Mental Health Final Project

Group 9: Nathan, Silma, & Wei

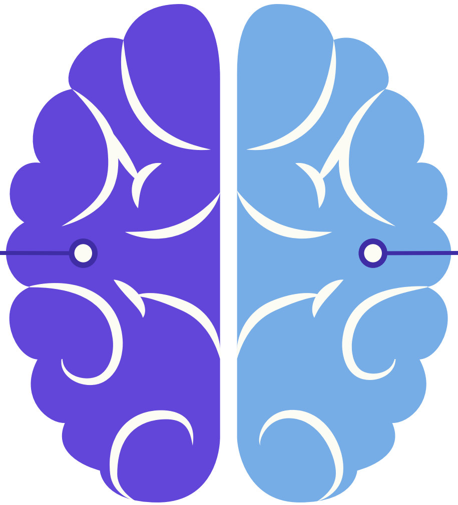
DATA EXPLORATION - Part 1

AMERICA ON TECH

- Get an overview of the data and identify variable types

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 25000 entries, 0 to 24999
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   id                 25000 non-null  int64
 1   gender             25000 non-null  object
 2   age                25000 non-null  int64
 3   major              25000 non-null  object
 4   gpa                25000 non-null  float64
 5   class_status       25000 non-null  object
 6   marital_status     24028 non-null  object
 7   have_depression    24471 non-null  object
 8   have_anxiety       24262 non-null  object
 9   have_panicattacks  24262 non-null  object
 10  seeked_treatment   24471 non-null  object
dtypes: float64(1), int64(2), object(8)
memory usage: 2.1+ MB
```



```
df.describe()
```

	id	age	gpa
count	25000.000000	25000.000000	25000.000000
mean	12500.500000	19.996560	2.904157
std	7217.022701	1.998717	0.635757
min	1.000000	17.000000	1.800000
25%	6250.750000	18.000000	2.360000
50%	12500.500000	20.000000	2.910000
75%	18750.250000	22.000000	3.460000
max	25000.000000	23.000000	4.000000

DATA EXPLORATION - Part 2

- Find mean, median, & mode
- Find std & variance for numerical features

MEAN

```
print("Mean of AGE", np.mean(df['age']))
print("Mean of GPA", np.mean(df['gpa']))
```

```
Mean of AGE 19.99656
Mean of GPA 2.904156799999987
```

MODE

```
print("Mode of AGE",  stats.mode(df['age']))
print("Mode of GPA", stats.mode(df['gpa']))
```

```
Mode of AGE ModeResult(mode=array([18], dtype=int64), count=array([3609]))
Mode of GPA ModeResult(mode=array([3.88]), count=array([147]))
```

MEDIAN

```
print("Median of AGE", np.median(df['age']))
print("Median of GPA", np.median(df['gpa']))
```

```
Median of AGE 20.0
Median of GPA 2.91
```

GPA has a negative direction skewness because mean<median<mode (2.90<2.91<3.88)
AGE has somewhat of a symmetrical skewness since the values are neither greater than or equal in the context of positive and negative direction

DATA EXPLORATION - Part 2 CONT.

AMERICA ON TECH

STANDARD DEVIATION

```
print("AGE STD", np.std(df['age']))
print("GPA STD", np.std(df['gpa']))
```

```
AGE STD 1.9986766037558397
GPA STD 0.635744446309813
```

The low Standard Deviation for both GPA and AGE shows the data is close to the MEAN value

VARIANCE

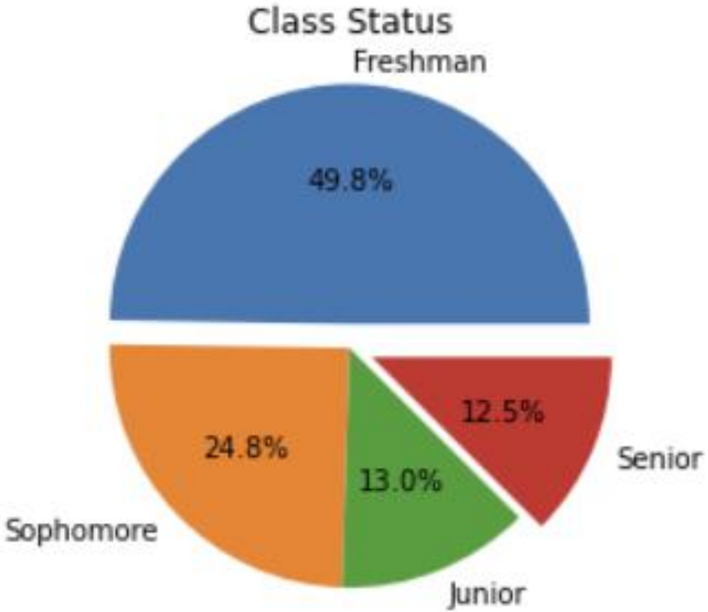
```
print("Variance of AGE", np.var(df['age']))
print("Variance of GPA", np.var(df['gpa']))
```

