols
```

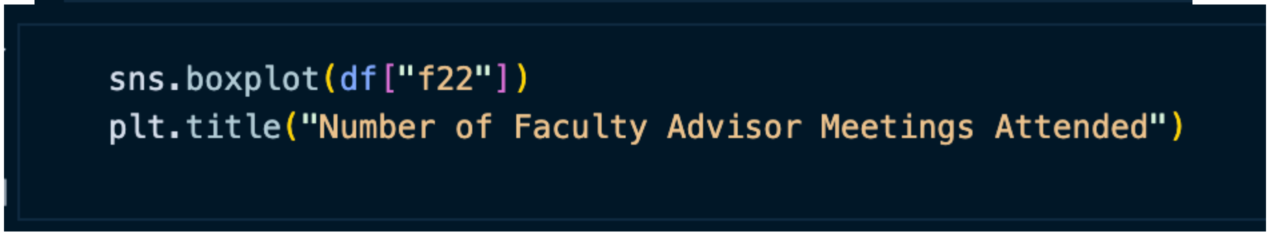


```
Output exceeds the size limit. Open the full output data in a text editor
New Name   Old Name
----------------------------------------
f0 = High School GPA
f1 = SAT Score
f2 = Federal Ethnic Group
f3 = Gender
f4 = Pell Grant Eligible? (1=yes, 0=no)
f5 = Attended Orientation? (1=yes, 0=no)
f6 = Attended Experience Day? (1=yes, 0=no)
f7 = Resident/Commuter (1=resident, 0=commuter)
f8 = Athlete? (1=yes, 0=no)
f9 = Completed Summer Bridge? (2=completed all, 1=completed at least half, 0=did not complete)
f10 = Dropout Proneness (percentile score before start of semester)
f11 = Predicted Academic Difficulty (percentile score before start of semester)
f12 = Educational Stress (percentile score before start of semester)
f13 = Receptivity to Institutional Help (percentile score before start of semester)
f14 = Receptivity to Academic Assistance (percentile score before start of semester)
f15 = Receptivity to Personal Counseling (percentile score before start of semester)
f16 = Receptivity to Social Engagement (percentile score before start of semester)
f17 = Receptivity to Career Guidance ((percentile score before start of semester)
f18 = Receptivity to Financial Guidance (percentile score before start of semester)
f19 = Desire to Transfer (percentile score before start of semester)
f20 = Completed Campus Event Requirement? (1=yes, 0=no)
f21 = Completed Community Service Requirement? (1=yes, 0=no)
f22 = Number of Faculty Advisor Meetings Attended
...
f29 = Completed Connect? (1=yes, 0=no)
f30 = Reason for not Completing Connect
Retained = Retained F17-F18? (1=yes, 0=no)
f32 = Reason not Retained
```

d. Boxplots are plotted on selected columns to confirm the outliers.

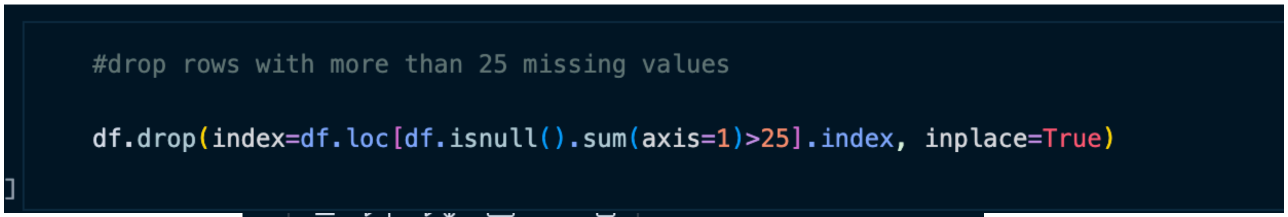


