ADS508 Data Science With Cloud Computing

Authors: Jacqueline Vo, Ben Ogle, Logan Van Dine
Company Name: AffordableU Foundation
Company Industry: Non-Profit
Company Size: 11-50
GitHub Repository: https://github.com/dsklnr/CollegeAffordability.git

Abstract

The future prospects and basic needs of students aspiring to pursue higher education are increasingly affected as college tuition costs have dramatically grown over recent years. This study aims to analyze the historical trends of college tuition costs to provide valuable insights and personalized guidance to high school students looking to enter college

Problem Statement

AffordableU Foundation is dedicated to providing the opportunity of a higher education to more students around the United States. Since most students view the cost of college tuition to be the largest obstacle, finding universities in their region or state that are affordable options is necessary. Higher education leads to more opportunities, which can increase the quality of life for many individuals. Additionally, the tangible skills that come from higher education enable low-income individuals to pursue higher-earning careers that can further provide monetary and educational support to their families.

AffordableU Foundation is interested in increasing the number of people who hold college degrees because it will lead to a better future for society. Having a society of individuals with different backgrounds, who are skilled workers will mold a society that can provide citizens with a higher quality of life. Additionally, having a skilled workforce ultimately benefits employees, employers, and consumers.

Goals

The project is designed to thoroughly and successfully perform predictive analytics and prescriptive analytics that can positively impact students. The AffordableU Foundation aims to pursue social justice by identifying universities that can provide the highest return on investment for those impacted by historical and systemic injustices such as low-income or first-generation students. Additionally, this project aims to provide personalized recommendations for at least 1,000 high school students considering pursuing higher education, and increase the conversion rate of high school students entering college for our 10 target high schools that currently have a low high school graduation rate.

Non-Goals

While our aim remains to increase the number of high school graduates attending prestigious universities, it is not within the scope of this project to guarantee any outcomes to students nor any predicted statistics of top universities. Furthermore, there is no guarantee that the assisted student will be accepted into any university. Additionally, this project does not guarantee that the predicted tuition rate will be the definitive rate in the future as this project is not affiliated with any university enrollment committee being studied.

Data Sources

The team has sourced four data sets from several sources including: Kaggle Inc., TuitionTracker and the U.S. Department of Agriculture. The datasets are Microsoft Excel files that will be extracted from the respective website source and stored in Amazon Web Services to be transformed, loaded, and analyzed. Given that some of the datasets from the U.S. Department of Agriculture and Department of Education are highly formatted Excel spreadsheets, more data cleaning will be required for those respective datasets. Some null and irrelevant values were found in the datasets, which will need to be removed, however no other risks are anticipated. We plan to store our data in AWS S3, which can be done by converting all data files to CSV format and manually uploading the files into buckets.

TuitionTracker

- NetPrice.csv
  (3,240 rows and 36 columns)
- StickerPrice.csv
  (3,240 rows and 123 columns)

U.S. Department of Agriculture

- PovertyEstimates.csv
  (3,195 rows and 34 columns)
- Unemployment.csv
  (3,277 rows and 100 columns)

Kaggle Inc.

- college_data.csv
  (58,123 rows and 33 columns)

Data Exploration

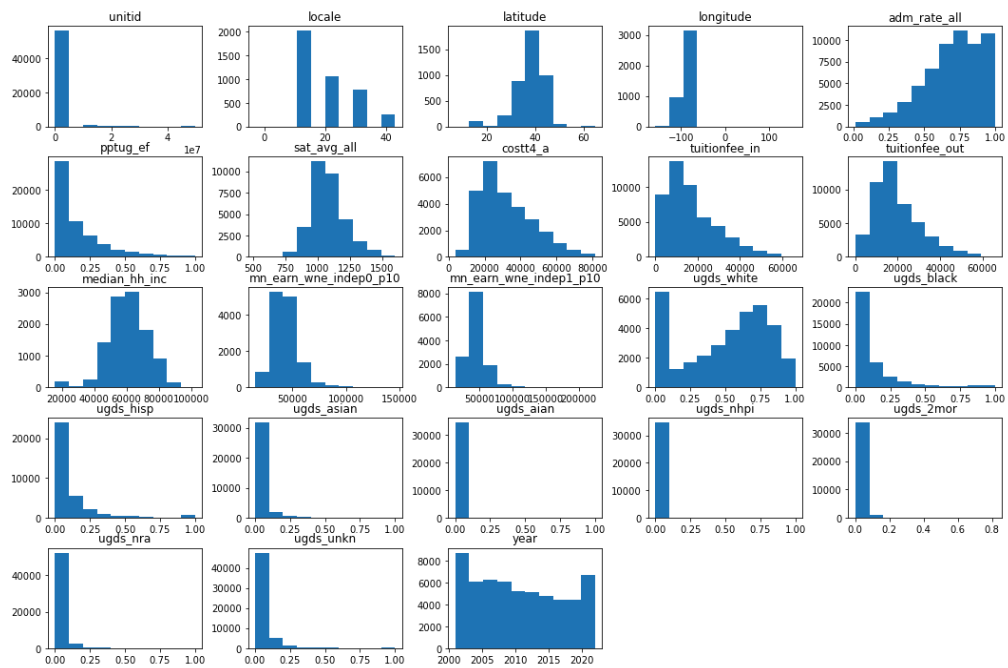
The data previously mentioned will be stored in AWS S3 buckets due to the raw CSV file size. There were two options when uploading the datasets to the S3 bucket, manually or via code. Our team chose to manually upload the data to ensure consistency across team members. Due to the sheer volume of the three data sources, a number of AWS tools were used to ingest and explore the data such as SageMaker, Athena DB, Data Wrangler, and Glue. More specifically, Athena DB was used to create several tables linked to the raw CSV datasets, which will be manipulated and processed using SQL. Additionally, Glue assisted in preparing the data to be modeled efficiently.

The exploratory data analysis provided the team with insight to the overall cleanliness and trends within the raw data. The team also determined which features would be valuable to both the data and business objectives upon combining tables. The team first used Athena DB to create data tables for the original datasets stored in the S3 bucket based on the data source: path_kaggle, path_tuition, and path_usda.

The dataset from Kaggle, hereinafter referred to as the University dataset, provided granular information on U.S. universities and had 58,123 rows and 33 columns. This dataset had geographical information on where each university was located, in addition to the admissions rate, average SAT score of students, median household income, institution type, racial demographics from the year 2001 to 2022. In analyzing the distributions of each feature as seen in Figure 1, there is a much larger distribution of data from the year 2000 compared to later years. The amount of students attending university on a part-time basis was found to be heavily skewed towards the right indicating that full-time students are much more represented in the data. Students can be enrolled in universities on a part-time basis due to life circumstances such as family commitments, jobs, or inability to afford a full-time tuition at that current time.

Figure 1

Histogram of University Numerical Features

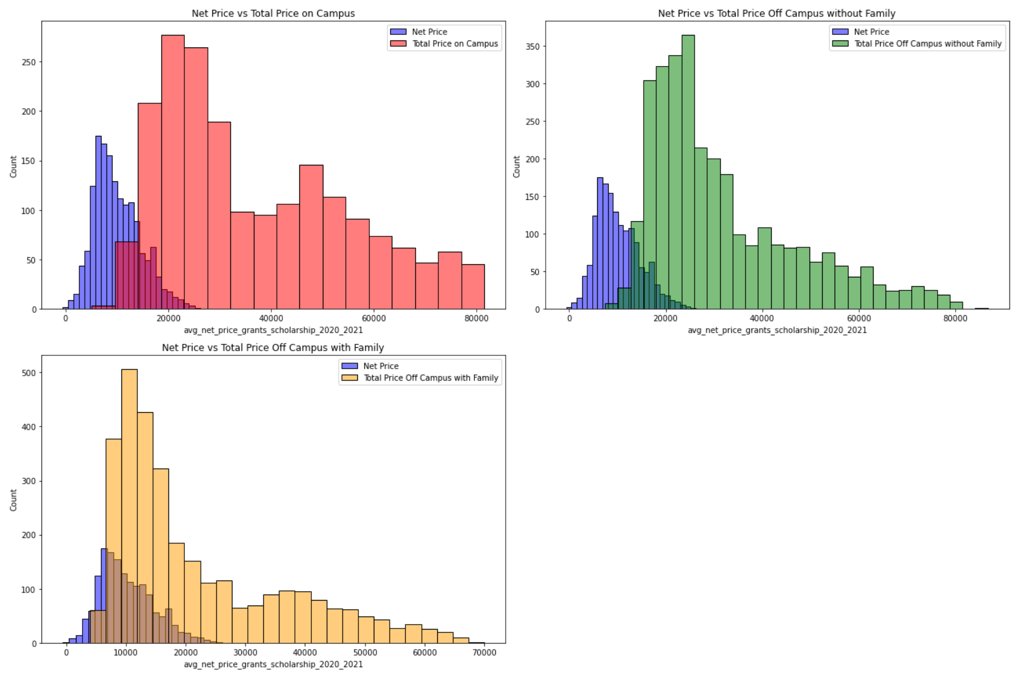


The StickerPrice and NetPrice datasets both came from the same source, TuitionTracker, and were both found to have 3,240 rows of data on U.S. universities, but with different columns that explored the actual quoted cost of tuition (123 columns) and the cost of tuition after financial aid respectively (36 columns). This historical data spanned from 2011 to 2022, and covered several factors that might influence tuition costs such as family household income and living either on or off campus with family support. Looking specifically at the year 2020 to 2021,

Figure 2 compares the total price of tuition to the net price of tuition after scholarships and grants given the scenario that a student is either living on-campus, off-campus without family, or off-campus with family. This figure shows that commuter students, or students that live off-campus with family have a sticker price that is most aligned with the net price, while students living on-campus or off-campus without family support have a higher cost for university.

Figure 2

Histogram of Sticker Price vs. Net Price (2020-2021)



The final two datasets came from the United States Department of Agriculture and revolved around societal influences that might play a role in pursuing higher education. The PovertyEstimates dataset had 3,277 rows and 100 columns of data regarding the estimated total number of individuals living below the poverty line, and further broken down into age groups such as 0-17 year olds, families with 5-17 year olds, children under 4 years old. The median household income was also included as a comparison to those living in poverty. While the other datasets had continual historical data that was collected annually, this dataset only had information from 2013 and 2021. This is most likely due to the difficulty of acquiring and collecting such data. On the other hand, the Unemployment dataset had annual data from the year 2000 to 2022 with consideration of U.S. states.

Given that the project started off with 5 datasets, there would be an excess of information that could have introduced noise if not cleaned. Throughout the exploratory data analysis phase, each dataset was evaluated based on descriptive statistics, number of null observations, and outliers. Some datasets had similar columns as with the case of datasets coming from similar sources. Thus only specific features from each dataset were chosen to be included in a merged dataset that would be further pre-processed and cleaned. The merged dataset included geographical features related to the university, tuition information, in addition to the related region's unemployment information.

Outliers for these chosen features were identified by calculating the Z-scores. The team chose a threshold of 3 standard deviations to determine outliers. Multicollinearity can also result in skewed predictions, which in turn would fail AffordableU Foundation clients, or high school students. Thus features with a high change of multicollinearity were also removed from the merged dataset.

Data Preparation

Data Scrubbing

To determine which null values to drop from the study, the number of null values for each column within all the tables were calculated. Based on those results, there was flexibility in utilizing columns from both the Poverty and Unemployment tables due to their minimal null values, each accounting for less than 2.56%. By dropping these null values, the team effectively integrated these datasets.

However, the majority of the Net Price table consisted of null values, making it a less favorable choice for integration into the merged dataset. The team opted to exclude "in-state on-campus total price" columns from the sticker price due to the high null percentage, exceeding 40%. Likewise, 40% of columns from the University table exhibited a similar null percentage. Despite these exclusions, pertinent data relevant to the analysis remains intact within the University table.

Feature Selection

All the datasets collected for this project were found to be extremely robust, but as seen in the exploratory data analysis, there was a surplus of features that if were all used, would result in noisy data. The team calculated the correlation score of all numerical features in the merged dataset, and removed those that exceeded a correlation score of 0.80. Additionally irrelevant features such as the longitude, latitude, and ID of each university were irrelevant for analyzing tuition prices, and were also removed.

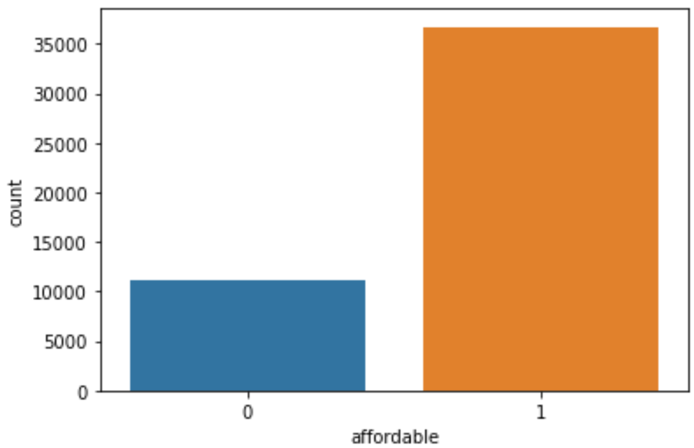
Feature Creation

Prior to the exploratory data analysis phase, a revised Poverty table was created to concatenate the area name and state. This feature engineering allowed the team to join the Poverty tables with the other tables while also avoiding the introduction of duplicate information. Additionally, the StickerPrice dataset exhibited comparable data regarding average tuition expenses as the University dataset. Consequently, StickerPrice was eliminated and will be integrated with NetPrice in an Athena DB table.

Additionally, the team wanted to determine what factors influenced the affordability of universities. Thus, the target variable used for modeling was created by calculating the national median income and identifying the minimum monthly payments an individual would be able to realistically afford given a loan term of 10 years and annual interest rate of 5.5%. This feature was applied to the tuition of each university, and resulted in either true or false that the specific university was found to be affordable. Figure 3 illustrates this calculation, and the distribution of universities identified as either affordable or unaffordable, indicating that there might be more universities that could be considered affordable based on the current market.

Figure 3

Distribution of Target Feature



Feature Transformation

The exploratory data analysis phase of the project showed that there were many categorical features that would be needed for the future model building phase. However, if not transformed into a numerical format, these features would not be able to be ingested by the machine learning models. The PredDeg feature in the University dataset categorized universities as either a primarily certificate, Associate degree, Bachelor degree, or Graduate degree granting institution. These four attributes would need to be encoded as a numeric rank from 1 to 4.

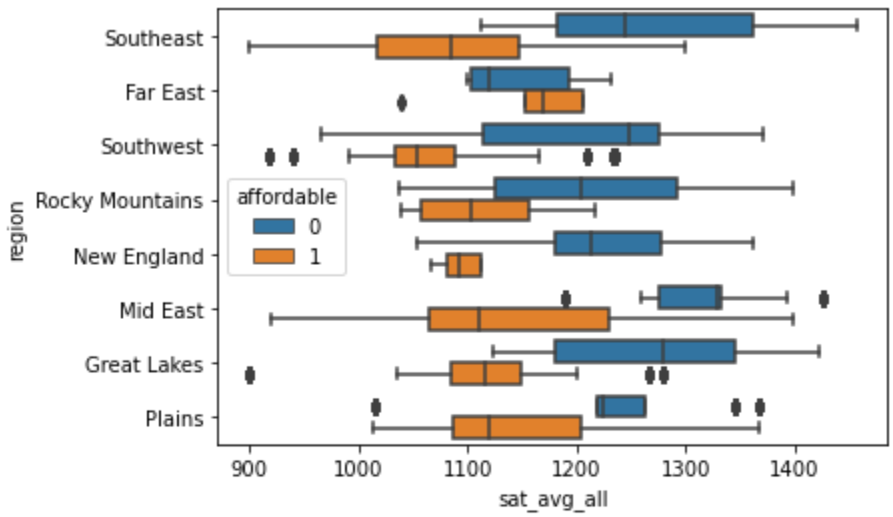
Similarly, the region of a university was a feature that would be analyzed to identify if location mattered on affordability. Since this feature was categorical but had no clear ranking or order, One Hot encoding was used to create dummy features that would depict the region in a binary format.

Additionally, the average SAT score of students was recorded in the University dataset. The College Board converted the SAT to a 2400-point scale after adding a Writing section from

2006 to 2016. This indicates that the average SAT scores from those years would need to be normalized with the rest of the data in consideration of the scale change. After normalizing both of these features, Figure 4 illustrates the relationship between U.S. regions and the average SAT scores of admitted students for universities that were identified as either unaffordable or affordable. From this feature transformation, it can be seen that unaffordable universities accepted students that generally had a higher SAT score with the exception of the Far East regions.

Figure 4

Boxplot of U.S. Regions, average SAT Scores, and Affordability



Given the volume of data, the overall dataset will be split into a 70-30 train-test split for model training and evaluation. This provides an adequate balance between maximizing the available data for training the model, while also having variance in the test set to provide a more reliable assessment of the model's performance.

Model Training

Several models were trained to determine the best fit model for the AffordableU Foundation that would accurately identify features that influence affordability. A baseline logistic regression model was created with no parameters to provide a basis of evaluation for this project due to the algorithm's simplicity. This model was run as a "bring-your-own-script" approach as no hyperparameter tuning was anticipated, and it was simple to create.

The team also worked with SageMaker built-in-algorithms to create models such as Extreme Gradient Boosting (XGBoost) and Random Forest. Both models excel with classification tasks as these are a variation of decision tree models that are highly interpretable. Sagemaker's built in XGBoost algorithm was used to train, evaluate, and test the model with several hyperparameters that would be tuned. These two predictions will be evaluated in comparison, one providing the controlled predictions and one with specifications on hyperparameters to hone in on the proper model.

Parameters

For the purpose of the SageMaker built-in algorithms, several parameters needed to be defined. First, an IAM role is defined in order to access the necessary AWS resources and a region of the instance is set using the boto3 package. This automatically retrieved the uniform resource identifier required to run the XGBoost algorithm. The Estimator object was then initialized with parameters including the XGBoost container, IAM role, instance count and type, and the output path. Additional hyperparameters are then defined for the specific XGBoost model such as max depth (5), eta (0.2), gamma (4), minimum child weight (6), subsample (0.8), silent (0), and the objective logistic regression function.

Instance Size/Count

Given the scope of this project and the given budget, we opted to use a large AWS EC2 instance that had 4 vCPUs, 16 GiB of RAM and up to 10 Gbps of bandwidth. This was priced at $0.192 per hour, which offered the best performance to cost ratio to run our model without compromising on compute time. The instance type is defined as 'ml.m5.large'. 'Ml' represents the machine learning instance for AWS Sagemaker, while 'm5' represents a general purpose workload. This project only requires the use of a single instance as seen by the independence of the 'large' parameter in the instance type.

Model Evaluation

These models were evaluated with classification metrics to compare performance between a baseline model and several tuned models. While more often than not, stakeholders find themselves concerned with the accuracy of a model, it is crucial that all aspects of the model's performance is looked at to identify false positives and false negatives. A report was created for each model to illustrate the precision, recall, f1-score, support, and accuracy of each model. In a project such as this, it was also important to introduce the most true positive instances, thus the f1-score was particularly relevant in the comparison of these models.

As seen in Table 1, the highest performing model was the random forest, followed by the baseline logistic regression, and lastly the XGBoost model. The XGBoost model did not appear to perform well upon deployment using Sagemaker and hyperparameter tuning. While the XGBoost exhibited high precision, the model did not have a strong recall, F1-score, nor accuracy score. This indicates that the model was not able to capture a large majority of the true positives. In comparison, the random forest model performed consistently well especially when compared to the baseline model. The high precision, recall, accuracy, and F1 score of the random forest model were indicative of the model's overall performance.

Table 1

Model Evaluation

Model	Precision	Recall	F1-score	Accuracy
Logistic regression	0.65	0.76	0.65	0.76
Random forest	0.95	0.95	0.95	0.95
XGBoost	0.82	0.31	0.23	0.31

Measuring Impact

If a successful model is built, there are implications that stakeholders at AffordableU and individual students would see large impacts such as:

1. Increased graduate high school seniors who enroll in fall semester at university after graduation by 15%. This increase would come from personalized recommendations for students based on their characteristics provided by the data model.
2. At least 60% of the graduating high school class would take the recommendation of the AffordableU Foundation

Measuring impact in the future can include different return on investment (ROI) measurements after this model has been implemented for a lengthy amount of time. Some of these ROI measurements of the model could include post-graduation earnings, career advancement opportunities, and overall satisfaction with their educational experience.