```
Variance of AGE 3.994708164009776
Variance of GPA 0.4041710010137074
```

The Variance for GPA and AGE is measuring the average degree each point differs from the mean

DATA VISUALIZATION

- To depict the summary statistics of the data
- To get a count of values in a categorical variable

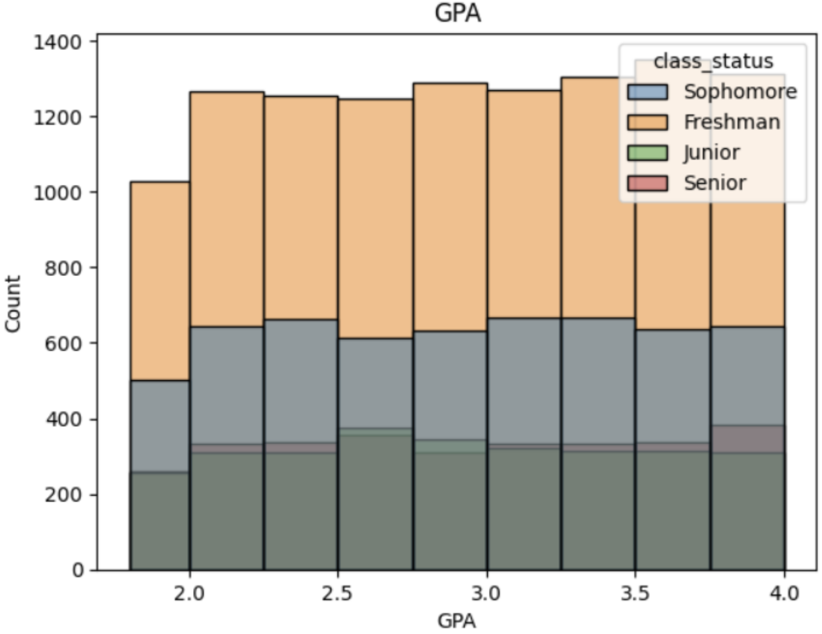



```
##Pie chart of Class Status
labels = ["Freshman", "Sophomore",
          "Junior", "Senior"]
plt.pie(df['class_status'].value_counts(), labels=labels,
        autopct='%1.1f%%',explode=(0.1,0,0,0.1))
plt.title("Class Status")
```

Freshman	49.8%
Sophomore	24.8%
Junior	13.0%
Senior	12.5%

DATA VISUALIZATION CONT.

```
# Histogram
plt.figure(6)
sns.histplot(data=df, x='gpa', bins = [1.8, 2.0, 2.25, 2.5, 2.75, 3, 3.25, 3.5, 3.75, 4], hue='class_status', palette='tab10')
plt.title("GPA")
plt.xlabel('GPA')
plt.ylabel("Count")
plt.show()
```

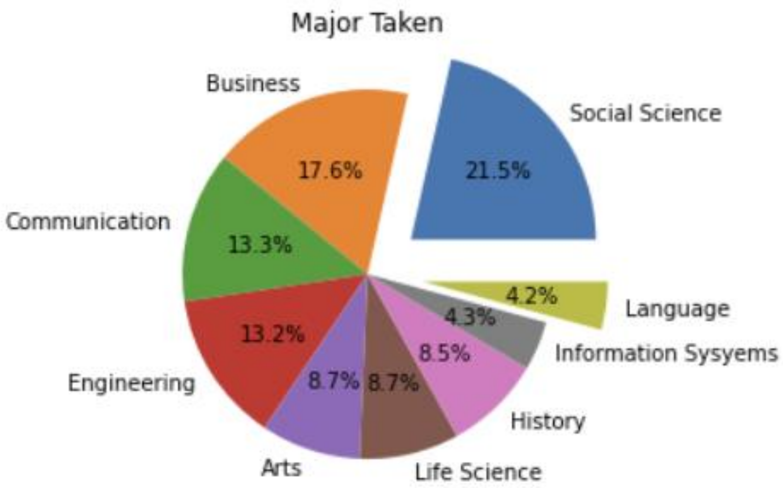


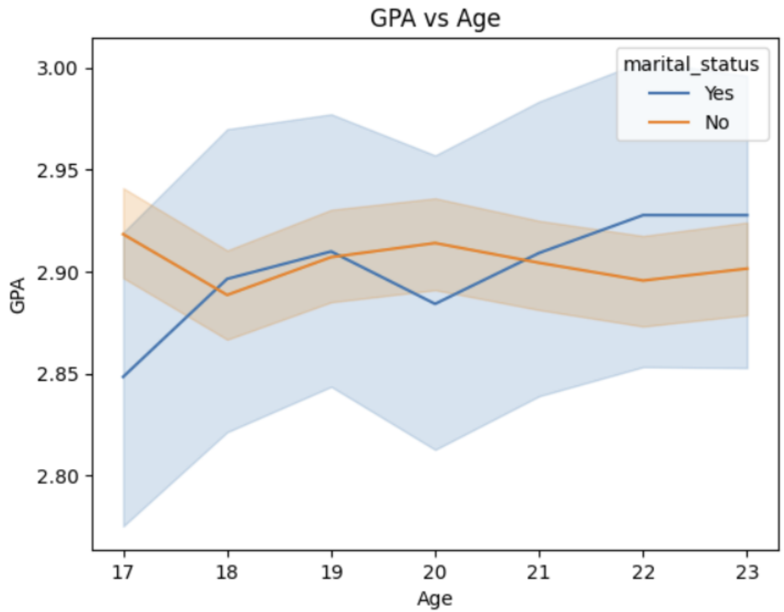
Histogram to depict the variations in Grade Point Averages

DATA VISUALIZATION CONT.