```
sns.boxplot(df['f24'])
plt.title("Number of Workshops Attended")
```

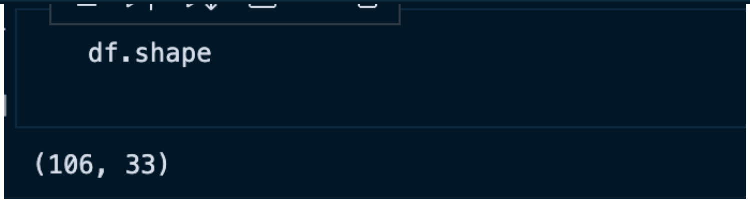


```
sns.boxplot(df["f22"])
plt.title("Number of Faculty Advisor Meetings Attended")
```

e. To drop the rows with more than 25 missing values and the resulting dataframe has 106 rows.



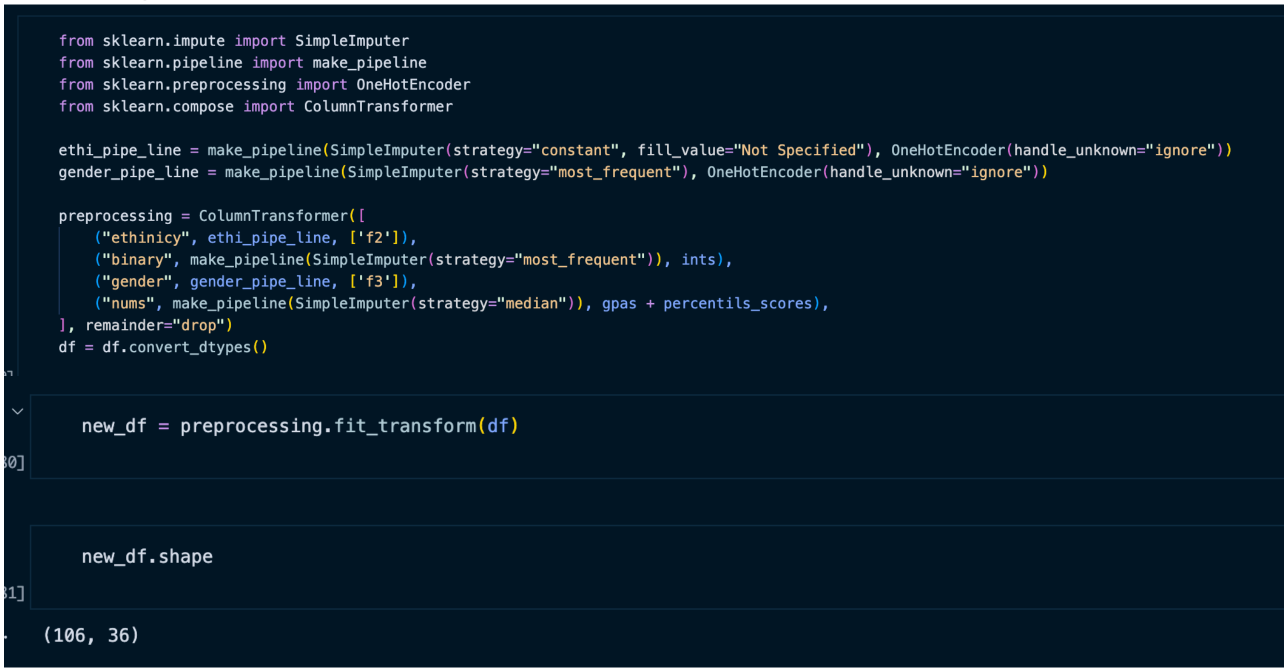
```
#drop rows with more than 25 missing values