Security Checklist, Privacy and Other Risks:

This data project will read from a public S3 bucket, which has four folders corresponding to the data sources for the project. There is no PHI, PII, or credit card data included in the project. User behavior will not be tracked and stored.

As a foundation working with young individuals, it is always important to consider potential biases and ethical concerns both within the data and business objectives. Some data bias and risks that need to be considered is confirmation bias or false causality in which not everyone who earns a college degree will make more money/have more opportunities than someone without a college degree. Additionally, availability bias needs to be considered as everyone can come from different academic and personal backgrounds, which can make it difficult to understand how different states or regions approach education, especially biases towards higher education.

Results from this study may push students away from attending college without understanding all the benefits of having a bachelor's degree. There needs to be a focus on finding affordable schools for a range of academic budgets. Without skilled workers, the economy as a whole would suffer. Secondly, as always, it is important that private information is kept confidential. The data being used is expected to be overall averages of Universities. If there is specific information to a respective student of a University, that data is to be anonymous and confidential. Similarly, once the data objectives have been met, it is vital that assistance provided to the student by the AffordableU Foundation is confidential between student and agent.

Future Enhancements

Deploy Numerous Models

One way to enhance this project is to expand the number of machine learning models used to predict affordable universities for aspiring students. Currently, the project has incorporated logistic regression models, two XGBoost models with a different process of selecting hyperparameters, along with a random forest model. However, enhancing the predictive capabilities and robustness of the system by creating additional models would be beneficial.

This enhancement would involve the creation of additional models, each tailored to address specific nuances and factors influencing the affordability of universities across the country. These models would undergo comprehensive evaluation to assess various performance metrics, including but not limited to overfitting, accuracy, precision, recall, F1 scores, and support.

By rigorously analyzing these metrics, the effectiveness and reliability of each model will be determined to aid in predicting affordable universities. This evaluation process ensures that the models selected for deployment are not only accurate but also robust and well-suited for real-world applications. Additionally, it allows for the identification of any potential pitfalls or areas for improvement, which will facilitate continuous refinement and optimization of the predictive models.

Utilize Personalized Demographics

As the project and foundation grows, the team hopes to bring each student's personal demographic information into the model to make real-time recommendations. Tailoring the model with specific, individual student demographics involves customizing the predictive model to account for the unique characteristics and circumstances of each student. This approach recognizes that different demographic groups may have distinct patterns and preferences that influence their enrollment decisions and academic success. Incorporating demographic-specific features into the model can provide valuable insights into individual student behavior and needs. For example, features such as parental education level, household income, high school performance, or extracurriculars can be used to personalize predictions based on the socioeconomic background or academic readiness of each student.

Along with the real-time recommendations based on acceptance rates, tuition costs, etc. of the university, this expansion in resources would assist in finding the best fit for the student based on prior student experiences, personalities, and other personal characteristics. Ultimately, AffordableU Foundation's goal is to connect prospective high school seniors to universities that they will have a desire to attend and be involved in daily life activities on campus, as well as academics.

Dashboards

Presentation of the model results will also be a valuable improvement to the Foundation's success and unique ability to connect with the students. As previously mentioned, numerous models can be created with an increase in resources to provide numerous suggestions to the student based on different variables analyzed. Our goal is to present these recommended Universities to the student in a way that catches their attention but is also informative and assistive in their University selection.

With an expansion of resources, AffordableU Foundation can create dashboards personalized to each student with tools like Power BI or Tableau. These dashboards will provide an in depth picture of the top three Universities recommended by the aforementioned models. Information would include, overall tuition cost, clubs, athletics, careers, academic programs, links to on campus tours, etc. These dashboards give the student the privilege of experiencing the university in an online environment. An introduction to university resources preemptively sets the student up for success prior to applications, enrollment, and attendance.

References

Carnevale, A., Gulish, A., & Peltier-Campbell, K. (2022). If not now, when?: The urgent need for an All-One-System approach to youth policy. In *Georgetown University Center on Education and the Workforce*. https://cew.georgetown.edu/cew-reports/allonesystem/

U.S.D.A. County-level Data sets. (2023, June). https://www.ers.usda.gov/data-products/county-level-data-sets/county-level-data-sets-download-data/

D'Amato, P. (2018, April 30). Tuition Tracker Datasets. *TuitionTracker*. https://www.tuitiontracker.org/

Sonoda, R. (2024, January 22). *U.S. university dataset from 2001 to 2022*. Kaggle. https://www.kaggle.com/datasets/ryusonoda/u-s-university-dataset-from-2001-to-2022

Understanding how much student debt you can afford. (2022). Consumer Financial Protection Bureau. https://files.consumerfinance.gov/f/documents/cfpb_building_block_activities_understanding-how-much-student-debt-afford_guide.pdf

College Affordability

April 14, 2024

1 1. Prepare Datasets

1.1 1.1 Import the S3 data into SageMaker

[ ]:
```
# Import packages
import boto3
import sagemaker
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, RandomizedSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, classification_report
from scipy.stats import uniform
from sagemaker.image_uris import retrieve
from sagemaker.inputs import TrainingInput
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/root/.config/sagemaker/config.yaml
```

[ ]:
```
sess = sagemaker.Session()
bucket = sess.default_bucket()
role = sagemaker.get_execution_role()
region = boto3.Session().region_name
account_id = boto3.client("sts").get_caller_identity().get("Account")

sm = boto3.Session().client(service_name="sagemaker", region_name=region)
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/root/.config/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
```

```
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/root/.config/sagemaker/config.yaml
```

1.2 1.2 Store S3 locations

[ ]:
```
s3_public_path_kaggle = "s3://collegeaffordability317/Kaggle/"
s3_public_path_tuition = "s3://collegeaffordability317/TuitionTracker/"
s3_public_path_usda = "s3://collegeaffordability317/USDA/"
```

[ ]:
```
%store s3_public_path_kaggle
%store s3_public_path_tuition
%store s3_public_path_usda
```

```
Stored 's3_public_path_kaggle' (str)
Stored 's3_public_path_tuition' (str)
Stored 's3_public_path_usda' (str)
```

1.3 1.3 Show all the data files for the project

[ ]:
```
!aws s3 ls $s3_public_path_kaggle
```

```
2024-03-25 00:16:36   14081604 college_data.csv
```

[ ]:
```
!aws s3 ls $s3_public_path_tuition --recursive
```

```
2024-03-24 19:25:25          0 TuitionTracker/DataDictionary/
2024-03-24 19:25:37      21527 TuitionTracker/DataDictionary/DataDictionary.xlsx
2024-03-24 19:24:22          0 TuitionTracker/GradRates/
2024-03-24 19:24:45     838246 TuitionTracker/GradRates/GradRates.csv
2024-03-24 19:24:55          0 TuitionTracker/NetPrice/
2024-03-24 19:25:12    1319314 TuitionTracker/NetPrice/NetPrice.csv
2024-03-24 19:23:55          0 TuitionTracker/RetentionRates/
2024-03-24 19:24:08    1382164 TuitionTracker/RetentionRates/RetentionRates.csv
2024-03-24 19:20:50          0 TuitionTracker/StickerPrice/
2024-03-24 19:21:49     653000 TuitionTracker/StickerPrice/StickerPrice.csv
```

[ ]:
```
!aws s3 ls $s3_public_path_usda --recursive
```

```
2024-03-24 19:26:33          0 USDA/Education/
2024-03-24 19:26:58    1610608 USDA/Education/Education.xlsx
2024-03-24 19:28:03          0 USDA/PovertyEstimates/
2024-03-24 23:00:25     449799 USDA/PovertyEstimates/PovertyEstimates.csv
2024-03-25 22:15:58     187092 USDA/PovertyTableWithState/20240325_221556_00069_
795ia_cfa99626-d392-4100-8b4f-0cfe8f1c1388.gz
2024-03-25 23:21:35     187092 USDA/PovertyTableWithState/20240325_232134_00063_
r69ka_db087c21-7823-41ba-8bf6-04d122f8c50d.gz
2024-03-31 20:06:23     187092 USDA/PovertyTableWithState/20240331_200621_00076_
```

```
h5iap_709e3c51-6e6e-4838-8662-f0807fbfbc84.gz
2024-03-24 19:28:24          0 USDA/Unemployment/
2024-03-24 22:59:49    1618804 USDA/Unemployment/Unemployment.csv
```

2 2. Athena DB

2.1 2.1 Data Wrangling

[ ]:
```
from pyathena import connect

# Set S3 staging directory -- this is a temporary directory used for Athena␣
↪queries
s3_staging_dir = "s3://{0}/athena/staging".format(bucket)
```

[ ]:
```
# Set Athena parameters
college_affordability_database = 'collegeaffordability317'
table_name = 'college_data'
```

[ ]:
```
conn = connect(region_name=region, s3_staging_dir=s3_staging_dir)
```

[ ]:
```
statement = "CREATE DATABASE IF NOT EXISTS {}".
↪format(college_affordability_database)
pd.read_sql(statement, conn)
```

[ ]:
```
Empty DataFrame
Columns: []
Index: []
```

[ ]:
```
statement = "SHOW DATABASES"

df_show = pd.read_sql(statement, conn)
df_show.head(10)
```

[ ]:
```
             database_name
0  collegeaffordability317
1                  default
2                   dsoaws
3   sagemaker_featurestore
```

2.2 2.2 Populate Tables

[ ]:
```
tuition_tracker_dir = 's3://collegeaffordability317/TuitionTracker/'
usda_dir = 's3://collegeaffordability317/USDA/'
kaggle_dir = 's3://collegeaffordability317/Kaggle/'
```

[ ]:
```
# Drop the table if it already exists
university_table = 'University'
pd.read_sql(f'DROP TABLE IF EXISTS {college_affordability_database}.
 ↪{university_table}', conn)

# Define the CREATE TABLE statement with data types in lowercase
create_university_table = f"""
CREATE EXTERNAL TABLE IF NOT EXISTS {college_affordability_database}.
 ↪{university_table} (
    UNITID INT,
    INSTNM STRING,
    CITY STRING,
    STABBR STRING,
    ZIP STRING,
    REGION STRING,
    PREDDEG STRING,
    LOCALE INT,
    LATITUDE FLOAT,
    LONGITUDE FLOAT,
    CCBASIC STRING,
    CCUGPROF STRING,
    CCSIZSET STRING,
    ADM_RATE_ALL FLOAT,
    PPTUG_EF FLOAT,
    SAT_AVG_ALL INT,
    COSTT4_A INT,
    CONTROL STRING,
    TUITIONFEE_IN INT,
    TUITIONFEE_OUT INT,
    MEDIAN_HH_INC FLOAT,
    MN_EARN_WNE_INDEP0_P10 INT,
    MN_EARN_WNE_INDEP1_P10 INT,
    UGDS_WHITE FLOAT,
    UGDS_BLACK FLOAT,
    UGDS_HISP FLOAT,
    UGDS_ASIAN FLOAT,
    UGDS_AIAN FLOAT,
    UGDS_NHPI FLOAT,
    UGDS_2MOR FLOAT,
    UGDS_NRA FLOAT,
    UGDS_UNKN FLOAT,
    year INT
    )
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LOCATION '{kaggle_dir}'
    TBLPROPERTIES ('skip.header.line.count'='1')
```

```
"""

# Execute create table statement
pd.read_sql(create_university_table, conn)

pd.read_sql(f'SELECT * FROM {college_affordability_database}.{university_table}␣
↪LIMIT 5', conn)
```

```
[ ]:    unitid                               instnm        city stabbr        zip  \
     0  100654             Alabama A & M University      Normal     AL      35762
     1  100663  University of Alabama at Birmingham  Birmingham     AL 35294-0110
     2  100706  University of Alabama in Huntsville  Huntsville     AL      35899
     3  100751            The University of Alabama  Tuscaloosa     AL 35487-0100
     4  100858                    Auburn University      Auburn     AL      36849

                                         region  \
     0  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
     1  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
     2  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
     3  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
     4  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)

                                         preddeg locale latitude longitude  …  \
     0  Predominantly bachelor's-degree granting   None     None      None  …
     1  Predominantly bachelor's-degree granting   None     None      None  …
     2  Predominantly bachelor's-degree granting   None     None      None  …
     3  Predominantly bachelor's-degree granting   None     None      None  …
     4  Predominantly bachelor's-degree granting   None     None      None  …

       ugds_white ugds_black ugds_hisp  ugds_asian  ugds_aian  ugds_nhpi ugds_2mor  \
     0       None       None      None        None       None       None      None
     1       None       None      None        None       None       None      None
     2       None       None      None        None       None       None      None
     3       None       None      None        None       None       None      None
     4       None       None      None        None       None       None      None

       ugds_nra  ugds_unkn  year
     0   0.0402     0.0017  2001
     1   0.0330     0.0255  2001
     2   0.0396     0.0000  2001
     3   0.0159     0.0000  2001
     4   0.0084     0.0016  2001

     [5 rows x 33 columns]
```

```
[ ]: # Drop the table if it already exists
sticker_price_table = 'StickerPrice'
```

```
pd.read_sql(f'DROP TABLE IF EXISTS {college_affordability_database}.
↪{sticker_price_table}', conn)

# Define the CREATE TABLE statement with data types in lowercase
create_sticker_price_table = f"""
CREATE EXTERNAL TABLE IF NOT EXISTS {college_affordability_database}.
↪{sticker_price_table} (
    unit_id INT,
    institution_name STRING,
    sector INT,
    total_price_in_state_on_campus_2021_2022 FLOAT,
    total_price_in_state_off_campus_wo_fam_2021_2022 FLOAT,
    total_price_in_state_off_campus_w_fam_2021_2022 FLOAT,
    total_price_in_state_on_campus_2020_2021 FLOAT,
    total_price_in_state_off_campus_wo_fam_2020_2021 FLOAT,
    total_price_in_state_off_campus_w_fam_2020_2021 FLOAT,
    total_price_in_state_on_campus_2019_2020 FLOAT,
    total_price_in_state_off_campus_wo_fam_2019_2020 FLOAT,
    total_price_in_state_off_campus_w_fam_2019_2020 FLOAT,
    total_price_in_state_on_campus_2018_2019 FLOAT,
    total_price_in_state_off_campus_wo_fam_2018_2019 FLOAT,
    total_price_in_state_off_campus_w_fam_2018_2019 FLOAT,
    total_price_in_state_on_campus_2017_2018 FLOAT,
    total_price_in_state_off_campus_wo_fam_2017_2018 FLOAT,
    total_price_in_state_off_campus_w_fam_2017_2018 FLOAT,
    total_price_in_state_on_campus_2016_2017 FLOAT,
    total_price_in_state_off_campus_wo_fam_2016_2017 FLOAT,
    total_price_in_state_off_campus_w_fam_2016_2017 FLOAT,
    total_price_in_state_on_campus_2015_2016 FLOAT,
    total_price_in_state_off_campus_wo_fam_2015_2016 FLOAT,
    total_price_in_state_off_campus_w_fam_2015_2016 FLOAT,
    total_price_in_state_on_campus_2014_2015 FLOAT,
    total_price_in_state_off_campus_wo_fam_2014_2015 FLOAT,
    total_price_in_state_off_campus_w_fam_2014_2015 FLOAT,
    total_price_in_state_on_campus_2013_2014 FLOAT,
    total_price_in_state_off_campus_wo_fam_2013_2014 FLOAT,
    total_price_in_state_off_campus_w_fam_2013_2014 FLOAT,
    total_price_in_state_on_campus_2012_2013 FLOAT,
    total_price_in_state_off_campus_wo_fam_2012_2013 FLOAT,
    total_price_in_state_off_campus_w_fam_2012_2013 FLOAT,
    total_price_in_state_on_campus_2011_2012 FLOAT,
    total_price_in_state_off_campus_wo_fam_2011_2012 FLOAT,
    total_price_in_state_off_campus_w_fam_2011_2012 FLOAT
    )
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LOCATION '{tuition_tracker_dir}/{sticker_price_table}'
```

```
    TBLPROPERTIES ('skip.header.line.count'='1')
"""

# Execute create table statement
pd.read_sql(create_sticker_price_table, conn)

pd.read_sql(f'SELECT * FROM {college_affordability_database}.
 ↪{sticker_price_table} LIMIT 5', conn)
```

[ ]:
```
   unit_id                                    institution_name  sector  \
0   180203                               Aaniiih Nakoda College       1
1   222178                         Abilene Christian University       2
2   497037  Abilene Christian University-Undergraduate Online       2
3   138558                   Abraham Baldwin Agricultural College       1
4   488031                             Abraham Lincoln University       3

   total_price_in_state_on_campus_2021_2022  \
0                                       NaN
1                                   55500.0
2                                       NaN
3                                   15727.0
4                                       NaN

   total_price_in_state_off_campus_wo_fam_2021_2022  \
0                                           17030.0
1                                           55500.0
2                                           30670.0
3                                           13965.0
4                                           27133.0

   total_price_in_state_off_campus_w_fam_2021_2022  \
0                                           8510.0
1                                          43872.0
2                                          19042.0
3                                           7765.0
4                                          11365.0

   total_price_in_state_on_campus_2020_2021  \
0                                       NaN
1                                   53672.0
2                                       NaN
3                                   15575.0
4                                       NaN

   total_price_in_state_off_campus_wo_fam_2020_2021  \
0                                           17030.0
1                                           53672.0
```

```
2                                          NaN
3                                      13865.0
4                                      25576.0

   total_price_in_state_off_campus_w_fam_2020_2021  \
0                                           8510.0
1                                          42322.0
2                                              NaN
3                                           7665.0
4                                          11176.0

   total_price_in_state_on_campus_2019_2020  …  \
0                                       NaN  …
1                                   51887.0  …
2                                       NaN  …
3                                   15479.0  …
4                                       NaN  …

   total_price_in_state_off_campus_w_fam_2014_2015  \
0                                           8510.0
1                                          34100.0
2                                              NaN
3                                           6894.0
4                                              NaN

   total_price_in_state_on_campus_2013_2014  \
0                                       NaN
1                                   41800.0
2                                       NaN
3                                   17503.0
4                                       NaN

   total_price_in_state_off_campus_wo_fam_2013_2014  \
0                                            17030.0
1                                            41800.0
2                                                NaN
3                                            13188.0
4                                                NaN

   total_price_in_state_off_campus_w_fam_2013_2014  \
0                                           8510.0
1                                          33000.0
2                                              NaN
3                                           7578.0
4                                              NaN

   total_price_in_state_on_campus_2012_2013  \
```

```
0                                          NaN
1                                      39900.0
2                                          NaN
3                                      16550.0
4                                          NaN

   total_price_in_state_off_campus_wo_fam_2012_2013  \
0                                           17030.0
1                                           39900.0
2                                               NaN
3                                           12619.0
4                                               NaN

   total_price_in_state_off_campus_w_fam_2012_2013  \
0                                           8510.0
1                                          31250.0
2                                              NaN
3                                           7009.0
4                                              NaN

   total_price_in_state_on_campus_2011_2012  \
0                                       NaN
1                                   38250.0
2                                       NaN
3                                   12347.0
4                                       NaN

   total_price_in_state_off_campus_wo_fam_2011_2012  \
0                                           17030.0
1                                               NaN
2                                               NaN
3                                               NaN
4                                               NaN

   total_price_in_state_off_campus_w_fam_2011_2012
0                                           8510.0
1                                              NaN
2                                              NaN
3                                              NaN
4                                              NaN

[5 rows x 36 columns]
```