df.drop(index=df.loc[df.isnull().sum(axis=1)>25].index, inplace=True)
```



```
df.shape
```

```
(106, 33)
```

f. The data pre-processing is done using following code and the resulting dataframe has 36 columns.



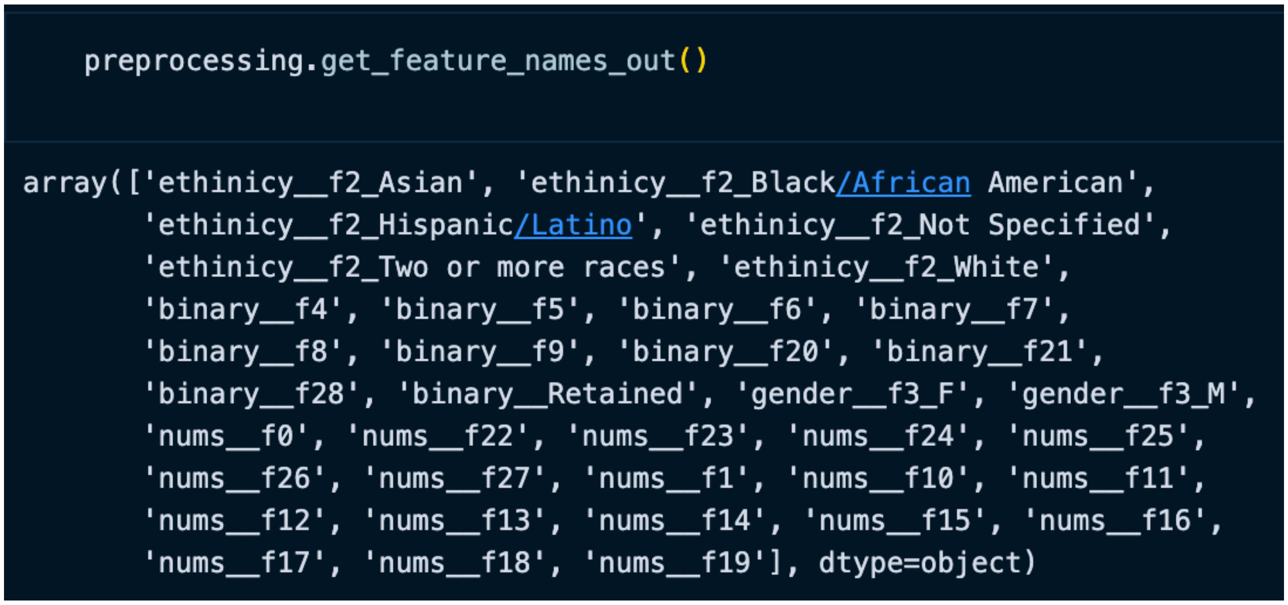
```
from sklearn.impute import SimpleImputer
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer

ethi_pipe_line = make_pipeline(SimpleImputer(strategy="constant", fill_value="Not Specified"), OneHotEncoder(handle_unknown="ignore"))
gender_pipe_line = make_pipeline(SimpleImputer(strategy="most_frequent"), OneHotEncoder(handle_unknown="ignore"))

preprocessing = ColumnTransformer([
    ("ethinicy", ethi_pipe_line, ['f2']),
    ("binary", make_pipeline(SimpleImputer(strategy="most_frequent")), ints),
    ("gender", gender_pipe_line, ['f3']),
    ("nums", make_pipeline(SimpleImputer(strategy="median")), gpas + percentils_scores),
], remainder="drop")
df = df.convert_dtypes()
```

```
new_df = preprocessing.fit_transform(df)
```

```
new_df.shape
```

```
(106, 36)
```

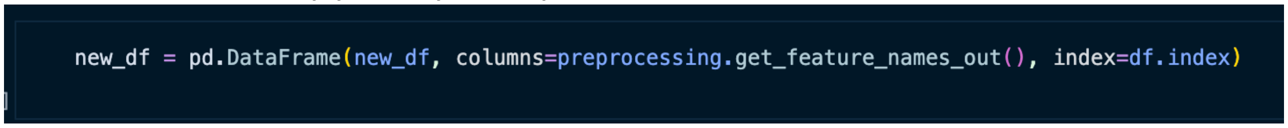
g. To check the column headers,