[ ]:
```
# Drop the table if it already exists
net_price_table = 'NetPrice'
pd.read_sql(f'DROP TABLE IF EXISTS {college_affordability_database}.
↪{net_price_table}', conn)
```

```python
# Define the CREATE TABLE statement with data types in lowercase
create_net_price_table = f"""
CREATE EXTERNAL TABLE IF NOT EXISTS {college_affordability_database}.
↪{net_price_table} (
    unit_id int,
    institution_name string,
    sector int,
    avg_net_price_grants_scholarship_2020_2021 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2020_2021 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2020_2021 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2020_2021 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2020_2021 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2020_2021 float,
    avg_net_price_grants_scholarship_2019_2020 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2019_2020 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2019_2020 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2019_2020 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2019_2020 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2019_2020 float,
    avg_net_price_grants_scholarship_2018_2019 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2018_2019 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2018_2019 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2018_2019 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2018_2019 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2018_2019 float,
    avg_net_price_grants_scholarship_2017_2018 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2017_2018 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2017_2018 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2017_2018 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2017_2018 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2017_2018 float,
    avg_net_price_grants_scholarship_2016_2017 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2016_2017 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2016_2017 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2016_2017 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2016_2017 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2016_2017 float,
    avg_net_price_grants_scholarship_2015_2016 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2015_2016 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2015_2016 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2015_2016 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2015_2016 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2015_2016 float,
    avg_net_price_grants_scholarship_2014_2015 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2014_2015 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2014_2015 float,
```

```
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2014_2015 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2014_2015 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2014_2015 float,
    avg_net_price_grants_scholarship_2013_2014 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2013_2014 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2013_2014 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2013_2014 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2013_2014 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2013_2014 float,
    avg_net_price_grants_scholarship_2012_2013 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2012_2013 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2012_2013 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2012_2013 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2012_2013 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2012_2013 float,
    avg_net_price_grants_scholarship_2011_2012 float,
    avg_net_price_income_0_30k_titleiv_fed_finaid_2011_2012 float,
    avg_net_price_income_30k_48k_titleiv_fed_finaid_2011_2012 float,
    avg_net_price_income_48k_75k_titleiv_fed_finaid_2011_2012 float,
    avg_net_price_income_75k_110k_titleiv_fed_finaid_2011_2012 float,
    avg_net_price_income_over_110k_titleiv_fed_finaid_2011_2012 float
    )
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LOCATION '{tuition_tracker_dir}/{net_price_table}'
    TBLPROPERTIES ('skip.header.line.count'='1')
"""

# Execute create table statement
pd.read_sql(create_net_price_table, conn)

pd.read_sql(f'SELECT * FROM {college_affordability_database}.{net_price_table}␣
 ↪LIMIT 5', conn)
```

[ ]:

```
   unit_id                                   institution_name  sector  \
0   180203                              Aaniiih Nakoda College       1
1   222178                        Abilene Christian University       2
2   497037  Abilene Christian University-Undergraduate Online       2
3   138558                  Abraham Baldwin Agricultural College       1
4   488031                         Abraham Lincoln University       3

   avg_net_price_grants_scholarship_2020_2021  \
0                                      8381.0
1                                         NaN
2                                         NaN
3                                      7744.0
4                                         NaN
```

```
   avg_net_price_income_0_30k_titleiv_fed_finaid_2020_2021  \
0                                             8119.0
1                                                NaN
2                                                NaN
3                                             4784.0
4                                                NaN

   avg_net_price_income_30k_48k_titleiv_fed_finaid_2020_2021  \
0                                             8326.0
1                                                NaN
2                                                NaN
3                                             5862.0
4                                                NaN

   avg_net_price_income_48k_75k_titleiv_fed_finaid_2020_2021  \
0                                            10138.0
1                                                NaN
2                                                NaN
3                                             8408.0
4                                                NaN

   avg_net_price_income_75k_110k_titleiv_fed_finaid_2020_2021  \
0                                                NaN
1                                                NaN
2                                                NaN
3                                            10953.0
4                                                NaN

   avg_net_price_income_over_110k_titleiv_fed_finaid_2020_2021  \
0                                                NaN
1                                                NaN
2                                                NaN
3                                            10568.0
4                                                NaN

   avg_net_price_grants_scholarship_2019_2020  …  \
0                                      7777.0  …
1                                         NaN  …
2                                         NaN  …
3                                      8106.0  …
4                                         NaN  …

   avg_net_price_income_30k_48k_titleiv_fed_finaid_2012_2013  \
0                                             5024.0
1                                                NaN
2                                                NaN
```

```
3                                          8862.0
4                                             NaN

   avg_net_price_income_48k_75k_titleiv_fed_finaid_2012_2013  \
0                                          3359.0
1                                             NaN
2                                             NaN
3                                         10959.0
4                                             NaN

   avg_net_price_income_75k_110k_titleiv_fed_finaid_2012_2013  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                         12342.0
4                                             NaN

   avg_net_price_income_over_110k_titleiv_fed_finaid_2012_2013  \
0                                             NaN
1                                             NaN
2                                             NaN
3                                         12946.0
4                                             NaN

   avg_net_price_grants_scholarship_2011_2012  \
0                                  13201.0
1                                      NaN
2                                      NaN
3                                   7518.0
4                                      NaN

   avg_net_price_income_0_30k_titleiv_fed_finaid_2011_2012  \
0                                         13133.0
1                                             NaN
2                                             NaN
3                                          6026.0
4                                             NaN

   avg_net_price_income_30k_48k_titleiv_fed_finaid_2011_2012  \
0                                         13769.0
1                                             NaN
2                                             NaN
3                                          6895.0
4                                             NaN

   avg_net_price_income_48k_75k_titleiv_fed_finaid_2011_2012  \
0                                         14069.0
```

```
1                                                     NaN
2                                                     NaN
3                                                  9511.0
4                                                     NaN

   avg_net_price_income_75k_110k_titleiv_fed_finaid_2011_2012  \
0                                                     NaN
1                                                     NaN
2                                                     NaN
3                                                 11080.0
4                                                     NaN

   avg_net_price_income_over_110k_titleiv_fed_finaid_2011_2012
0                                                     NaN
1                                                     NaN
2                                                     NaN
3                                                 11182.0
4                                                     NaN

[5 rows x 63 columns]
```

[ ]:
```
# Drop the table if it already exists
poverty_table = 'PovertyEstimates'
pd.read_sql(f'DROP TABLE IF EXISTS {college_affordability_database}.
 ↪{poverty_table}', conn)

# Define the CREATE TABLE statement with data types in lowercase
create_poverty_table = f"""
CREATE EXTERNAL TABLE IF NOT EXISTS {college_affordability_database}.
 ↪{poverty_table} (
    FIPS_Code INT,
    Stabr STRING,
    Area_name STRING,
    Rural_urban_Continuum_Code_2003 STRING,
    Urban_Influence_Code_2003 STRING,
    Rural_urban_Continuum_Code_2013 STRING,
    Urban_Influence_Code_2013 STRING,
    POVALL_2021 STRING,
    CI90LBALL_2021 STRING,
    CI90UBALL_2021 STRING,
    PCTPOVALL_2021 STRING,
    CI90LBALLP_2021 STRING,
    CI90UBALLP_2021 STRING,
    POV017_2021 STRING,
    CI90LB017_2021 STRING,
    CI90UB017_2021 STRING,
    PCTPOV017_2021 STRING,
```

```
    CI90LB017P_2021 STRING,
    CI90UB017P_2021 STRING,
    POV517_2021 STRING,
    CI90LB517_2021 STRING,
    CI90UB517_2021 STRING,
    PCTPOV517_2021 STRING,
    CI90LB517P_2021 STRING,
    CI90UB517P_2021 STRING,
    MEDHHINC_2021 STRING,
    CI90LBINC_2021 STRING,
    CI90UBINC_2021 STRING,
    POV04_2021 STRING,
    CI90LB04_2021 STRING,
    CI90UB04_2021 STRING,
    PCTPOV04_2021 STRING,
    CI90LB04P_2021 STRING,
    CI90UB04P_2021 STRING
    )
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LOCATION '{usda_dir}/{poverty_table}'
    TBLPROPERTIES ('skip.header.line.count'='1')
"""

# Execute create table statement
pd.read_sql(create_poverty_table, conn)

pd.read_sql(f'SELECT * FROM {college_affordability_database}.{poverty_table}␣
 ↪LIMIT 5', conn)
```

[ ]:

```
  fips_code stabr       area_name rural_urban_continuum_code_2003  \
0         0    US   United States
1      1000    AL         Alabama
2      1001    AL  Autauga County                               2
3      1003    AL  Baldwin County                               4
4      1005    AL  Barbour County                               6

  urban_influence_code_2003 rural_urban_continuum_code_2013  \
0
1
2                         2                               2
3                         5                               3
4                         6                               6

  urban_influence_code_2013 povall_2021 ci90lball_2021 ci90uball_2021  …  \
0                              41393176       41149497       41636855  …
1                                800848         782169         819527  …
```

```
2                         2       6296           4772           7820  …
3                         2      25526          21599          29453  …
4                         6       5089           3773           6405  …

  ci90ub517p_2021 medhhinc_2021 ci90lbinc_2021 ci90ubinc_2021 pov04_2021  \
0            16.3         69717          69583          69851    3349149
1            22.5         53990          53218          54762      71220
2            20.4         66444          60061          72827
3            18.5         65658          60723          70593
4            44.6         38649          34308          42990

  ci90lb04_2021 ci90ub04_2021 pctpov04_2021 ci90lb04p_2021 ci90ub04p_2021
0       3299669       3398629          18.3             18           18.6
1         66888         75552          25.1           23.6           26.6
2
3
4

[5 rows x 34 columns]
```

[ ]:
```
# Drop the table if it already exists
unemployment_table = 'Unemployment'
pd.read_sql(f'DROP TABLE IF EXISTS {college_affordability_database}.
 ↪{unemployment_table}', conn)

# Define the CREATE TABLE statement with data types in lowercase
create_unemployment_table = f"""
CREATE EXTERNAL TABLE IF NOT EXISTS {college_affordability_database}.
 ↪{unemployment_table} (
    FIPS_Code INT,
    State STRING,
    Area_Name STRING,
    Rural_Urban_Continuum_Code_2013 INT,
    Urban_Influence_Code_2013 INT,
    Metro_2013 INT,
    Civilian_labor_force_2000 INT,
    Employed_2000 INT,
    Unemployed_2000 INT,
    Unemployment_rate_2000 FLOAT,
    Civilian_labor_force_2001 INT,
    Employed_2001 INT,
    Unemployed_2001 INT,
    Unemployment_rate_2001 FLOAT,
    Civilian_labor_force_2002 INT,
    Employed_2002 INT,
    Unemployed_2002 INT,
    Unemployment_rate_2002 FLOAT,
```

```
    Civilian_labor_force_2003 INT,
    Employed_2003 INT,
    Unemployed_2003 INT,
    Unemployment_rate_2003 FLOAT,
    Civilian_labor_force_2004 INT,
    Employed_2004 INT,
    Unemployed_2004 INT,
    Unemployment_rate_2004 FLOAT,
    Civilian_labor_force_2005 INT,
    Employed_2005 INT,
    Unemployed_2005 INT,
    Unemployment_rate_2005 FLOAT,
    Civilian_labor_force_2006 INT,
    Employed_2006 INT,
    Unemployed_2006 INT,
    Unemployment_rate_2006 FLOAT,
    Civilian_labor_force_2007 INT,
    Employed_2007 INT,
    Unemployed_2007 INT,
    Unemployment_rate_2007 FLOAT,
    Civilian_labor_force_2008 INT,
    Employed_2008 INT,
    Unemployed_2008 INT,
    Unemployment_rate_2008 FLOAT,
    Civilian_labor_force_2009 INT,
    Employed_2009 INT,
    Unemployed_2009 INT,
    Unemployment_rate_2009 FLOAT,
    Civilian_labor_force_2010 INT,
    Employed_2010 INT,
    Unemployed_2010 INT,
    Unemployment_rate_2010 FLOAT,
    Civilian_labor_force_2011 INT,
    Employed_2011 INT,
    Unemployed_2011 INT,
    Unemployment_rate_2011 FLOAT,
    Civilian_labor_force_2012 INT,
    Employed_2012 INT,
    Unemployed_2012 INT,
    Unemployment_rate_2012 FLOAT,
    Civilian_labor_force_2013 INT,
    Employed_2013 INT,
    Unemployed_2013 INT,
    Unemployment_rate_2013 FLOAT,
    Civilian_labor_force_2014 INT,
    Employed_2014 INT,
    Unemployed_2014 INT,
```

```
    Unemployment_rate_2014 FLOAT,
    Civilian_labor_force_2015 INT,
    Employed_2015 INT,
    Unemployed_2015 INT,
    Unemployment_rate_2015 FLOAT,
    Civilian_labor_force_2016 INT,
    Employed_2016 INT,
    Unemployed_2016 INT,
    Unemployment_rate_2016 FLOAT,
    Civilian_labor_force_2017 INT,
    Employed_2017 INT,
    Unemployed_2017 INT,
    Unemployment_rate_2017 FLOAT,
    Civilian_labor_force_2018 INT,
    Employed_2018 INT,
    Unemployed_2018 INT,
    Unemployment_rate_2018 FLOAT,
    Civilian_labor_force_2019 INT,
    Employed_2019 INT,
    Unemployed_2019 INT,
    Unemployment_rate_2019 FLOAT,
    Civilian_labor_force_2020 INT,
    Employed_2020 INT,
    Unemployed_2020 INT,
    Unemployment_rate_2020 FLOAT,
    Civilian_labor_force_2021 INT,
    Employed_2021 INT,
    Unemployed_2021 INT,
    Unemployment_rate_2021 FLOAT,
    Civilian_labor_force_2022 INT,
    Employed_2022 INT,
    Unemployed_2022 INT,
    Unemployment_rate_2022 FLOAT,
    Median_Household_Income_2021 INT,
    Med_HH_Income_Percent_of_State_Total_2021 FLOAT
    )
    ROW FORMAT DELIMITED
    FIELDS TERMINATED BY ','
    LOCATION '{usda_dir}/{unemployment_table}'
    TBLPROPERTIES ('skip.header.line.count'='1')
"""

# Execute create table statement
pd.read_sql(create_unemployment_table, conn)

pd.read_sql(f'SELECT * FROM {college_affordability_database}.
 ↪{unemployment_table} LIMIT 5', conn)
```

```
[ ]:    fips_code state          area_name  rural_urban_continuum_code_2013  \
     0          0    US      United States                              NaN
     1       1000    AL            Alabama                              NaN
     2       1001    AL  Autauga County AL                              2.0
     3       1003    AL  Baldwin County AL                              3.0
     4       1005    AL  Barbour County AL                              6.0

        urban_influence_code_2013  metro_2013  civilian_labor_force_2000  \
     0                        NaN         NaN                  142601576
     1                        NaN         NaN                    2147173
     2                        2.0         1.0                      21861
     3                        2.0         1.0                      69979
     4                        6.0         0.0                      11449

        employed_2000  unemployed_2000  unemployment_rate_2000  …  \
     0      136904853          5696723                     4.0  …
     1        2047731            99442                     4.6  …
     2          20971              890                     4.1  …
     3          67370             2609                     3.7  …
     4          10812              637                     5.6  …

        civilian_labor_force_2021  employed_2021  unemployed_2021  \
     0                  162229903      153544980          8684923
     1                    2259349        2183330            76019
     2                      26545          25809              736
     3                      99953          97034             2919
     4                       8280           7821              459

        unemployment_rate_2021  civilian_labor_force_2022  employed_2022  \
     0                     5.4                  164781642      158766998
     1                     3.4                    2286028        2226670
     2                     2.8                      26789          26181
     3                     2.9                     102849         100432
     4                     5.5                       8241           7906

        unemployed_2022  unemployment_rate_2022  median_household_income_2021  \
     0          6014644                     3.7                         69717
     1            59358                     2.6                         53990
     2              608                     2.3                         66444
     3             2417                     2.4                         65658
     4              335                     4.1                         38649

        med_hh_income_percent_of_state_total_2021
     0                                        NaN
     1                                      100.0
     2                                      123.1
     3                                      121.6
```

```
4                                        71.6

[5 rows x 100 columns]
```

2.3 2.3 Transform Tables

[ ]:
```
# Load Athena tables into Pandas dataframes

University = pd.read_sql(f'SELECT * FROM {college_affordability_database}.
 ↪{university_table}', conn)
NetPrice = pd.read_sql(f'SELECT * FROM {college_affordability_database}.
 ↪{net_price_table}', conn)
StickerPrice = pd.read_sql(f'SELECT * FROM {college_affordability_database}.
 ↪{sticker_price_table}', conn)
Poverty = pd.read_sql(f'SELECT * FROM {college_affordability_database}.
 ↪{poverty_table}', conn)
Unemployment = pd.read_sql(f'SELECT * FROM {college_affordability_database}.
 ↪{unemployment_table}', conn)
```

[ ]:
```
# Combine the area name and state arrbriviation columns so we can join the table
Poverty['area_name_with_state'] = Poverty['area_name'] + ' ' + Poverty['stabr']
Poverty.head()
```

[ ]:
```
   fips_code stabr       area_name rural_urban_continuum_code_2003  \
0          0    US   United States
1       1000    AL         Alabama
2       1001    AL  Autauga County                               2
3       1003    AL  Baldwin County                               4
4       1005    AL  Barbour County                               6

  urban_influence_code_2003 rural_urban_continuum_code_2013  \
0
1
2                         2                               2
3                         5                               3
4                         6                               6

  urban_influence_code_2013 povall_2021 ci90lball_2021 ci90uball_2021  …  \
0                              41393176       41149497       41636855  …
1                                800848         782169         819527  …
2                         2        6296           4772           7820  …
3                         2       25526          21599          29453  …
4                         6        5089           3773           6405  …

  medhhinc_2021 ci90lbinc_2021 ci90ubinc_2021 pov04_2021 ci90lb04_2021  \
0         69717          69583          69851    3349149       3299669
```

```
1         53990          53218          54762      71220          66888
2         66444          60061          72827
3         65658          60723          70593
4         38649          34308          42990

  ci90ub04_2021 pctpov04_2021 ci90lb04p_2021 ci90ub04p_2021  \
0       3398629          18.3             18           18.6
1         75552          25.1           23.6           26.6
2
3
4

  area_name_with_state
0     United States US
1           Alabama AL
2    Autauga County AL
3    Baldwin County AL
4    Barbour County AL

[5 rows x 35 columns]
```

[ ]:

```
# Drop rows when tuitionfee_out is null
University.dropna(subset = ['tuitionfee_out'], inplace = True)

# Calculate if a university is affordable based on the national median income␣
↪level
usa_row = Unemployment[Unemployment['area_name'] == 'United States']
national_median_income = usa_row['median_household_income_2021'].values[0]

# interest rate for direct subsedized and unsubsidized federal loans
annual_interest_rate = 0.055 # 5.5% annual interest rate
monthly_interest_rate = annual_interest_rate / 12

# federal loans are on a 10 year term
loan_term_years = 10

# Total number of payments
total_payments = loan_term_years * 12

# Calculate the monthly payment
monthly_payment = University['tuitionfee_out'] * (monthly_interest_rate * (1 +␣
↪monthly_interest_rate)**total_payments) / ((1 +␣
↪monthly_interest_rate)**total_payments - 1)

# Federal government recommends student loan payments stay under 10% of total␣
↪monthly income
# Assumes two people in household with student loan debt
```

```python
University['affordable'] = monthly_payment <= national_median_income / 12 * 0.05
University['affordable'] = University['affordable'].astype(int)
```

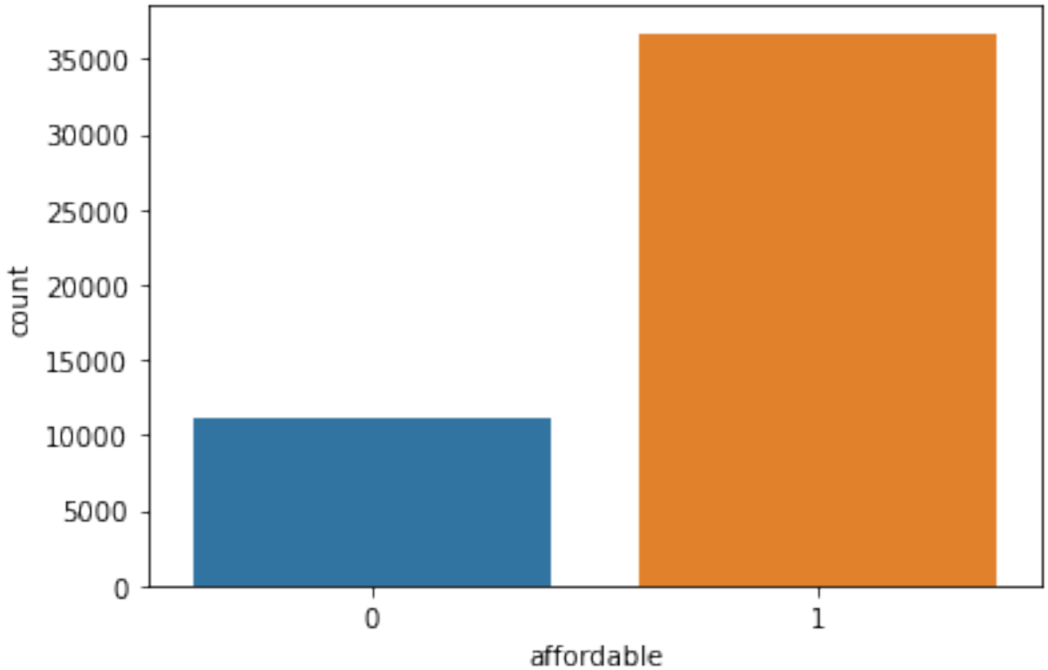
[ ]:
```python
University.head()
```

[ ]:
```
   unitid                               instnm        city stabbr         zip  \
0  100654             Alabama A & M University      Normal     AL       35762
1  100663  University of Alabama at Birmingham  Birmingham     AL  35294-0110
2  100706  University of Alabama in Huntsville  Huntsville     AL       35899
3  100751            The University of Alabama  Tuscaloosa     AL  35487-0100
4  100858                    Auburn University      Auburn     AL       36849

                                  region  \
0  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
1  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
2  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
3  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)
4  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)

                                  preddeg  locale  latitude  longitude  …  \
0  Predominantly bachelor's-degree granting     NaN       NaN        NaN  …
1  Predominantly bachelor's-degree granting     NaN       NaN        NaN  …
2  Predominantly bachelor's-degree granting     NaN       NaN        NaN  …
3  Predominantly bachelor's-degree granting     NaN       NaN        NaN  …
4  Predominantly bachelor's-degree granting     NaN       NaN        NaN  …

  ugds_black ugds_hisp ugds_asian  ugds_aian  ugds_nhpi  ugds_2mor  ugds_nra  \
0        NaN       NaN        NaN        NaN        NaN        NaN    0.0402
1        NaN       NaN        NaN        NaN        NaN        NaN    0.0330
2        NaN       NaN        NaN        NaN        NaN        NaN    0.0396
3        NaN       NaN        NaN        NaN        NaN        NaN    0.0159
4        NaN       NaN        NaN        NaN        NaN        NaN    0.0084

  ugds_unkn  year  affordable
0    0.0017  2001           1
1    0.0255  2001           1
2    0.0000  2001           1
3    0.0000  2001           1
4    0.0016  2001           1

[5 rows x 34 columns]
```

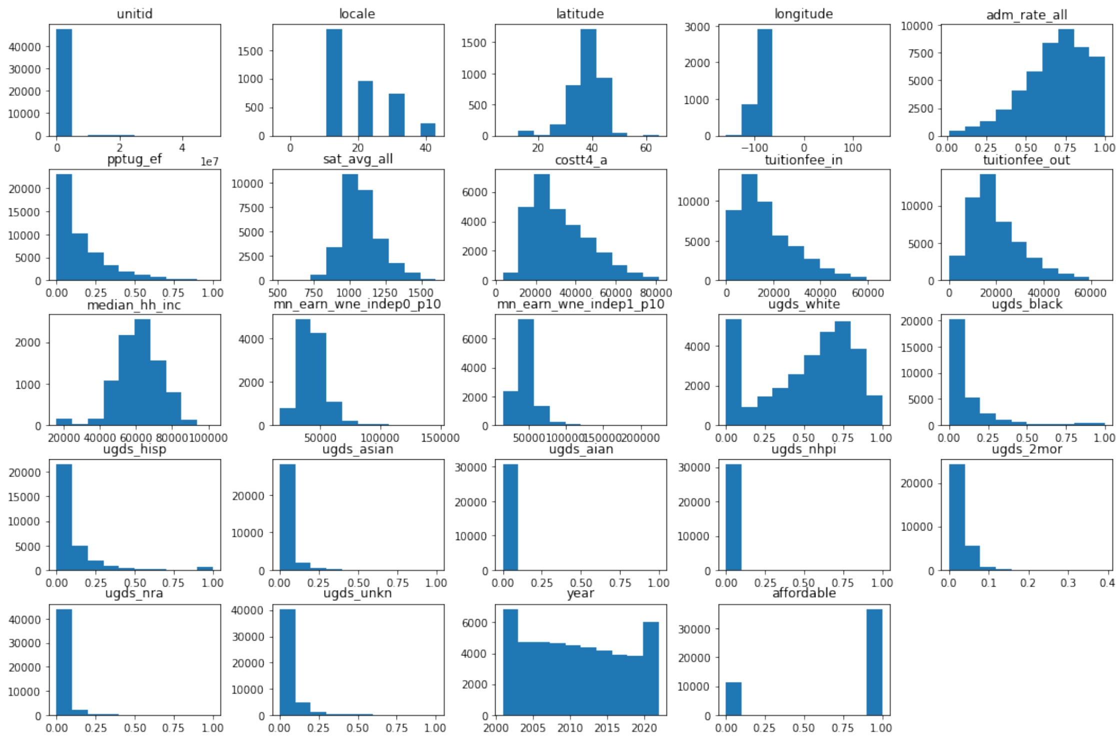
[ ]:
```python
# Show number of affordable/non-affordable universities
value_counts = University['affordable'].value_counts()
value_counts
```

```
[ ]: 1    36711
     0    11144
     Name: affordable, dtype: int64
```

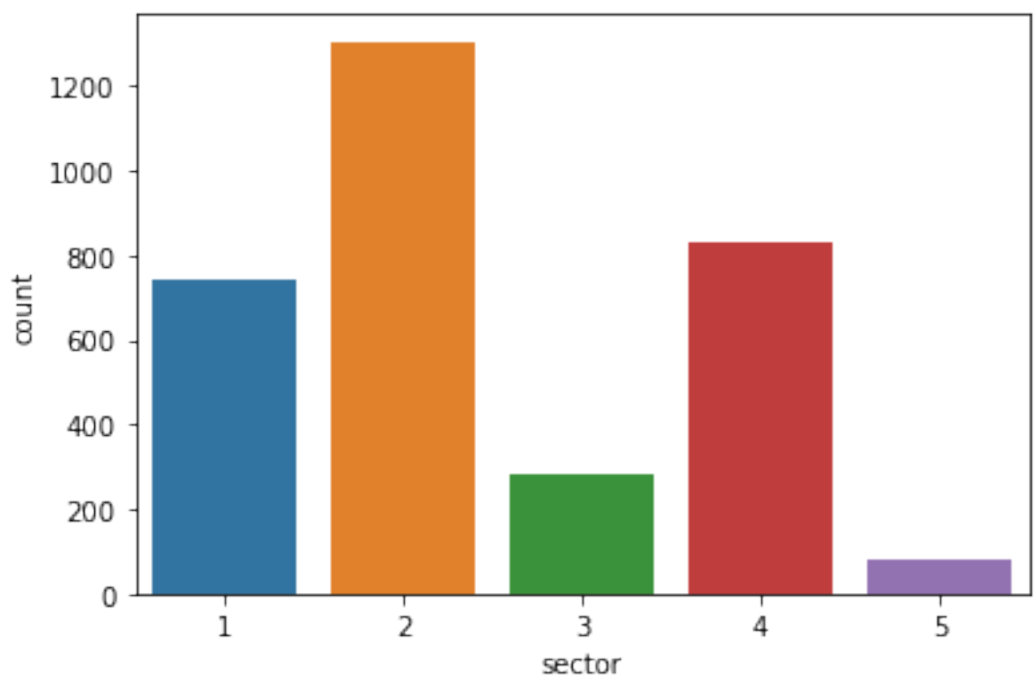
```
[ ]: sns.countplot(data=University, x='affordable')
     plt.show()
```




```
[ ]: University.hist(grid=False, figsize=(18,12))
     plt.show()
```

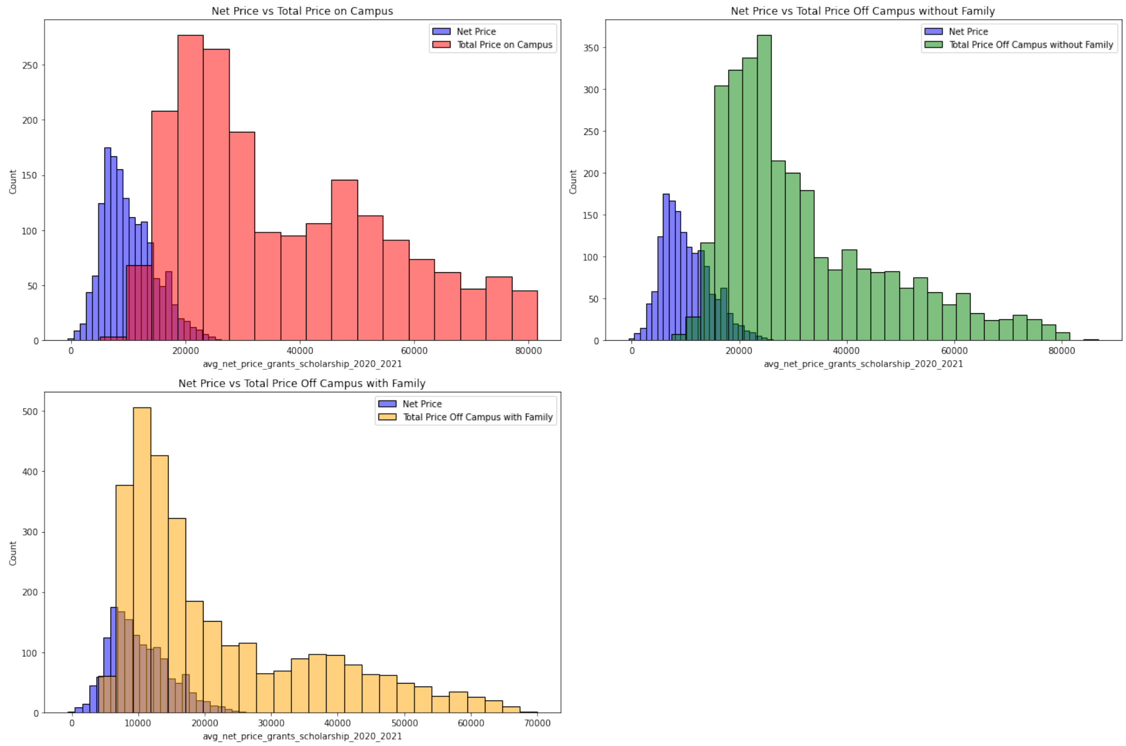


[ ]:
```
sns.countplot(data=NetPrice, x='sector')
plt.show()
```



[ ]:

```
fig, axes = plt.subplots(2, 2, figsize=(18, 12))

# Plot histograms in each subplot
sns.histplot(data=NetPrice, x='avg_net_price_grants_scholarship_2020_2021',␣
 ↪color='blue', alpha=0.5, label='Net Price', ax=axes[0, 0])
sns.histplot(data=StickerPrice, x='total_price_in_state_on_campus_2020_2021',␣
 ↪color='red', alpha=0.5, label='Total Price on Campus', ax=axes[0, 0])
axes[0, 0].legend()
axes[0, 0].set_title('Net Price vs Total Price on Campus')

sns.histplot(data=NetPrice, x='avg_net_price_grants_scholarship_2020_2021',␣
 ↪color='blue', alpha=0.5, label='Net Price', ax=axes[0, 1])
sns.histplot(data=StickerPrice,␣
 ↪x='total_price_in_state_off_campus_wo_fam_2020_2021', color='green', alpha=0.
 ↪5, label='Total Price Off Campus without Family', ax=axes[0, 1])
axes[0, 1].legend()
axes[0, 1].set_title('Net Price vs Total Price Off Campus without Family')

sns.histplot(data=NetPrice, x='avg_net_price_grants_scholarship_2020_2021',␣
 ↪color='blue', alpha=0.5, label='Net Price', ax=axes[1, 0])
sns.histplot(data=StickerPrice,␣
 ↪x='total_price_in_state_off_campus_w_fam_2020_2021', color='orange', alpha=0.
 ↪5, label='Total Price Off Campus with Family', ax=axes[1, 0])
axes[1, 0].legend()
axes[1, 0].set_title('Net Price vs Total Price Off Campus with Family')

axes[1, 1].axis('off')

plt.tight_layout()
plt.show()
```



[ ]:
```
Unemployment.head()
```

[ ]:
```
   fips_code state          area_name  rural_urban_continuum_code_2013  \
0          0    US      United States                              NaN
1       1000    AL            Alabama                              NaN
2       1001    AL  Autauga County AL                              2.0
3       1003    AL  Baldwin County AL                              3.0
4       1005    AL  Barbour County AL                              6.0

   urban_influence_code_2013  metro_2013  civilian_labor_force_2000  \
0                        NaN         NaN                142601576.0
1                        NaN         NaN                  2147173.0
2                        2.0         1.0                    21861.0
3                        2.0         1.0                    69979.0
4                        6.0         0.0                    11449.0

   employed_2000  unemployed_2000  unemployment_rate_2000  …  \
0    136904853.0        5696723.0                     4.0  …
1      2047731.0          99442.0                     4.6  …
2        20971.0            890.0                     4.1  …
3        67370.0           2609.0                     3.7  …
4        10812.0            637.0                     5.6  …
```

```
   civilian_labor_force_2021  employed_2021  unemployed_2021  \
0                162229903.0    153544980.0        8684923.0
1                  2259349.0      2183330.0          76019.0
2                    26545.0        25809.0            736.0
3                    99953.0        97034.0           2919.0
4                     8280.0         7821.0            459.0

   unemployment_rate_2021  civilian_labor_force_2022  employed_2022  \
0                     5.4                164781642.0    158766998.0
1                     3.4                  2286028.0      2226670.0
2                     2.8                    26789.0        26181.0
3                     2.9                   102849.0       100432.0
4                     5.5                     8241.0         7906.0

   unemployed_2022  unemployment_rate_2022  median_household_income_2021  \
0        6014644.0                     3.7                       69717.0
1          59358.0                     2.6                       53990.0
2            608.0                     2.3                       66444.0
3           2417.0                     2.4                       65658.0
4            335.0                     4.1                       38649.0

   med_hh_income_percent_of_state_total_2021
0                                        NaN
1                                      100.0
2                                      123.1
3                                      121.6
4                                       71.6

[5 rows x 100 columns]
```

2.4 2.1 Join Datasets

[ ]:
```
from pandasql import sqldf

# Define the SQL query
combine_tables_query = """
SELECT DISTINCT
        uni.UNITID,
        uni.INSTNM,
        uni.STABBR,
        uni.CITY,
        uni.REGION,
        uni.ADM_RATE_ALL,
        uni.SAT_AVG_ALL,
        uni.COSTT4_A,
        uni.TUITIONFEE_IN,
        uni.TUITIONFEE_OUT,
```

```
        uni.MN_EARN_WNE_INDEP1_P10,
        uni.PREDDEG,
        uni.CONTROL,
        pov.Stabr AS state_abbr,
        pov.Area_name_with_state,
        pov.POVALL_2021,
        pov.CI90LBALL_2021,
        pov.CI90UBALL_2021,
        pov.MEDHHINC_2021,
        pov.CI90LBINC_2021,
        pov.CI90UBINC_2021,
        unemp.Unemployment_rate_2000,
        unemp.Unemployment_rate_2001,
        unemp.Unemployment_rate_2002,
        unemp.Unemployment_rate_2003,
        unemp.Unemployment_rate_2004,
        unemp.Unemployment_rate_2005,
        unemp.Unemployment_rate_2006,
        unemp.Unemployment_rate_2007,
        unemp.Unemployment_rate_2008,
        unemp.Unemployment_rate_2009,
        unemp.Unemployment_rate_2010,
        unemp.Unemployment_rate_2011,
        unemp.Unemployment_rate_2013,
        unemp.Unemployment_rate_2014,
        unemp.Unemployment_rate_2015,
        unemp.Unemployment_rate_2016,
        unemp.Unemployment_rate_2017,
        unemp.Unemployment_rate_2018,
        unemp.Unemployment_rate_2019,
        unemp.Unemployment_rate_2020,
        unemp.Unemployment_rate_2021,
        unemp.Unemployment_rate_2022,
        unemp.Median_Household_Income_2021,
        net.avg_net_price_grants_scholarship_2020_2021,
        net.avg_net_price_grants_scholarship_2019_2020,
        net.avg_net_price_grants_scholarship_2018_2019,
        net.avg_net_price_grants_scholarship_2017_2018,
        net.avg_net_price_grants_scholarship_2016_2017,
        net.avg_net_price_grants_scholarship_2015_2016,
        net.avg_net_price_grants_scholarship_2014_2015,
        net.avg_net_price_grants_scholarship_2013_2014,
        net.avg_net_price_grants_scholarship_2012_2013,
        net.avg_net_price_grants_scholarship_2011_2012,
        unemp.Unemployment_rate_2012,
        uni.affordable
FROM University uni
```

```
JOIN Poverty pov ON uni.STABBR = pov.Stabr
JOIN Unemployment unemp ON pov.FIPS_Code = unemp.FIPS_Code
JOIN NetPrice net ON uni.INSTNM = net.institution_name
WHERE uni.year = 2022
"""

# Execute the SQL query on the Pandas DataFrames
df = sqldf(combine_tables_query, globals())

# Display the result
df.head()
```

[ ]:

```
   unitid                     instnm stabbr    city  \
0  100654  Alabama A & M University     AL  Normal
1  100654  Alabama A & M University     AL  Normal
2  100654  Alabama A & M University     AL  Normal
3  100654  Alabama A & M University     AL  Normal
4  100654  Alabama A & M University     AL  Normal

                                         region  adm_rate_all  sat_avg_all  \
0  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)      0.716006        954.0
1  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)      0.716006        954.0
2  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)      0.716006        954.0
3  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)      0.716006        954.0
4  Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)      0.716006        954.0

   costt4_a  tuitionfee_in  tuitionfee_out  …  \
0   21924.0        10024.0         18634.0  …
1   21924.0        10024.0         18634.0  …
2   21924.0        10024.0         18634.0  …
3   21924.0        10024.0         18634.0  …
4   21924.0        10024.0         18634.0  …

   avg_net_price_grants_scholarship_2018_2019  \
0                                     14604.0
1                                     14604.0
2                                     14604.0
3                                     14604.0
4                                     14604.0

  avg_net_price_grants_scholarship_2017_2018  \
0                                    13956.0
1                                    13956.0
2                                    13956.0
3                                    13956.0
4                                    13956.0
```

```
  avg_net_price_grants_scholarship_2016_2017  \
0                                    15812.0
1                                    15812.0
2                                    15812.0
3                                    15812.0
4                                    15812.0

  avg_net_price_grants_scholarship_2015_2016  \
0                                    15547.0
1                                    15547.0
2                                    15547.0
3                                    15547.0
4                                    15547.0

  avg_net_price_grants_scholarship_2014_2015  \
0                                    13203.0
1                                    13203.0
2                                    13203.0
3                                    13203.0
4                                    13203.0

  avg_net_price_grants_scholarship_2013_2014  \
0                                    14746.0
1                                    14746.0
2                                    14746.0
3                                    14746.0
4                                    14746.0

  avg_net_price_grants_scholarship_2012_2013  \
0                                    12887.0
1                                    12887.0
2                                    12887.0
3                                    12887.0
4                                    12887.0

  avg_net_price_grants_scholarship_2011_2012 unemployment_rate_2012 affordable
0                                    11108.0                    8.2          1
1                                    11108.0                    7.1          1
2                                    11108.0                    7.7          1
3                                    11108.0                   11.8          1
4                                    11108.0                    8.8          1

[5 rows x 56 columns]
```

3 3. Exploratory Data Analysis

[ ]:
```
# get number of rows and columns
print('Number of Rows in University - ', University.shape[0])
print('Number of Columns in University - ', University.shape[1], '\n')

print('Number of Rows in NetPrice - ', NetPrice.shape[0])
print('Number of Columns in NetPrice - ', NetPrice.shape[1], '\n')

print('Number of Rows in Poverty - ', Poverty.shape[0])
print('Number of Columns in Poverty - ', Poverty.shape[1], '\n')

print('Number of Rows in Unemployment - ', Unemployment.shape[0])
print('Number of Columns in Unemployment - ', Unemployment.shape[1], '\n')
```

```
Number of Rows in University -  47855
Number of Columns in University -  34

Number of Rows in NetPrice -  3240
Number of Columns in NetPrice -  63

Number of Rows in Poverty -  3195
Number of Columns in Poverty -  35

Number of Rows in Unemployment -  3277
Number of Columns in Unemployment -  100
```

3.1 3.1 Data Types and NULLS

[ ]:
```
# get number of rows and columns
print('Number of Rows - ', df.shape[0])
print('Number of Columns - ', df.shape[1], '\n')

# inspect datatypes and nulls
data_types = df.dtypes
data_types = pd.DataFrame(data_types)
data_types = data_types.assign(Null_Values =
                               df.isnull().sum())
data_types.reset_index(inplace = True)
data_types.rename(columns={0:'Data Type',
                          'index': 'Field',
                          'Null_Values': "Nulls"})
```

```
Number of Rows -  137872
Number of Columns -  56
```