```
preprocessing.get_feature_names_out()
```

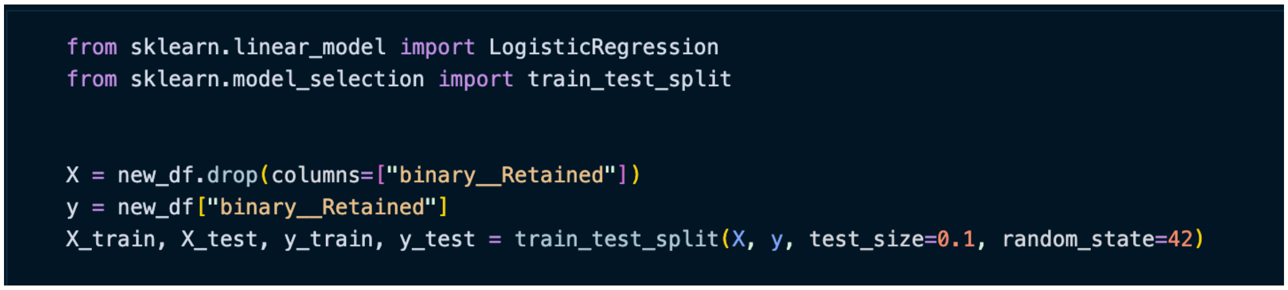
```
array(['ethinicy__f2_Asian', 'ethinicy__f2_Black/African American',
       'ethinicy__f2_Hispanic/Latino', 'ethinicy__f2_Not Specified',
       'ethinicy__f2_Two or more races', 'ethinicy__f2_White',
       'binary__f4', 'binary__f5', 'binary__f6', 'binary__f7',
       'binary__f8', 'binary__f9', 'binary__f20', 'binary__f21',
       'binary__f28', 'binary__Retained', 'gender__f3_F', 'gender__f3_M',
       'nums__f0', 'nums__f22', 'nums__f23', 'nums__f24', 'nums__f25',
       'nums__f26', 'nums__f27', 'nums__f1', 'nums__f10', 'nums__f11',
       'nums__f12', 'nums__f13', 'nums__f14', 'nums__f15', 'nums__f16',
       'nums__f17', 'nums__f18', 'nums__f19'], dtype=object)
```

h. The pre-processor genertes a numpy array as the output, below converts the numpy array into pandas dataframe.



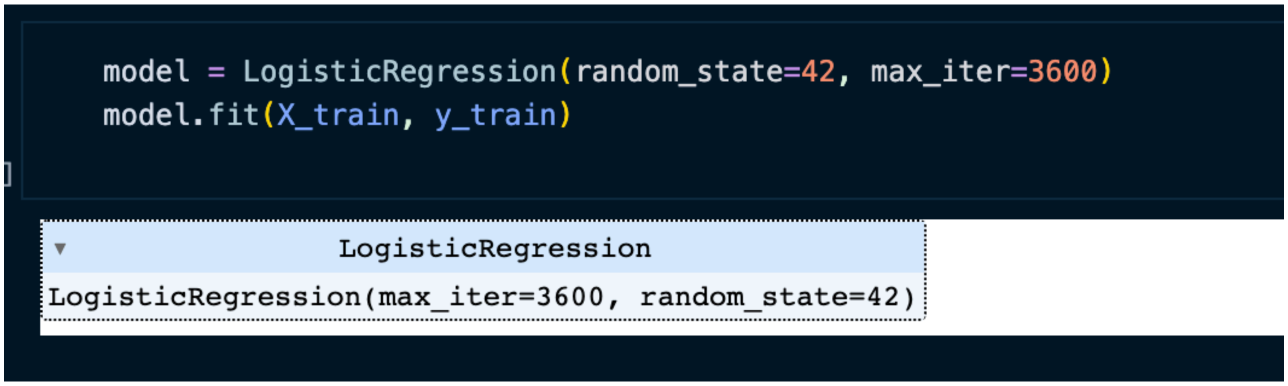
```
new_df = pd.DataFrame(new_df, columns=preprocessing.get_feature_names_out(), index=df.index)
```

i. For separating train and test data sets.



```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


X = new_df.drop(columns=["binary__Retained"])
y = new_df["binary__Retained"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=42)
```

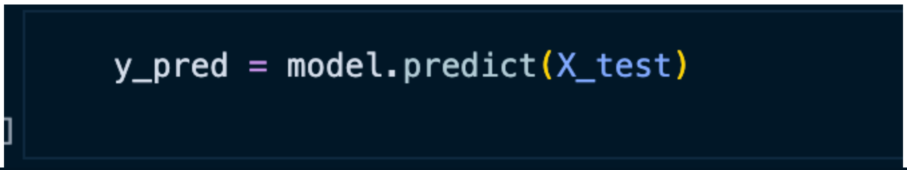
j. To train the model



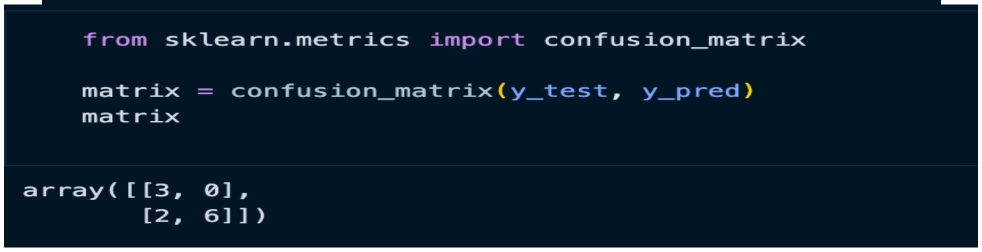
```
model = LogisticRegression(random_state=42, max_iter=3600)
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression(max_iter=3600, random_state=42)
```

k. Here is the code to make predictions on test set and computing confusion matrix.



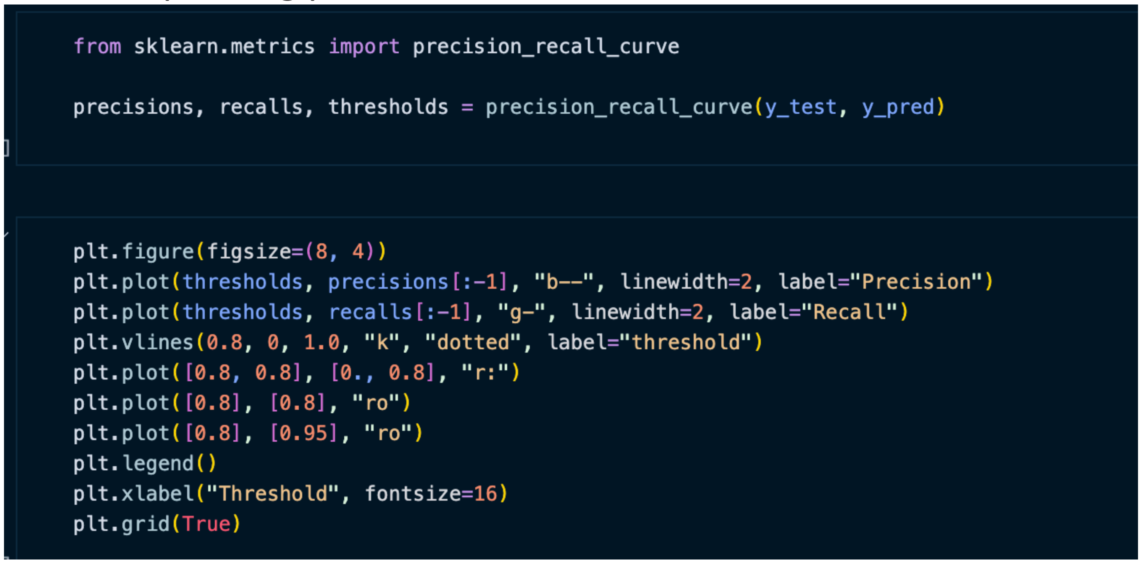
```
y_pred = model.predict(X_test)
```