[ ]:

	Field	Data Type	Nulls
0	unitid	int64	0
1	instnm	object	0
2	stabbr	object	0
3	city	object	0
4	region	object	0
5	adm_rate_all	float64	0
6	sat_avg_all	float64	48619
7	costt4_a	float64	4001
8	tuitionfee_in	float64	0
9	tuitionfee_out	float64	0
10	mn_earn_wne_indep1_p10	float64	36033
11	preddeg	object	0
12	control	object	0
13	state_abbr	object	0
14	area_name_with_state	object	0
15	povall_2021	object	0
16	ci90lball_2021	object	0
17	ci90uball_2021	object	0
18	medhhinc_2021	object	0
19	ci90lbinc_2021	object	0
20	ci90ubinc_2021	object	0
21	unemployment_rate_2000	float64	28
22	unemployment_rate_2001	float64	28
23	unemployment_rate_2002	float64	28
24	unemployment_rate_2003	float64	28
25	unemployment_rate_2004	float64	28
26	unemployment_rate_2005	float64	196
27	unemployment_rate_2006	float64	196
28	unemployment_rate_2007	float64	28
29	unemployment_rate_2008	float64	28
30	unemployment_rate_2009	float64	28
31	unemployment_rate_2010	float64	8
32	unemployment_rate_2011	float64	8
33	unemployment_rate_2013	float64	8
34	unemployment_rate_2014	float64	8
35	unemployment_rate_2015	float64	8
36	unemployment_rate_2016	float64	8
37	unemployment_rate_2017	float64	8
38	unemployment_rate_2018	float64	8
39	unemployment_rate_2019	float64	8
40	unemployment_rate_2020	float64	0
41	unemployment_rate_2021	float64	0
42	unemployment_rate_2022	float64	0
43	median_household_income_2021	float64	0
44	avg_net_price_grants_scholarship_2020_2021	float64	92267
45	avg_net_price_grants_scholarship_2019_2020	float64	92267

```
46  avg_net_price_grants_scholarship_2018_2019   float64  92562
47  avg_net_price_grants_scholarship_2017_2018   float64  92562
48  avg_net_price_grants_scholarship_2016_2017   float64  92562
49  avg_net_price_grants_scholarship_2015_2016   float64  92919
50  avg_net_price_grants_scholarship_2014_2015   float64  92956
51  avg_net_price_grants_scholarship_2013_2014   float64  93610
52  avg_net_price_grants_scholarship_2012_2013   float64  93966
53  avg_net_price_grants_scholarship_2011_2012   float64  94446
54                      unemployment_rate_2012   float64      8
55                                  affordable     int64      0
```

3.2 3.2 Summary Statistics

[ ]:
```
#summary statistics
stats = pd.DataFrame(df.describe()).T
stats
```

[ ]:
```
                                            count           mean  \
unitid                                   137872.0  215348.687435
adm_rate_all                             137872.0       0.733948
sat_avg_all                               89253.0    1151.858660
costt4_a                                 133871.0   36742.799083
tuitionfee_in                            137872.0   23755.840555
tuitionfee_out                           137872.0   27619.961624
mn_earn_wne_indep1_p10                   101839.0   50801.502371
unemployment_rate_2000                   137844.0       4.396701
unemployment_rate_2001                   137844.0       5.086958
unemployment_rate_2002                   137844.0       5.928520
unemployment_rate_2003                   137844.0       6.181507
unemployment_rate_2004                   137844.0       5.852187
unemployment_rate_2005                   137676.0       5.534203
unemployment_rate_2006                   137676.0       5.077598
unemployment_rate_2007                   137844.0       5.042835
unemployment_rate_2008                   137844.0       6.101547
unemployment_rate_2009                   137844.0       9.488022
unemployment_rate_2010                   137864.0       9.768224
unemployment_rate_2011                   137864.0       9.078855
unemployment_rate_2013                   137864.0       7.650666
unemployment_rate_2014                   137864.0       6.384567
unemployment_rate_2015                   137864.0       5.635530
unemployment_rate_2016                   137864.0       5.361369
unemployment_rate_2017                   137864.0       4.746569
unemployment_rate_2018                   137864.0       4.199532
unemployment_rate_2019                   137864.0       3.996497
unemployment_rate_2020                   137872.0       7.208137
unemployment_rate_2021                   137872.0       4.964397
unemployment_rate_2022                   137872.0       3.758887
```

```
median_household_income_2021                137872.0   60422.055421
avg_net_price_grants_scholarship_2020_2021   45605.0   13594.144392
avg_net_price_grants_scholarship_2019_2020   45605.0   13609.008990
avg_net_price_grants_scholarship_2018_2019   45310.0   13320.887641
avg_net_price_grants_scholarship_2017_2018   45310.0   13175.989031
avg_net_price_grants_scholarship_2016_2017   45310.0   13104.512094
avg_net_price_grants_scholarship_2015_2016   44953.0   12762.443441
avg_net_price_grants_scholarship_2014_2015   44916.0   12554.149323
avg_net_price_grants_scholarship_2013_2014   44262.0   12100.449415
avg_net_price_grants_scholarship_2012_2013   43906.0   12024.602993
avg_net_price_grants_scholarship_2011_2012   43426.0   11692.709943
unemployment_rate_2012                      137864.0       8.201889
affordable                                  137872.0       0.528983

                                                     std            min  \
unitid                                      93979.963627  100654.000000
adm_rate_all                                    0.206004       0.039152
sat_avg_all                                   129.362088     776.000000
costt4_a                                    16320.646884    8118.000000
tuitionfee_in                               15257.251778     480.000000
tuitionfee_out                              13156.366922     480.000000
mn_earn_wne_indep1_p10                      15799.219146   23400.000000
unemployment_rate_2000                          1.598401       1.300000
unemployment_rate_2001                          1.707137       1.600000
unemployment_rate_2002                          1.811223       1.600000
unemployment_rate_2003                          1.866708       1.900000
unemployment_rate_2004                          1.734494       1.600000
unemployment_rate_2005                          1.702856       2.000000
unemployment_rate_2006                          1.613475       1.600000
unemployment_rate_2007                          1.621889       1.400000
unemployment_rate_2008                          1.999010       1.300000
unemployment_rate_2009                          3.031887       2.000000
unemployment_rate_2010                          2.918881       2.000000
unemployment_rate_2011                          2.776444       1.400000
unemployment_rate_2013                          2.388310       1.200000
unemployment_rate_2014                          2.050799       1.200000
unemployment_rate_2015                          1.813046       1.800000
unemployment_rate_2016                          1.746241       1.600000
unemployment_rate_2017                          1.511904       1.500000
unemployment_rate_2018                          1.348143       1.200000
unemployment_rate_2019                          1.330921       0.800000
unemployment_rate_2020                          2.207568       1.600000
unemployment_rate_2021                          1.689318       0.900000
unemployment_rate_2022                          1.188747       0.600000
median_household_income_2021                15836.881547   25653.000000
avg_net_price_grants_scholarship_2020_2021   4090.581038    2695.000000
avg_net_price_grants_scholarship_2019_2020   4111.857018    2958.000000
```

avg_net_price_grants_scholarship_2018_2019	4249.506289	2465.000000
avg_net_price_grants_scholarship_2017_2018	4177.775661	2420.000000
avg_net_price_grants_scholarship_2016_2017	4023.185233	2304.000000
avg_net_price_grants_scholarship_2015_2016	3675.339126	3156.000000
avg_net_price_grants_scholarship_2014_2015	3786.272570	2345.000000
avg_net_price_grants_scholarship_2013_2014	3494.014386	1640.000000
avg_net_price_grants_scholarship_2012_2013	3325.762179	1993.000000
avg_net_price_grants_scholarship_2011_2012	3422.225105	1323.000000
unemployment_rate_2012	2.614596	1.100000
affordable	0.499161	0.000000

	25%	50%	\
unitid	155812.000000	198899.000000	
adm_rate_all	0.626667	0.776997	
sat_avg_all	1064.000000	1127.000000	
costt4_a	23078.000000	33560.000000	
tuitionfee_in	10044.000000	19469.000000	
tuitionfee_out	17488.000000	25570.000000	
mn_earn_wne_indep1_p10	41200.000000	48300.000000	
unemployment_rate_2000	3.400000	4.100000	
unemployment_rate_2001	4.000000	4.800000	
unemployment_rate_2002	4.800000	5.700000	
unemployment_rate_2003	4.900000	5.900000	
unemployment_rate_2004	4.700000	5.600000	
unemployment_rate_2005	4.400000	5.300000	
unemployment_rate_2006	4.100000	4.900000	
unemployment_rate_2007	4.000000	4.800000	
unemployment_rate_2008	4.800000	5.900000	
unemployment_rate_2009	7.400000	9.100000	
unemployment_rate_2010	7.800000	9.500000	
unemployment_rate_2011	7.300000	8.800000	
unemployment_rate_2013	6.100000	7.500000	
unemployment_rate_2014	5.000000	6.200000	
unemployment_rate_2015	4.400000	5.400000	
unemployment_rate_2016	4.300000	5.100000	
unemployment_rate_2017	3.700000	4.500000	
unemployment_rate_2018	3.300000	4.000000	
unemployment_rate_2019	3.100000	3.800000	
unemployment_rate_2020	5.700000	7.100000	
unemployment_rate_2021	3.900000	4.700000	
unemployment_rate_2022	3.000000	3.600000	
median_household_income_2021	50109.000000	57584.000000	
avg_net_price_grants_scholarship_2020_2021	11076.000000	13639.000000	
avg_net_price_grants_scholarship_2019_2020	11175.000000	13481.000000	
avg_net_price_grants_scholarship_2018_2019	10455.000000	13212.000000	
avg_net_price_grants_scholarship_2017_2018	10491.000000	12913.000000	
avg_net_price_grants_scholarship_2016_2017	10895.000000	13017.000000	

avg_net_price_grants_scholarship_2015_2016	10330.000000	12647.000000
avg_net_price_grants_scholarship_2014_2015	10166.000000	12550.000000
avg_net_price_grants_scholarship_2013_2014	10003.000000	12046.000000
avg_net_price_grants_scholarship_2012_2013	9971.000000	11873.000000
avg_net_price_grants_scholarship_2011_2012	9632.000000	11792.000000
unemployment_rate_2012	6.500000	8.000000
affordable	0.000000	1.000000

	75%	max
unitid	227331.000000	497268.0
adm_rate_all	0.890354	1.0
sat_avg_all	1228.000000	1537.0
costt4_a	47588.000000	81531.0
tuitionfee_in	34481.000000	66064.0
tuitionfee_out	35500.000000	66064.0
mn_earn_wne_indep1_p10	55700.000000	224600.0
unemployment_rate_2000	5.100000	17.3
unemployment_rate_2001	5.800000	17.6
unemployment_rate_2002	6.800000	19.6
unemployment_rate_2003	7.100000	17.8
unemployment_rate_2004	6.600000	20.2
unemployment_rate_2005	6.300000	21.0
unemployment_rate_2006	5.800000	20.7
unemployment_rate_2007	5.700000	20.2
unemployment_rate_2008	7.100000	22.6
unemployment_rate_2009	11.200000	28.3
unemployment_rate_2010	11.500000	29.4
unemployment_rate_2011	10.500000	29.3
unemployment_rate_2013	9.000000	25.6
unemployment_rate_2014	7.500000	24.3
unemployment_rate_2015	6.500000	24.6
unemployment_rate_2016	6.100000	24.2
unemployment_rate_2017	5.500000	19.7
unemployment_rate_2018	4.900000	18.8
unemployment_rate_2019	4.600000	20.7
unemployment_rate_2020	8.400000	22.6
unemployment_rate_2021	5.800000	19.5
unemployment_rate_2022	4.300000	14.7
median_household_income_2021	66601.000000	153716.0
avg_net_price_grants_scholarship_2020_2021	16433.000000	26179.0
avg_net_price_grants_scholarship_2019_2020	16166.000000	27675.0
avg_net_price_grants_scholarship_2018_2019	15886.000000	40090.0
avg_net_price_grants_scholarship_2017_2018	15636.000000	44661.0
avg_net_price_grants_scholarship_2016_2017	15664.000000	40927.0
avg_net_price_grants_scholarship_2015_2016	15235.000000	25097.0
avg_net_price_grants_scholarship_2014_2015	14625.000000	41647.0
avg_net_price_grants_scholarship_2013_2014	14418.000000	25138.0

```
avg_net_price_grants_scholarship_2012_2013   14299.000000   24247.0
avg_net_price_grants_scholarship_2011_2012   13890.000000   24674.0
unemployment_rate_2012                           9.600000      27.7
affordable                                       1.000000       1.0
```

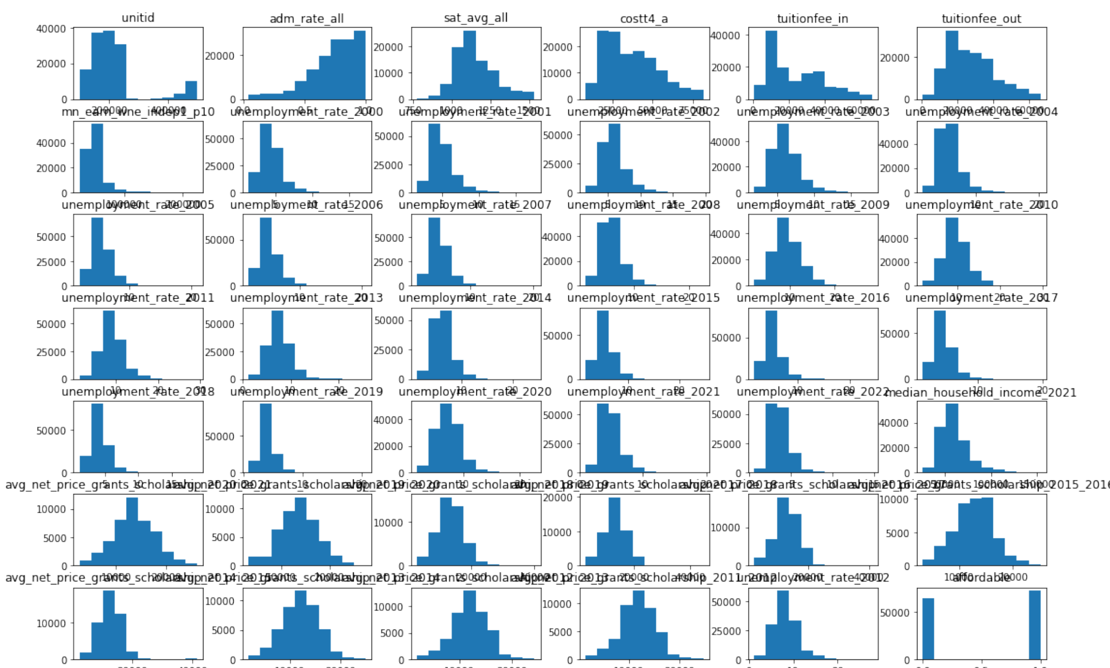
Outliers with Z-score (Threshold 3)

```
[ ]: # In-state tuition data
tuitionfee_in_data = df['tuitionfee_in']

# Z-scores
z_scores_in = (tuitionfee_in_data - np.mean(tuitionfee_in_data)) / np.
 ↪std(tuitionfee_in_data)
threshold = 3

# Find indices of outliers
outliers_in = np.where(np.abs(z_scores_in) > threshold)[0]

# Detail information about outliers
outlier_details = df.iloc[outliers_in]

# Print information about outliers
print("Outliers in tuitionfee_in:")
print(outlier_details)

# Summary statistics
print("\nSummary Statistics:")
print("Mean Tuition Fee:", np.mean(tuitionfee_in_data))
print("Median Tuition Fee:", np.median(tuitionfee_in_data))
print("Standard Deviation:", np.std(tuitionfee_in_data))
print("Number of Outliers:", len(outliers_in))
```

```
Outliers in tuitionfee_in:
Empty DataFrame
Columns: [unitid, instnm, stabbr, city, region, adm_rate_all, sat_avg_all,
costt4_a, tuitionfee_in, tuitionfee_out, mn_earn_wne_indep1_p10, preddeg,
control, state_abbr, area_name_with_state, povall_2021, ci90lball_2021,
ci90uball_2021, medhhinc_2021, ci90lbinc_2021, ci90ubinc_2021,
unemployment_rate_2000, unemployment_rate_2001, unemployment_rate_2002,
unemployment_rate_2003, unemployment_rate_2004, unemployment_rate_2005,
unemployment_rate_2006, unemployment_rate_2007, unemployment_rate_2008,
unemployment_rate_2009, unemployment_rate_2010, unemployment_rate_2011,
unemployment_rate_2013, unemployment_rate_2014, unemployment_rate_2015,
unemployment_rate_2016, unemployment_rate_2017, unemployment_rate_2018,
unemployment_rate_2019, unemployment_rate_2020, unemployment_rate_2021,
unemployment_rate_2022, median_household_income_2021,
avg_net_price_grants_scholarship_2020_2021,
avg_net_price_grants_scholarship_2019_2020,
```

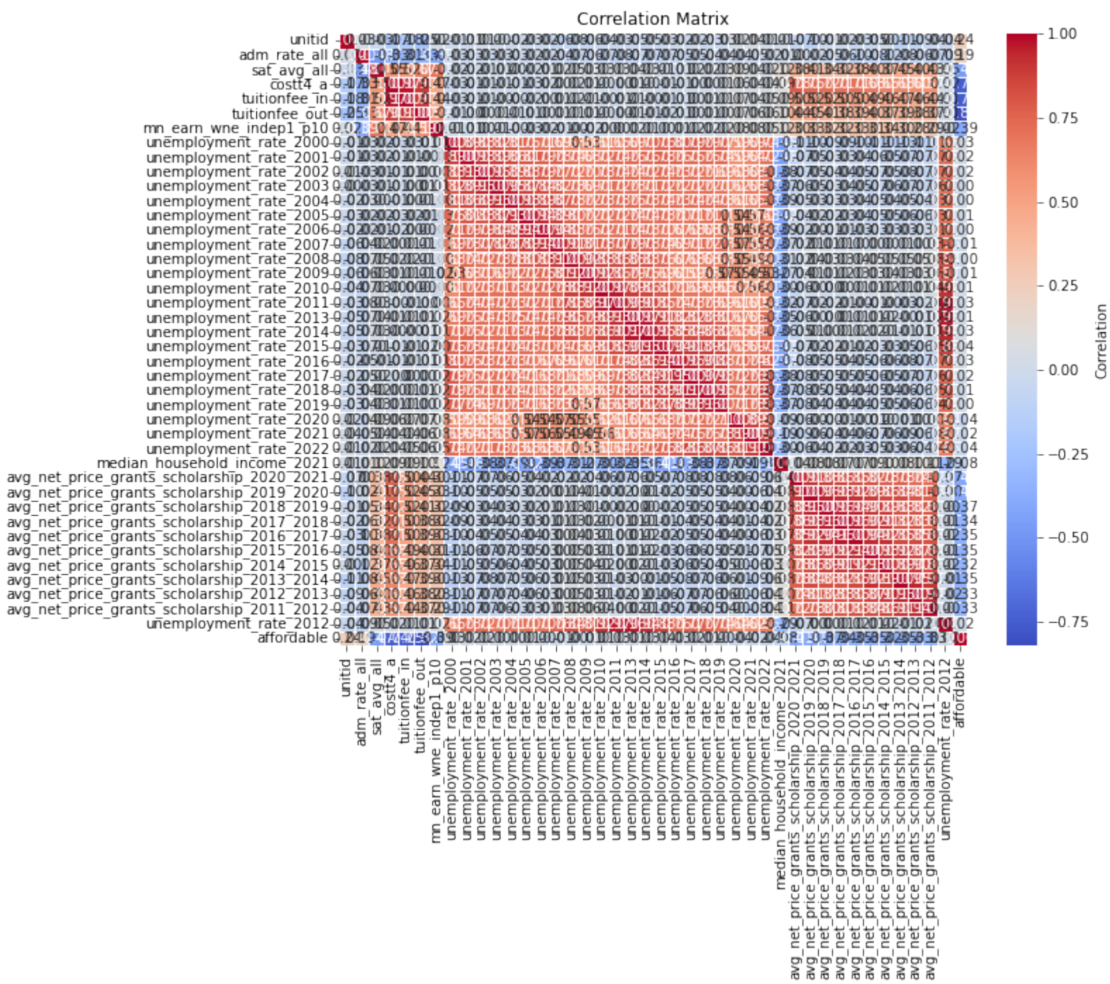
```
avg_net_price_grants_scholarship_2018_2019,
avg_net_price_grants_scholarship_2017_2018,
avg_net_price_grants_scholarship_2016_2017,
avg_net_price_grants_scholarship_2015_2016,
avg_net_price_grants_scholarship_2014_2015,
avg_net_price_grants_scholarship_2013_2014,
avg_net_price_grants_scholarship_2012_2013,
avg_net_price_grants_scholarship_2011_2012, unemployment_rate_2012, affordable]
Index: []

[0 rows x 56 columns]

Summary Statistics:
Mean Tuition Fee: 23755.840555007544
Median Tuition Fee: 19469.0
Standard Deviation: 15257.196447110557
Number of Outliers: 0
```