```
from sklearn.metrics import confusion_matrix

matrix = confusion_matrix(y_test, y_pred)
matrix
```

```
array([[3, 0],
       [2, 6]])
```

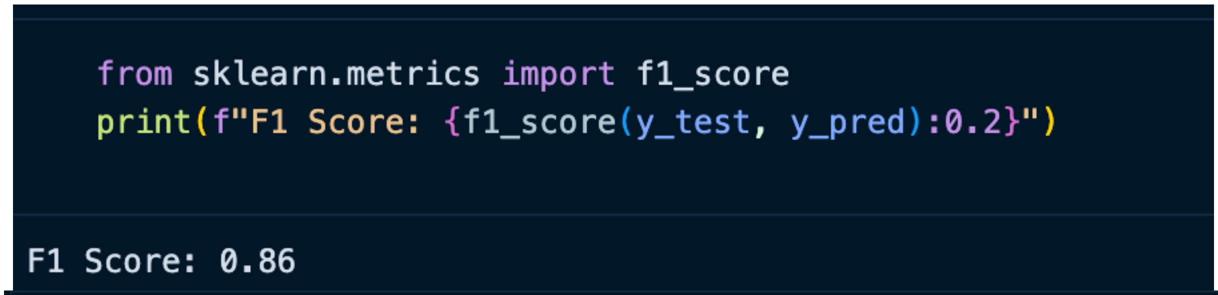
l. Code for plotting precision-recall curve



```
from sklearn.metrics import precision_recall_curve

precisions, recalls, thresholds = precision_recall_curve(y_test, y_pred)
```

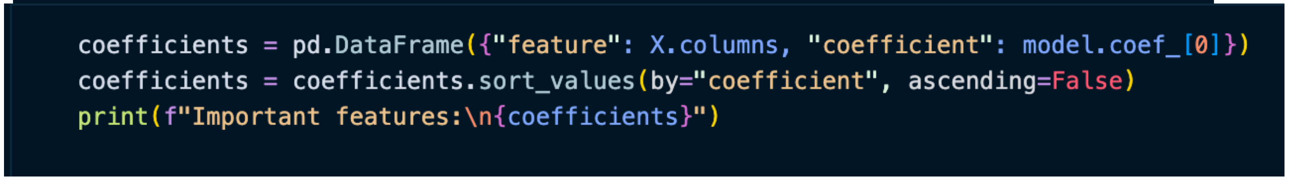
```
plt.figure(figsize=(8, 4))
plt.plot(thresholds, precisions[:-1], "b--", linewidth=2, label="Precision")
plt.plot(thresholds, recalls[:-1], "g-", linewidth=2, label="Recall")
plt.vlines(0.8, 0, 1.0, "k", "dotted", label="threshold")
plt.plot([0.8, 0.8], [0., 0.8], "r:")
plt.plot([0.8], [0.8], "ro")
plt.plot([0.8], [0.95], "ro")
plt.legend()
plt.xlabel("Threshold", fontsize=16)
plt.grid(True)
```

m. Finally, for computing f1-score and check the co-efficients



```python
from sklearn.metrics import f1_score
print(f"F1 Score: {f1_score(y_test, y_pred):0.2}")
```

```
F1 Score: 0.86
```



```python
coefficients = pd.DataFrame({"feature": X.columns, "coefficient": model.coef_[0]})
coefficients = coefficients.sort_values(by="coefficient", ascending=False)
print(f"Important features:\n{coefficients}")
```