[ ]:
```python
df.hist(grid=False, figsize=(18,12))
plt.show()
```



3.3 Multicollinearity Test

[ ]:
```python
# assign correlation function to new variable
corr = df.corr()
corr

# Create a heatmap using seaborn
plt.figure(figsize=(10, 8))
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5, mask=corr.isnull(), cbar_kws={'label': 'Correlation'})
plt.title('Correlation Matrix')
plt.show()
```



[ ]:
```python
# This is not representative of what we will drop yet. Of course in-state and out-of-state tutition of correlated, but will not be dropped.
# Included it in case we find something interesting on joined table
```

```
[ ]: # Calculate the correlation of affordability with all other attributes
correlation_with_target = df.corrwith(df['affordable'])

# Define correlation thresholds
upper_threshold = 0.4
lower_threshold = -0.4

# Get attributes above and below thresholds
above_threshold = correlation_with_target[correlation_with_target.abs() >␣
 ↪upper_threshold]
below_threshold = correlation_with_target[correlation_with_target.abs() <␣
 ↪lower_threshold]

# Show most correlated attributes
print(above_threshold)
print(below_threshold)
```

```
sat_avg_all                                 -0.471855
costt4_a                                    -0.743210
tuitionfee_in                               -0.751890
tuitionfee_out                              -0.821012
avg_net_price_grants_scholarship_2020_2021   -0.408468
avg_net_price_grants_scholarship_2019_2020   -0.407662
affordable                                   1.000000
dtype: float64
Series([], dtype: float64)
```

```
[ ]: cor_matrix = df.corr().abs()
upper_tri = cor_matrix.where(np.triu(np.ones(cor_matrix.shape),
                                     k=1).astype(bool))

to_drop = [column for column in upper_tri.columns if
           any(upper_tri[column] > 0.80)]
print('Columns with chance of multicollinearity: %s'%to_drop)
```

```
Columns with chance of multicollinearity: ['tuitionfee_in', 'tuitionfee_out',
'unemployment_rate_2001', 'unemployment_rate_2002', 'unemployment_rate_2003',
'unemployment_rate_2004', 'unemployment_rate_2005', 'unemployment_rate_2006',
'unemployment_rate_2007', 'unemployment_rate_2008', 'unemployment_rate_2009',
'unemployment_rate_2010', 'unemployment_rate_2011', 'unemployment_rate_2013',
'unemployment_rate_2014', 'unemployment_rate_2015', 'unemployment_rate_2016',
'unemployment_rate_2017', 'unemployment_rate_2018', 'unemployment_rate_2019',
'unemployment_rate_2021', 'unemployment_rate_2022',
'avg_net_price_grants_scholarship_2019_2020',
'avg_net_price_grants_scholarship_2018_2019',
'avg_net_price_grants_scholarship_2017_2018',
'avg_net_price_grants_scholarship_2016_2017',
```

```
'avg_net_price_grants_scholarship_2015_2016',
'avg_net_price_grants_scholarship_2014_2015',
'avg_net_price_grants_scholarship_2013_2014',
'avg_net_price_grants_scholarship_2012_2013',
'avg_net_price_grants_scholarship_2011_2012', 'unemployment_rate_2012',
'affordable']
```

4 4. Data Preparation

4.0.1 Handle Duplicate, Null, and Multicollinearity Features

[ ]:
```python
# Drop redundant rows
df = df.drop_duplicates()
```

[ ]:
```python
# Get null values for each column in the university dataset
null_cols = df.isnull().sum()
total_rows = df['instnm'].count()
percent_col_null = null_cols/total_rows * 100
percent_col_null.sort_values(ascending=False)
```

[ ]:
```
avg_net_price_grants_scholarship_2011_2012    68.502669
avg_net_price_grants_scholarship_2012_2013    68.154520
avg_net_price_grants_scholarship_2013_2014    67.896310
avg_net_price_grants_scholarship_2014_2015    67.421957
avg_net_price_grants_scholarship_2015_2016    67.395120
avg_net_price_grants_scholarship_2016_2017    67.136184
avg_net_price_grants_scholarship_2017_2018    67.136184
avg_net_price_grants_scholarship_2018_2019    67.136184
avg_net_price_grants_scholarship_2019_2020    66.922218
avg_net_price_grants_scholarship_2020_2021    66.922218
sat_avg_all                                   35.263868
mn_earn_wne_indep1_p10                        26.135111
costt4_a                                       2.901967
unemployment_rate_2005                         0.142161
unemployment_rate_2006                         0.142161
unemployment_rate_2008                         0.020309
unemployment_rate_2009                         0.020309
unemployment_rate_2000                         0.020309
unemployment_rate_2001                         0.020309
unemployment_rate_2002                         0.020309
unemployment_rate_2003                         0.020309
unemployment_rate_2004                         0.020309
unemployment_rate_2007                         0.020309
unemployment_rate_2014                         0.005802
unemployment_rate_2013                         0.005802
unemployment_rate_2011                         0.005802
unemployment_rate_2010                         0.005802
```

```
unemployment_rate_2015                        0.005802
unemployment_rate_2016                        0.005802
unemployment_rate_2018                        0.005802
unemployment_rate_2019                        0.005802
unemployment_rate_2017                        0.005802
unemployment_rate_2012                        0.005802
unemployment_rate_2020                        0.000000
unemployment_rate_2022                        0.000000
median_household_income_2021                  0.000000
unemployment_rate_2021                        0.000000
unitid                                        0.000000
instnm                                        0.000000
ci90ubinc_2021                                0.000000
stabbr                                        0.000000
city                                          0.000000
region                                        0.000000
adm_rate_all                                  0.000000
tuitionfee_in                                 0.000000
tuitionfee_out                                0.000000
preddeg                                       0.000000
control                                       0.000000
state_abbr                                    0.000000
area_name_with_state                          0.000000
povall_2021                                   0.000000
ci90lball_2021                                0.000000
ci90uball_2021                                0.000000
medhhinc_2021                                 0.000000
ci90lbinc_2021                                0.000000
affordable                                    0.000000
dtype: float64
```

[ ]:
```
# Get the size of the dataframe before dropping null values
df.shape
```

[ ]:
```
(137872, 56)
```

[ ]:
```
# Drop null values
df = df.dropna()
```

[ ]:
```
# Get the size of the dataframe after dropping null values
df.shape
```

[ ]:
```
(32944, 56)
```

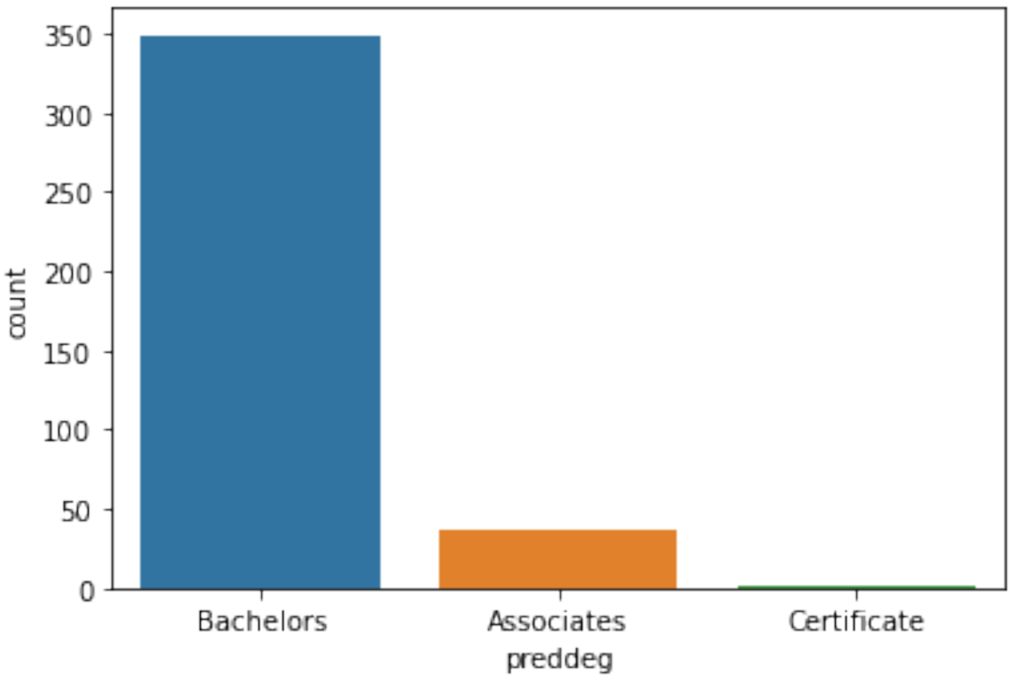
4.1 4.3 Additional Visualizations to Understand Bias

[ ]:
```
# Distribution of degree granted at universities
degree_count_query = """
SELECT preddeg, COUNT(preddeg) as count
FROM (
    SELECT DISTINCT instnm, preddeg
    FROM df
) AS unique_degrees
GROUP BY preddeg
ORDER BY count DESC;
"""
# Execute the SQL query on the Pandas DataFrames
degree_count = sqldf(degree_count_query, globals())
degree_count
```

[ ]:
```
                                    preddeg  count
0   Predominantly bachelor's-degree granting    349
1  Predominantly associate's-degree granting     36
2  Predominantly certificate-degree granting      2
```

4.1.1 4.3.1 Barplot of Primary Degree Granted at Universities

[ ]:
```
degree_labels = ['Bachelors', 'Associates', 'Certificate']

sns.barplot(data=degree_count, x='preddeg', y='count')
plt.xticks(ticks=range(len(degree_labels)), labels=degree_labels)
plt.show()
```



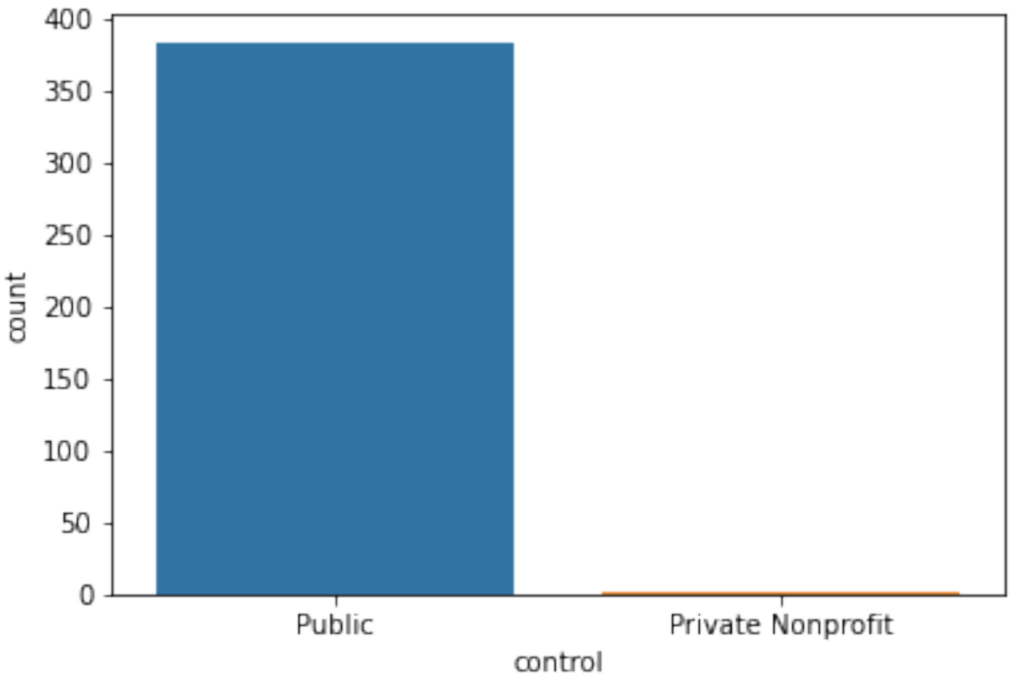

```
[ ]: # Certificates are under-represented, with only 2 observations so these will be␣
     ↪removed
     df = df[df.preddeg != 'Predominantly certificate-degree granting']

     degrees = {'Predominantly bachelor\'s-degree granting': 2,
                'Predominantly associate\'s-degree granting': 1}

     # Replace the values in the preddeg column using the mapping dictionary
     df['preddeg'] = df['preddeg'].map(degrees)
```

```
[ ]: # Distribution of university control system
     control_count_query = """
     SELECT control, COUNT(control) as count
     FROM (
         SELECT DISTINCT instnm, control
         FROM df
     ) AS unique_count
     GROUP BY control
     ORDER BY count DESC;
     """
     # Execute the SQL query on the Pandas DataFrames
     control_count = sqldf(control_count_query, globals())
     control_count
```

```
[ ]:            control  count
     0           Public    384
     1  Private nonprofit      1
```

4.1.2 4.3.2 Barplot of University Funding Control

```
[ ]: control_labels = ['Public','Private Nonprofit']

     sns.barplot(data=control_count, x='control', y='count')
     plt.xticks(ticks=range(len(control_labels)), labels=control_labels)
     plt.show()

     # Only one observation of private non-profit universities were found, thus␣
      ↪control labels will be removed from the dataset
```




```
[ ]: # Define all U.S. regions. Since these are cetegories, we might consider using␣
      ↪one-hot encoder
     regions = {'Southeast (AL AR FL GA KY LA MS NC SC TN VA WV)':'Southeast',
                'Great Lakes (IL IN MI OH WI)':'Great Lakes',
                'Southwest (AZ NM OK TX)':'Southwest',
                'Mid East (DE DC MD NJ NY PA)':'Mid East',
                'Plains (IA KS MN MO NE ND SD)':'Plains',
                'Far West (AK CA HI NV OR WA)':'Far East',
                'Rocky Mountains (CO ID MT UT WY)':'Rocky Mountains',
```

```
            'New England (CT ME MA NH RI VT)':'New England',
            'U.S. Service Schools':'U.S. Service Schools'}

df['region'] = df['region'].replace(regions)
```

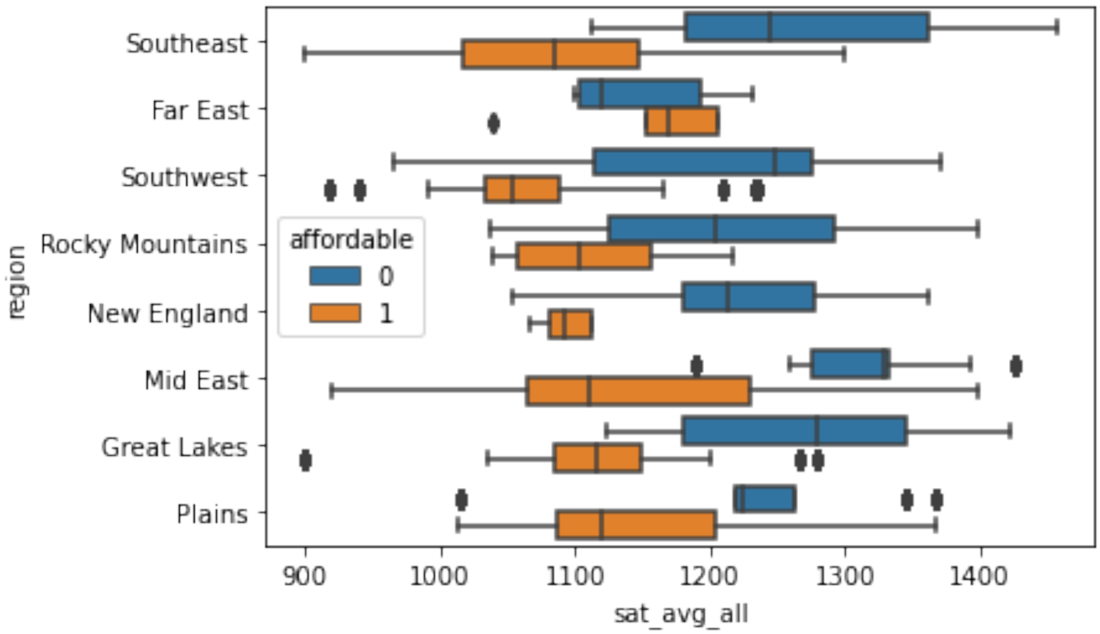
[ ]:
```
# Distribution of university control system
region_count_query = """
SELECT region, COUNT(*) as count
FROM (
    SELECT DISTINCT instnm, region
    FROM df
) AS unique_region
GROUP BY region
ORDER BY count DESC;
"""
# Execute the SQL query on the Pandas DataFrames
region_count = sqldf(region_count_query, globals())
region_count
```

[ ]:

	region	count
0	Southeast	119
1	Great Lakes	77
2	Mid East	53
3	Southwest	47
4	Plains	39
5	New England	22
6	Rocky Mountains	18
7	Far East	10

4.1.3 4.3.3 Boxplot of U.S. Regions, SAT Scores, and Affordability

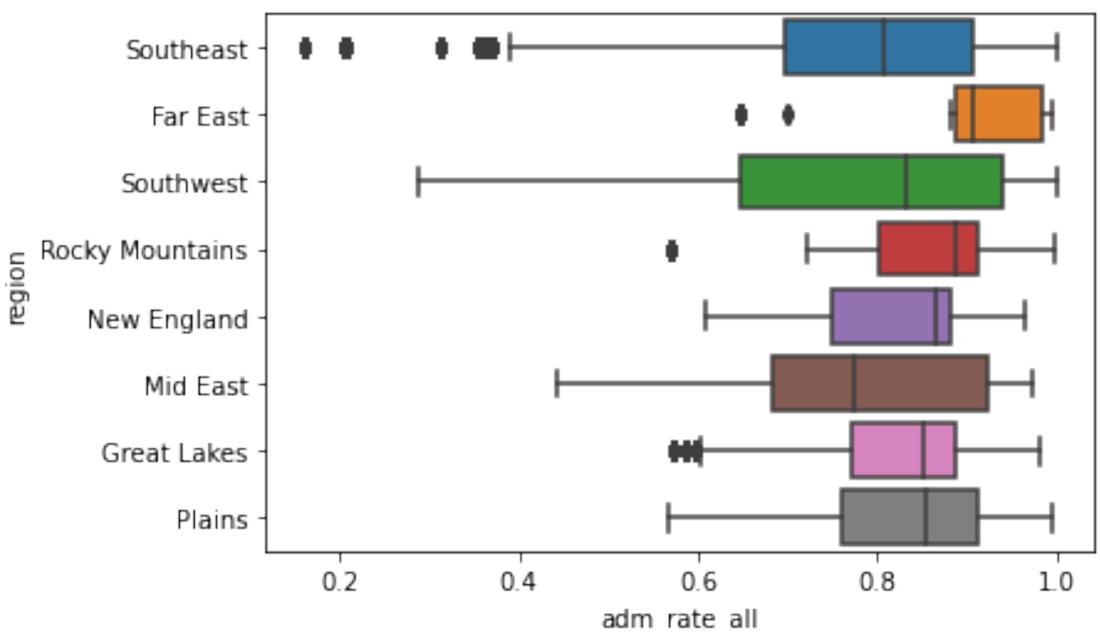
[ ]:
```
sns.boxplot(data=df, y='region', x='sat_avg_all', hue='affordable')
plt.figure(figsize=(10, 8))
plt.show()
```



<Figure size 720x576 with 0 Axes>

4.1.4 4.3.4 Boxplot of U.S. Regions and Admissions Rate

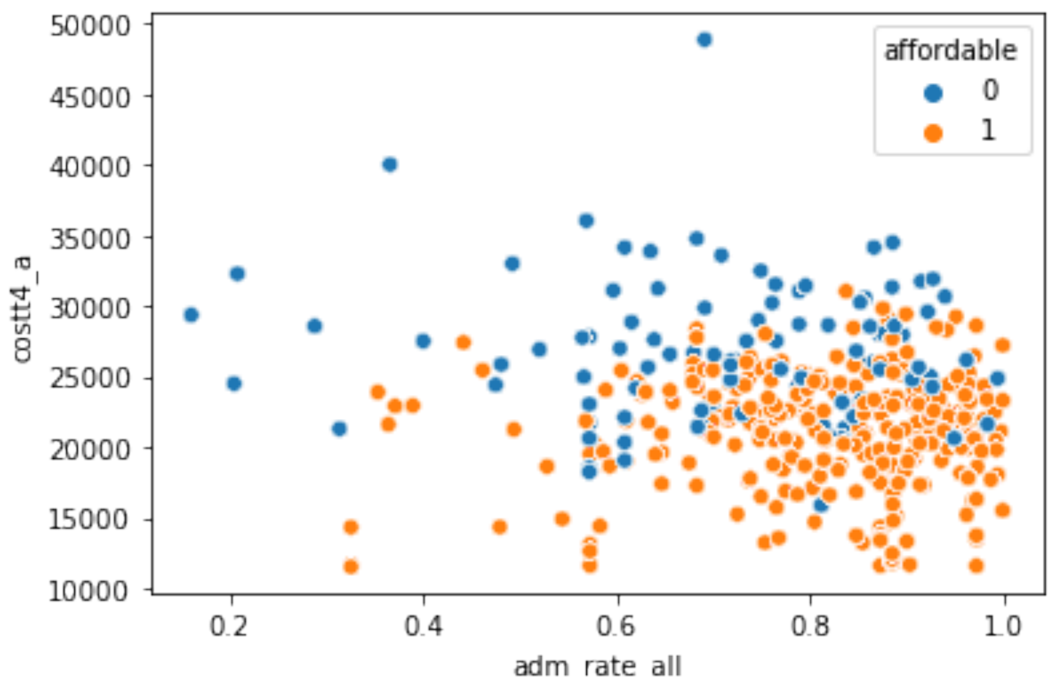
[ ]:
```
sns.boxplot(data=df, y='region', x='adm_rate_all')
plt.figure(figsize=(10, 8))
plt.show()
```



<Figure size 720x576 with 0 Axes>

4.1.5 4.3.5 Scatterplot of Admissions Rate, Overall Cost of Tuition, and Affordability

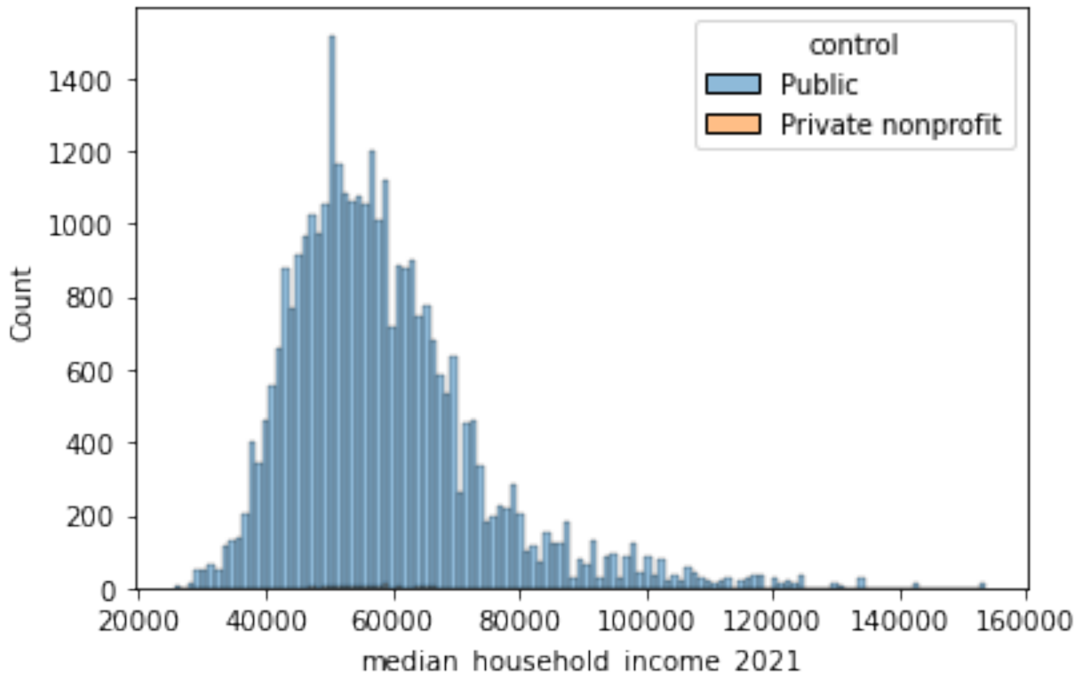
[ ]:
```
sns.scatterplot(data=df, x='adm_rate_all', y='costt4_a', hue='affordable')
plt.figure(figsize=(10, 8))
plt.show()
```



<Figure size 720x576 with 0 Axes>

4.1.6 4.3.6 Histogram of 2021 Median Household Income and Enrollment in University Funding Control

[ ]:
```
sns.histplot(data=df, x='median_household_income_2021', hue='control')
plt.figure(figsize=(10, 8))
plt.show()
```




```
<Figure size 720x576 with 0 Axes>
```

[ ]:
```python
import os

# Define the folder name
folder_name = 'data'

# Check if the folder exists, if not, create it
if not os.path.exists(folder_name):
    os.makedirs(folder_name)

# Save the DataFrame to a CSV file inside the folder
df.to_csv(os.path.join(folder_name, 'combined_data.csv'), index=False)
```

4.2 4.4 Encoding Categorical Labels

[ ]:
```python
df = pd.read_csv('data/combined_data.csv')
drop_categorical =␣
↪['instnm','stabbr','city','state_abbr','control','area_name_with_state']
df = df.drop(columns=drop_categorical)
df = df.drop(df.columns[0], axis=1)
```

[ ]:
```python
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
```

```python
# Encoding U.S. regions
encoded_df = pd.get_dummies(df, columns=['region'])

encoded_df.head()
```

```
[ ]:    adm_rate_all  sat_avg_all  costt4_a  tuitionfee_in  tuitionfee_out  \
     0      0.716006        954.0   21924.0        10024.0         18634.0
     1      0.716006        954.0   21924.0        10024.0         18634.0
     2      0.716006        954.0   21924.0        10024.0         18634.0
     3      0.716006        954.0   21924.0        10024.0         18634.0
     4      0.716006        954.0   21924.0        10024.0         18634.0

        mn_earn_wne_indep1_p10  preddeg  povall_2021  ci90lball_2021  \
     0                 38100.0        2       800848          782169
     1                 38100.0        2         6296            4772
     2                 38100.0        2        25526           21599
     3                 38100.0        2         5089            3773
     4                 38100.0        2         4204            3324

        ci90uball_2021  …  unemployment_rate_2012  affordable  region_Far East  \
     0          819527  …                     8.2           1                0
     1            7820  …                     7.1           1                0
     2           29453  …                     7.7           1                0
     3            6405  …                    11.8           1                0
     4            5084  …                     8.8           1                0

        region_Great Lakes  region_Mid East  region_New England  region_Plains  \
     0                   0                0                   0              0
     1                   0                0                   0              0
     2                   0                0                   0              0
     3                   0                0                   0              0
     4                   0                0                   0              0

        region_Rocky Mountains  region_Southeast  region_Southwest
     0                       0                 1                 0
     1                       0                 1                 0
     2                       0                 1                 0
     3                       0                 1                 0
     4                       0                 1                 0

     [5 rows x 56 columns]
```

4.3 4.5 Normalize Numerical Values

[ ]:
```
from sklearn.preprocessing import MinMaxScaler

# Columns to apply MinMaxScaler
numerical_features = ['costt4_a','tuitionfee_in','tuitionfee_out',␣
 ↪'mn_earn_wne_indep1_p10', 'povall_2021','medhhinc_2021',
                          ␣
 ↪'unemployment_rate_2000','unemployment_rate_2001','unemployment_rate_2002','unemployment_ra
 ↪'unemployment_rate_2004','unemployment_rate_2005',
                           'unemployment_rate_2006',␣
 ↪'unemployment_rate_2007','unemployment_rate_2008','unemployment_rate_2009','unemployment_ra
                          ␣
 ↪'unemployment_rate_2013','unemployment_rate_2014','unemployment_rate_2015','unemployment_ra
 ↪'unemployment_rate_2018',
                           'unemployment_rate_2019', 'unemployment_rate_2020',␣
 ↪'unemployment_rate_2021',␣
 ↪'unemployment_rate_2022','median_household_income_2021',
                          ␣
 ↪'avg_net_price_grants_scholarship_2020_2021','avg_net_price_grants_scholarship_2019_2020','
                           'avg_net_price_grants_scholarship_2017_2018',␣
 ↪'avg_net_price_grants_scholarship_2016_2017',␣
 ↪'avg_net_price_grants_scholarship_2015_2016',
                           'avg_net_price_grants_scholarship_2014_2015',␣
 ↪'avg_net_price_grants_scholarship_2013_2014',␣
 ↪'avg_net_price_grants_scholarship_2012_2013',␣
 ↪'avg_net_price_grants_scholarship_2011_2012',
                           'unemployment_rate_2012',]

scaler = MinMaxScaler()

# Store scaled values into new df
scaled_df = encoded_df.copy()
scaled_df[numerical_features] = scaler.
 ↪fit_transform(encoded_df[numerical_features])

# Print scaled features
scaled_df.head()
```

[ ]:
```
   adm_rate_all  sat_avg_all  costt4_a  tuitionfee_in  tuitionfee_out  \
0      0.716006        954.0  0.278705       0.262365         0.31364
1      0.716006        954.0  0.278705       0.262365         0.31364
2      0.716006        954.0  0.278705       0.262365         0.31364
3      0.716006        954.0  0.278705       0.262365         0.31364
4      0.716006        954.0  0.278705       0.262365         0.31364

   mn_earn_wne_indep1_p10  preddeg  povall_2021  ci90lball_2021  \
```

```
0             0.193314       2     0.194248          782169
1             0.193314       2     0.001526            4772
2             0.193314       2     0.006191           21599
3             0.193314       2     0.001234            3773
4             0.193314       2     0.001019            3324

   ci90uball_2021  …  unemployment_rate_2012  affordable  region_Far East  \
0          819527  …                0.311404           1                0
1            7820  …                0.263158           1                0
2           29453  …                0.289474           1                0
3            6405  …                0.469298           1                0
4            5084  …                0.337719           1                0

   region_Great Lakes  region_Mid East  region_New England  region_Plains  \
0                   0                0                   0              0
1                   0                0                   0              0
2                   0                0                   0              0
3                   0                0                   0              0
4                   0                0                   0              0

   region_Rocky Mountains  region_Southeast  region_Southwest
0                       0                 1                 0
1                       0                 1                 0
2                       0                 1                 0
3                       0                 1                 0
4                       0                 1                 0

[5 rows x 56 columns]
```

[ ]:
```
scaled_df.dtypes
```

[ ]:
```
adm_rate_all                              float64
sat_avg_all                               float64
costt4_a                                  float64
tuitionfee_in                             float64
tuitionfee_out                            float64
mn_earn_wne_indep1_p10                    float64
preddeg                                     int64
povall_2021                               float64
ci90lball_2021                              int64
ci90uball_2021                              int64
medhhinc_2021                             float64
ci90lbinc_2021                              int64
ci90ubinc_2021                              int64
unemployment_rate_2000                    float64
unemployment_rate_2001                    float64
unemployment_rate_2002                    float64
```

```
unemployment_rate_2003                             float64
unemployment_rate_2004                             float64
unemployment_rate_2005                             float64
unemployment_rate_2006                             float64
unemployment_rate_2007                             float64
unemployment_rate_2008                             float64
unemployment_rate_2009                             float64
unemployment_rate_2010                             float64
unemployment_rate_2011                             float64
unemployment_rate_2013                             float64
unemployment_rate_2014                             float64
unemployment_rate_2015                             float64
unemployment_rate_2016                             float64
unemployment_rate_2017                             float64
unemployment_rate_2018                             float64
unemployment_rate_2019                             float64
unemployment_rate_2020                             float64
unemployment_rate_2021                             float64
unemployment_rate_2022                             float64
median_household_income_2021                       float64
avg_net_price_grants_scholarship_2020_2021         float64
avg_net_price_grants_scholarship_2019_2020         float64
avg_net_price_grants_scholarship_2018_2019         float64
avg_net_price_grants_scholarship_2017_2018         float64
avg_net_price_grants_scholarship_2016_2017         float64
avg_net_price_grants_scholarship_2015_2016         float64
avg_net_price_grants_scholarship_2014_2015         float64
avg_net_price_grants_scholarship_2013_2014         float64
avg_net_price_grants_scholarship_2012_2013         float64
avg_net_price_grants_scholarship_2011_2012         float64
unemployment_rate_2012                             float64
affordable                                           int64
region_Far East                                      uint8
region_Great Lakes                                   uint8
region_Mid East                                      uint8
region_New England                                   uint8
region_Plains                                        uint8
region_Rocky Mountains                               uint8
region_Southeast                                     uint8
region_Southwest                                     uint8
dtype: object
```

5 5. Modeling

[ ]:
```python
from sklearn.model_selection import train_test_split

# Define features and target variable
X = scaled_df.drop(columns=['affordable'])
y = scaled_df['affordable']

# Split the data into train and temporary sets (70% train, 30% temp)
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3,␣
↪random_state=42)

# Split the temporary set into validation and test sets (50% for validation,␣
↪50% for test)
X_valid, X_test, y_valid, y_test = train_test_split(X_temp, y_temp, test_size=0.
↪5, random_state=42)
```

[ ]:
```python
# Save data to CSV files
train_df = pd.concat([y_train, X_train], axis=1)
valid_df = pd.concat([y_valid, X_valid], axis=1)
test_df = pd.concat([y_test, X_test], axis=1)

# Save the combined DataFrame to a CSV file
train_df.to_csv(os.path.join('data', 'train_combined.csv'), index=False)
X_train.to_csv(os.path.join('data', 'train_features.csv'), index=False)
y_train.to_csv(os.path.join('data', 'train_labels.csv'), index=False,␣
↪header=False)

valid_df.to_csv(os.path.join('data', 'valid_combined.csv'), index=False)
X_valid.to_csv(os.path.join('data', 'valid_features.csv'), index=False)
y_valid.to_csv(os.path.join('data', 'valid_labels.csv'), index=False,␣
↪header=False)

test_df.to_csv(os.path.join('data', 'test_combined.csv'), index=False)
X_test.to_csv(os.path.join('data', 'test_features.csv'), index=False)
y_test.to_csv(os.path.join('data', 'test_labels.csv'), index=False,␣
↪header=False)

# Create a subset based on correlated attributes
columns_to_include = ['affordable', 'sat_avg_all', 'costt4_a',␣
↪'avg_net_price_grants_scholarship_2020_2021',
                      'avg_net_price_grants_scholarship_2019_2020']

train_subset = train_df[columns_to_include].copy()
valid_subset = valid_df[columns_to_include].copy()
test_subset = test_df[columns_to_include].copy()
```

```
train_subset.to_csv(os.path.join('data', 'train_subset.csv'), index=False,␣
↪header=False)
valid_subset.to_csv(os.path.join('data', 'valid_subset.csv'), index=False,␣
↪header=False)
test_subset.to_csv(os.path.join('data', 'test_subset.csv'), index=False,␣
↪header=False)

X_test_subset = test_subset.drop(test_subset.columns[0], axis=1)
X_test_subset.to_csv(os.path.join('data', 'X_test_subset.csv'), index=False,␣
↪header=False)
```

[ ]:
```
# Print the shapes of the resulting sets
print("Train set shape:", X_train.shape)
print("Train set shape:", y_train.shape)

print("Validation set shape:", X_valid.shape)
print("Validation set shape:", y_valid.shape)

print("Test set shape:", X_test.shape)
print("Test set shape:", y_test.shape)
```

```
Train set shape: (22997, 55)
Train set shape: (22997,)
Validation set shape: (4928, 55)
Validation set shape: (4928,)
Test set shape: (4928, 55)
Test set shape: (4928,)
```

[ ]:
```
# Private bucket for JVo
import boto3

# Create a private S3 bucket
s3 = boto3.resource('s3')
bucket_name = "collegeaffordability508"
bucket = s3.create_bucket(Bucket=bucket_name, ACL='private')

# Upload CSV files to S3
s3_client = boto3.client('s3')
s3_private_path = "s3://{}/".format(bucket_name)
```

[ ]:
```
# Upload train data
s3_client.upload_file('data/train_subset.csv', bucket_name, 'train/train_subset.
↪csv')
s3_client.upload_file('data/train_combined.csv', bucket_name, 'train/
↪train_combined.csv')
s3_client.upload_file('data/train_features.csv', bucket_name, 'train/
↪train_features.csv')
```

```
s3_client.upload_file('data/train_labels.csv', bucket_name, 'train/train_labels.
 ↪csv')

# Upload validation data
s3_client.upload_file('data/valid_subset.csv', bucket_name, 'validation/
 ↪valid_subset.csv')
s3_client.upload_file('data/valid_combined.csv', bucket_name, 'validation/
 ↪valid_combined.csv')
s3_client.upload_file('data/valid_features.csv', bucket_name, 'validation/
 ↪valid_features.csv')
s3_client.upload_file('data/valid_labels.csv', bucket_name, 'validation/
 ↪valid_labels.csv')

# Upload test data
s3_client.upload_file('data/test_subset.csv', bucket_name, 'test/test_subset.
 ↪csv')
s3_client.upload_file('data/X_test_subset.csv', bucket_name, 'test/
 ↪X_test_subset.csv')
s3_client.upload_file('data/test_combined.csv', bucket_name, 'test/
 ↪test_combined.csv')
s3_client.upload_file('data/test_features.csv', bucket_name, 'test/
 ↪test_features.csv')
s3_client.upload_file('data/test_labels.csv', bucket_name, 'test/test_labels.
 ↪csv')
```

5.0.1 5.1 Logistic Regression

[ ]:
```
# Train an inital logistic regression model
log_reg_model = LogisticRegression()
log_reg_model.fit(X_train, y_train)

# Predictions on validation set
y_valid_pred = log_reg_model.predict(X_valid)

# Evaluate the model on validation set
valid_accuracy = accuracy_score(y_valid, y_valid_pred)
print("Validation Accuracy:", valid_accuracy.round(2))

# Predictions on test set
y_test_pred = log_reg_model.predict(X_test)

# Evaluate the model on test set
test_accuracy = accuracy_score(y_test, y_test_pred)
print("Test Accuracy:", test_accuracy.round(2))

# Print classification report
```

```
print("\nClassification Report:")
print(classification_report(y_valid, y_valid_pred))
```

```
Validation Accuracy: 0.76
Test Accuracy: 0.76

Classification Report:
              precision    recall  f1-score   support

           0       0.33      0.00      0.00      1199
           1       0.76      1.00      0.86      3729

    accuracy                           0.76      4928
   macro avg       0.55      0.50      0.43      4928
weighted avg       0.65      0.76      0.65      4928
```

5.0.2 5.2 XG Boost

[ ]:
```
# Define IAM role
role = sagemaker.get_execution_role()

# Set the region of the instance
my_region = boto3.session.Session().region_name

# This line automatically looks for the XGBoost image URI and
# builds an XGBoost container.
xgboost_container = sagemaker.image_uris.retrieve("xgboost",
                                                  my_region,
                                                  "latest")

print("Success - the MySageMakerInstance is in the " + my_region + \
      " region. You will use the " + xgboost_container + \
      " container for your SageMaker endpoint.")
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/root/.config/sagemaker/config.yaml
Success - the MySageMakerInstance is in the us-east-1 region. You will use the
811284229777.dkr.ecr.us-east-1.amazonaws.com/xgboost:latest container for your
SageMaker endpoint.
```

[ ]:
```
# Load the data from S3
bucket_name = 'collegeaffordability508'
s3_client = boto3.client('s3')
```

```python
train_data_uri = f's3://{bucket_name}/train/train_subset.csv'
validation_data_uri = f's3://{bucket_name}/validation/valid_subset.csv'

sess = sagemaker.Session()

# Retrieve the XGBoost container image
xgboost_container = retrieve(region=boto3.Session().region_name,␣
 ↪framework='xgboost', version='latest')

# Define the estimator
xgb = sagemaker.estimator.Estimator(xgboost_container,
                                    role,
                                    instance_count=1,
                                    instance_type='ml.m5.xlarge',
                                    output_path=f's3://{bucket_name}/output',
                                    sagemaker_session=sess)

# Parse in the hyperparameters
xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        num_round=100)
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/root/.config/sagemaker/config.yaml
```

[ ]:
```python
# Define data channels for training
train_input = TrainingInput(train_data_uri, content_type='text/csv')
validation_input = TrainingInput(validation_data_uri, content_type='text/csv')

# Train the XGBoost model
xgb.fit({'train': train_input, 'validation': validation_input})
```

```
INFO:sagemaker:Creating training-job with name: xgboost-2024-04-14-17-06-28-654
```

```
2024-04-14 17:06:28 Starting - Starting the training job…
2024-04-14 17:06:44 Starting - Preparing the instances for training…
2024-04-14 17:07:15 Downloading - Downloading input data…
2024-04-14 17:07:40 Downloading - Downloading the training image…
2024-04-14 17:08:25 Training - Training image download completed. Training in
progress.
2024-04-14 17:08:25 Uploading - Uploading generated training
```

```
modelArguments: train
[2024-04-14:17:08:18:INFO] Running standalone xgboost training.
[2024-04-14:17:08:18:INFO] File size need to be processed in the node:

1.78mb. Available memory size in the node: 7994.32mb
[2024-04-14:17:08:18:INFO] Determined delimiter of CSV input is ','
[17:08:18] S3DistributionType set as FullyReplicated
[17:08:18] 22997x4 matrix with 91988 entries loaded from

/opt/ml/input/data/train?format=csv&label_column=0&delimiter=,
[2024-04-14:17:08:18:INFO] Determined delimiter of CSV input is ','
[17:08:18] S3DistributionType set as FullyReplicated
[17:08:18] 4928x4 matrix with 19712 entries loaded from

/opt/ml/input/data/validation?format=csv&label_column=0&delimiter=,
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 36

extra nodes, 0 pruned nodes, max_depth=5
[0]#011train-error:0.069922#011validation-error:0.06737
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38

extra nodes, 4 pruned nodes, max_depth=5
[1]#011train-error:0.054442#011validation-error:0.050122
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 46

extra nodes, 0 pruned nodes, max_depth=5
[2]#011train-error:0.042875#011validation-error:0.041599
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 44

extra nodes, 4 pruned nodes, max_depth=5
[3]#011train-error:0.035918#011validation-error:0.033279
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38

extra nodes, 2 pruned nodes, max_depth=5
[4]#011train-error:0.032222#011validation-error:0.026583
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38

extra nodes, 0 pruned nodes, max_depth=5
[5]#011train-error:0.034135#011validation-error:0.029627
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 34

extra nodes, 2 pruned nodes, max_depth=5
[6]#011train-error:0.031526#011validation-error:0.026989
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 48

extra nodes, 4 pruned nodes, max_depth=5
[7]#011train-error:0.028221#011validation-error:0.024148
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 34

extra nodes, 0 pruned nodes, max_depth=5
[8]#011train-error:0.023525#011validation-error:0.018466
```

```
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 34
extra nodes, 4 pruned nodes, max_depth=5
[9]#011train-error:0.023525#011validation-error:0.018466
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 42
extra nodes, 2 pruned nodes, max_depth=5
[10]#011train-error:0.023525#011validation-error:0.018466
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 42
extra nodes, 4 pruned nodes, max_depth=5
[11]#011train-error:0.017437#011validation-error:0.012784
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 40
extra nodes, 4 pruned nodes, max_depth=5
[12]#011train-error:0.015741#011validation-error:0.011161
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30
extra nodes, 4 pruned nodes, max_depth=5
[13]#011train-error:0.013263#011validation-error:0.00832
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 30
extra nodes, 4 pruned nodes, max_depth=5
[14]#011train-error:0.010132#011validation-error:0.006494
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28
extra nodes, 8 pruned nodes, max_depth=5
[15]#011train-error:0.010132#011validation-error:0.006494
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 34
extra nodes, 0 pruned nodes, max_depth=5
[16]#011train-error:0.010132#011validation-error:0.006494
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32
extra nodes, 2 pruned nodes, max_depth=5
[17]#011train-error:0.008131#011validation-error:0.00487
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24
extra nodes, 0 pruned nodes, max_depth=5
[18]#011train-error:0.007175#011validation-error:0.004464
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32
extra nodes, 2 pruned nodes, max_depth=5
[19]#011train-error:0.007175#011validation-error:0.004464
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28
extra nodes, 4 pruned nodes, max_depth=5
[20]#011train-error:0.008131#011validation-error:0.00487
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 38
extra nodes, 4 pruned nodes, max_depth=5
[21]#011train-error:0.00587#011validation-error:0.003653
```

```
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24
extra nodes, 0 pruned nodes, max_depth=5
[22]#011train-error:0.007175#011validation-error:0.004464
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 36
extra nodes, 0 pruned nodes, max_depth=5
[23]#011train-error:0.003305#011validation-error:0.002232
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18
extra nodes, 4 pruned nodes, max_depth=5
[24]#011train-error:0.002131#011validation-error:0.001623
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28
extra nodes, 0 pruned nodes, max_depth=5
[25]#011train-error:0.002131#011validation-error:0.001623
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 26
extra nodes, 2 pruned nodes, max_depth=5
[26]#011train-error:0.002131#011validation-error:0.001623
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20
extra nodes, 0 pruned nodes, max_depth=5
[27]#011train-error:0.002131#011validation-error:0.001623
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 26
extra nodes, 0 pruned nodes, max_depth=5
[28]#011train-error:0.002131#011validation-error:0.001623
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18
extra nodes, 4 pruned nodes, max_depth=5
[29]#011train-error:0.001913#011validation-error:0.00142
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24
extra nodes, 2 pruned nodes, max_depth=5
[30]#011train-error:0.00187#011validation-error:0.000812
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20
extra nodes, 4 pruned nodes, max_depth=5
[31]#011train-error:0.00187#011validation-error:0.000812
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12
extra nodes, 4 pruned nodes, max_depth=5
[32]#011train-error:0.001522#011validation-error:0.000609
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 28
extra nodes, 2 pruned nodes, max_depth=5
[33]#011train-error:0.001174#011validation-error:0.000406
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16
extra nodes, 4 pruned nodes, max_depth=5
[34]#011train-error:0.001174#011validation-error:0.000406
```

```
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14
extra nodes, 4 pruned nodes, max_depth=5
[35]#011train-error:0.001522#011validation-error:0.000609
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14
extra nodes, 4 pruned nodes, max_depth=5
[36]#011train-error:0.001522#011validation-error:0.000609
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 42
extra nodes, 4 pruned nodes, max_depth=5
[37]#011train-error:0.000957#011validation-error:0.000203
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 32
extra nodes, 2 pruned nodes, max_depth=5
[38]#011train-error:0.000957#011validation-error:0.000203
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12
extra nodes, 6 pruned nodes, max_depth=5
[39]#011train-error:0.000609#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14
extra nodes, 4 pruned nodes, max_depth=5
[40]#011train-error:0.000609#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 26
extra nodes, 4 pruned nodes, max_depth=5
[41]#011train-error:0.000609#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 22
extra nodes, 2 pruned nodes, max_depth=5
[42]#011train-error:0.000609#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 24
extra nodes, 8 pruned nodes, max_depth=5
[43]#011train-error:0.000435#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 22
extra nodes, 2 pruned nodes, max_depth=5
[44]#011train-error:0.000435#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14
extra nodes, 6 pruned nodes, max_depth=5
[45]#011train-error:0.000435#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16
extra nodes, 2 pruned nodes, max_depth=5
[46]#011train-error:0.000304#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18
extra nodes, 4 pruned nodes, max_depth=5
[47]#011train-error:0#011validation-error:0
```

```
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14
extra nodes, 2 pruned nodes, max_depth=5
[48]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16
extra nodes, 8 pruned nodes, max_depth=5
[49]#011train-error:0.000304#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20
extra nodes, 6 pruned nodes, max_depth=5
[50]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 22
extra nodes, 8 pruned nodes, max_depth=5
[51]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 10 pruned nodes, max_depth=5
[52]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12
extra nodes, 6 pruned nodes, max_depth=5
[53]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14
extra nodes, 2 pruned nodes, max_depth=5
[54]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 22
extra nodes, 2 pruned nodes, max_depth=5
[55]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16
extra nodes, 2 pruned nodes, max_depth=5
[56]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 4 pruned nodes, max_depth=5
[57]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12
extra nodes, 4 pruned nodes, max_depth=5
[58]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 22
extra nodes, 6 pruned nodes, max_depth=5
[59]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra
nodes, 14 pruned nodes, max_depth=3
[60]#011train-error:0#011validation-error:0
```

```
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16
extra nodes, 8 pruned nodes, max_depth=5
[61]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12
extra nodes, 2 pruned nodes, max_depth=5
[62]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra
nodes, 6 pruned nodes, max_depth=3
[63]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 16
extra nodes, 8 pruned nodes, max_depth=4
[64]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12
extra nodes, 4 pruned nodes, max_depth=5
[65]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20
extra nodes, 2 pruned nodes, max_depth=5
[66]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 12 pruned nodes, max_depth=4
[67]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 8 pruned nodes, max_depth=5
[68]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 12 pruned nodes, max_depth=0
[69]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 6 pruned nodes, max_depth=4
[70]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 6 pruned nodes, max_depth=5
[71]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 6 pruned nodes, max_depth=5
[72]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 20
extra nodes, 8 pruned nodes, max_depth=5
[73]#011train-error:0#011validation-error:0
```

```
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 20 pruned nodes, max_depth=0
[74]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra
nodes, 14 pruned nodes, max_depth=3
[75]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 2 pruned nodes, max_depth=5
[76]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 12 pruned nodes, max_depth=0
[77]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 18
extra nodes, 8 pruned nodes, max_depth=5
[78]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 12 pruned nodes, max_depth=0
[79]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 4 pruned nodes, max_depth=5
[80]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 8 extra
nodes, 14 pruned nodes, max_depth=4
[81]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 14
extra nodes, 12 pruned nodes, max_depth=5
[82]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 14 pruned nodes, max_depth=5
[83]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 18 pruned nodes, max_depth=0
[84]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 20 pruned nodes, max_depth=0
[85]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 10 pruned nodes, max_depth=5
[86]#011train-error:0#011validation-error:0
```

```
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 20 pruned nodes, max_depth=0
[87]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 8 pruned nodes, max_depth=5
[88]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 12 pruned nodes, max_depth=0
[89]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 14 pruned nodes, max_depth=0
[90]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 10
extra nodes, 8 pruned nodes, max_depth=5
[91]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra
nodes, 6 pruned nodes, max_depth=3
[92]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 10 pruned nodes, max_depth=0
[93]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 12
extra nodes, 4 pruned nodes, max_depth=5
[94]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 16 pruned nodes, max_depth=0
[95]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 20 pruned nodes, max_depth=0
[96]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 10 pruned nodes, max_depth=0
[97]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 0 extra
nodes, 18 pruned nodes, max_depth=0
[98]#011train-error:0#011validation-error:0
[17:08:18] src/tree/updater_prune.cc:74: tree pruning end, 1 roots, 6 extra
nodes, 16 pruned nodes, max_depth=3
[99]#011train-error:0#011validation-error:0
```

```
2024-04-14 17:08:37 Completed - Training job completed
Training seconds: 82
Billable seconds: 82
```

[ ]:
```python
from sagemaker.serializers import CSVSerializer
from sklearn.metrics import accuracy_score, classification_report

# Load the test labels from S3
test_labels_uri = f's3://{bucket_name}/test/test_labels.csv'
test_labels = pd.read_csv(test_labels_uri)

# Assuming test_labels contains the ground truth labels

# Define the test data URI
test_data_uri = f's3://{bucket_name}/test/X_test_subset.csv'

# Read the test data from S3
test_data = pd.read_csv(test_data_uri)

# Convert the test data to CSV format
csv_data = test_data.to_csv(index=False, header=False).rstrip('\n')

# Deploy the model
xgb_predictor = xgb.deploy(initial_instance_count=1, instance_type='ml.m5.
↪large', serializer=CSVSerializer())

# Evaluate the model on the test data
test_predictions = xgb_predictor.predict(csv_data).decode('utf-8')

# Convert predictions to DataFrame
predicted_labels = pd.DataFrame([int(float(x)) for x in test_predictions.
↪split(',')])

# Calculate accuracy score
accuracy = accuracy_score(test_labels, predicted_labels)

# Generate classification report
classification_rep = classification_report(test_labels, predicted_labels)

print("Accuracy Score:", accuracy)
print("Classification Report:\n", classification_rep)
```

```
INFO:sagemaker:Creating model with name: xgboost-2024-04-14-17-56-14-238
INFO:sagemaker:Creating endpoint-config with name
xgboost-2024-04-14-17-56-14-238
INFO:sagemaker:Creating endpoint with name xgboost-2024-04-14-17-56-14-238

-----!Accuracy Score: 0.23807590826060482
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.24      1.00      0.38      1173
           1       0.00      0.00      0.00      3754

    accuracy                           0.24      4927
   macro avg       0.12      0.50      0.19      4927
weighted avg       0.06      0.24      0.09      4927

/opt/conda/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/opt/conda/lib/python3.8/site-packages/sklearn/metrics/_classification.py:1471:
UndefinedMetricWarning: Precision and F-score are ill-defined and being set to
0.0 in labels with no predicted samples. Use `zero_division` parameter to
control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Try Improving the Model with Randomly Selected Hyperparameters

[ ]:
```python
#import boto3
#import sagemaker
from sagemaker.tuner import IntegerParameter, ContinuousParameter,␣
 ↪HyperparameterTuner

# Load the data from S3
bucket_name = 'collegeaffordability508'
train_data_uri = f's3://{bucket_name}/train/train_subset.csv'
validation_data_uri = f's3://{bucket_name}/validation/valid_subset.csv'

# Define the SageMaker session
sess = sagemaker.Session()

# Retrieve the XGBoost container image
xgboost_container = sagemaker.image_uris.retrieve("xgboost", sess.
 ↪boto_region_name, "latest")

# Define the estimator
xgb = sagemaker.estimator.Estimator(xgboost_container,
```

```python
                                    role,
                                    instance_count=1,
                                    instance_type='ml.m5.xlarge',
                                    output_path=f's3://{bucket_name}/output',
                                    sagemaker_session=sess)

# Define hyperparameter ranges for random search
hyperparameter_ranges = {
    "max_depth": IntegerParameter(3, 10),
    "eta": ContinuousParameter(0.01, 0.5),
    "gamma": ContinuousParameter(0, 10),
    "min_child_weight": IntegerParameter(1, 10),
    "subsample": ContinuousParameter(0.5, 1),
    "num_round": IntegerParameter(50, 200)
}

# Define the objective metric for hyperparameter optimization
objective_metric_name = "validation:auc"

# Define the hyperparameter tuner
hyperparameter_tuner = HyperparameterTuner(estimator=xgb,
                                           ␣
  ↪objective_metric_name=objective_metric_name,
                                           ␣
  ↪hyperparameter_ranges=hyperparameter_ranges,
                                           strategy="Random",
                                           max_jobs=10,
                                           max_parallel_jobs=2)
```

```
sagemaker.config INFO - Not applying SDK defaults from location:
/etc/xdg/sagemaker/config.yaml
sagemaker.config INFO - Not applying SDK defaults from location:
/root/.config/sagemaker/config.yaml
```

```
INFO:sagemaker.image_uris:Ignoring unnecessary instance type: None.
```

[ ]:
```python
# Define data channels for training
train_input = TrainingInput(train_data_uri, content_type='text/csv')
validation_input = TrainingInput(validation_data_uri, content_type='text/csv')

# Train the model with hyperparameter tuning
hyperparameter_tuner.fit({"train": train_input, "validation": validation_input})
```

```
WARNING:sagemaker.estimator:No finished training job found associated with this
estimator. Please make sure this estimator is only used for building workflow
config
WARNING:sagemaker.estimator:No finished training job found associated with this
estimator. Please make sure this estimator is only used for building workflow
```

```
config
INFO:sagemaker:Creating hyperparameter tuning job with name: xgboost-240414-1801

…!
```

[ ]:
```python
# Define the test data URI
test_data_uri = f's3://{bucket_name}/test/X_test_subset.csv'
test_labels_uri = f's3://{bucket_name}/test/test_labels.csv'

# Load the test data and labels from S3
test_data = pd.read_csv(test_data_uri)
test_labels = pd.read_csv(test_labels_uri)

# Deploy the best model found during hyperparameter tuning
xgb_predictor = hyperparameter_tuner.deploy(initial_instance_count=1,␣
 ↪instance_type='ml.m5.large', serializer=CSVSerializer())

# Convert the test data to CSV format
csv_data = test_data.to_csv(index=False, header=False).rstrip('\n')

# Evaluate the model on the test data
test_predictions = xgb_predictor.predict(csv_data).decode('utf-8')

# Convert predictions to DataFrame
predicted_labels = pd.DataFrame([int(float(x)) for x in test_predictions.
 ↪split(',')])

# Calculate accuracy score
accuracy = accuracy_score(test_labels, predicted_labels)

# Generate classification report
classification_rep = classification_report(test_labels, predicted_labels)

print("Accuracy Score:", accuracy)
print("Classification Report:\n", classification_rep)
```

```
2024-04-14 18:04:20 Starting - Preparing the instances for training
2024-04-14 18:04:20 Downloading - Downloading the training image
2024-04-14 18:04:20 Training - Training image download completed. Training in
progress.
2024-04-14 18:04:20 Uploading - Uploading generated training model
2024-04-14 18:04:20 Completed - Resource reused by training job:
xgboost-240414-1801-004-040277d0

INFO:sagemaker:Creating model with name: xgboost-2024-04-14-19-13-11-797

INFO:sagemaker:Creating endpoint-config with name
```

```
xgboost-240414-1801-002-799b86a7
INFO:sagemaker:Creating endpoint with name xgboost-240414-1801-002-799b86a7
```

```
-----!Accuracy Score: 0.3109397199106962
Classification Report:
              precision    recall  f1-score   support

           0       0.26      1.00      0.41      1173
           1       1.00      0.10      0.17      3754

    accuracy                           0.31      4927
   macro avg       0.63      0.55      0.29      4927
weighted avg       0.82      0.31      0.23      4927
```

[ ]:
```
# Delete the endpoint to keep costs minimal
xgb_predictor.delete_endpoint()
```

```
INFO:sagemaker:Deleting endpoint configuration with name:
xgboost-240414-1801-002-799b86a7
INFO:sagemaker:Deleting endpoint with name: xgboost-240414-1801-002-799b86a7
```

5.0.3 5.3 Random Forrest

[ ]:
```
from sklearn.ensemble import RandomForestClassifier

rf_model = RandomForestClassifier(n_estimators = 500,
                            max_depth = 4,
                            max_features = 3,
                            bootstrap = True,
                            random_state = 508)
rf_model.fit(X_train, y_train)

y_valid_pred = rf_model.predict(X_valid)


# Evaluate the model on validation set
valid_accuracy = accuracy_score(y_valid, y_valid_pred)
print("Validation Accuracy:", valid_accuracy.round(2))

# Predictions on test set
y_test_pred = rf_model.predict(X_test)

# Evaluate the model on test set
test_accuracy = accuracy_score(y_test, y_test_pred)
print("Test Accuracy:", test_accuracy.round(2))

# Print classification report
```

```
print("\nClassification Report:")
print(classification_report(y_valid, y_valid_pred))
```

```
Validation Accuracy: 0.95
Test Accuracy: 0.95

Classification Report:
              precision    recall  f1-score   support

           0       1.00      0.79      0.89      1199
           1       0.94      1.00      0.97      3729

    accuracy                           0.95      4928
   macro avg       0.97      0.90      0.93      4928
weighted avg       0.95      0.95      0.95      4928
```

Release Sagemaker Resources

[ ]:
```
%%javascript

try {
    Jupyter.notebook.save_checkpoint();
    Jupyter.notebook.session.delete();
}
catch(err) {
    // NoOp
}
```

```
<IPython.core.display.Javascript object>
```

[ ]:
```
%%html

<p><b>Shutting down your kernel for this notebook to release resources.</b></p>
<button class="sm-command-button" data-commandlinker-command="kernelmenu:
↪shutdown" style="display:none;">Shutdown Kernel</button>

<script>
try {
    els = document.getElementsByClassName("sm-command-button");
    els[0].click();
}
catch(err) {
    // NoOp
}
</script>
```

```
<IPython.core.display.HTML object>
```