```
##Pie chart of major
labels = ["Social Science","Business",
          "Communication", "Engineering",
          "Arts", "Life Science", "History",
          "Information Sysyems", "Language"]
plt.pie(df['major'].value_counts(), labels=labels, autopct='%1.1f%%',
        explode=(0.3,0,0,0,0,0,0,0,0.3))
plt.title("Major Taken")
```

```
# GPA By Age, Hue 'Marital Status'
plt.figure(5)
sns.lineplot(data=df, x='age', y='gpa', hue='marital_status', palette='tab10')
plt.title("GPA vs Age")
plt.xlabel('Age')
plt.ylabel("GPA")
plt.show()
```

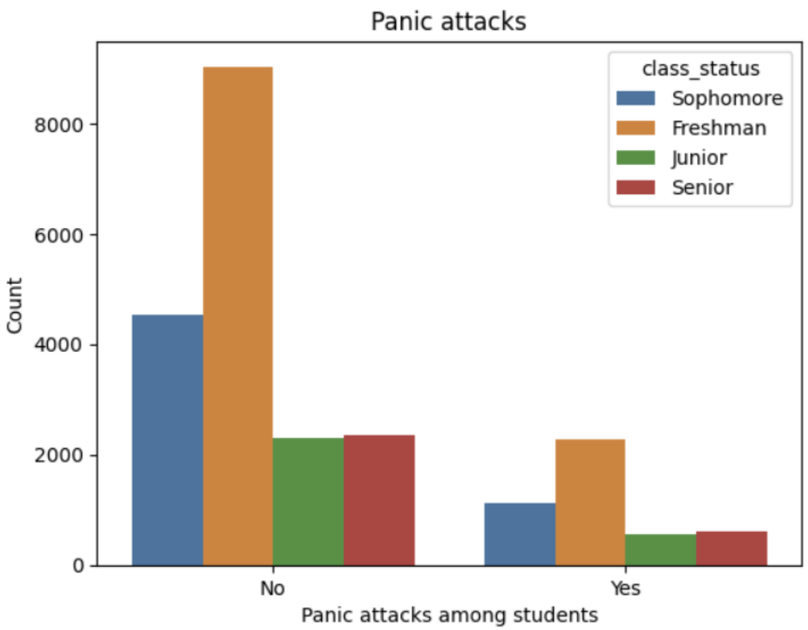


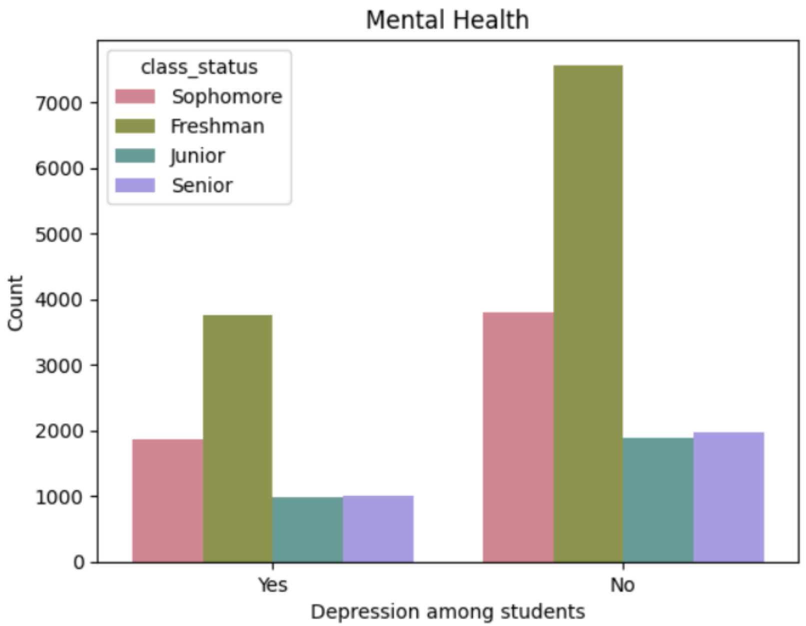


DATA VISUALIZATION CONT.

```
# Panic attacks by class status
plt.figure(1)
sns.countplot(data=df, x='have_panicattacks', hue='class_status', palette='tab10')
plt.title("Panic attacks")
plt.xlabel('Panic attacks among students')
plt.ylabel("Count")
plt.show()
```

```
# Mental health by class status
plt.figure(2)
sns.countplot(data=df, x='have_depression', hue='class_status', palette='husl')
plt.title("Mental Health")
plt.xlabel('Depression among students')
plt.ylabel("Count")
plt.show()
```

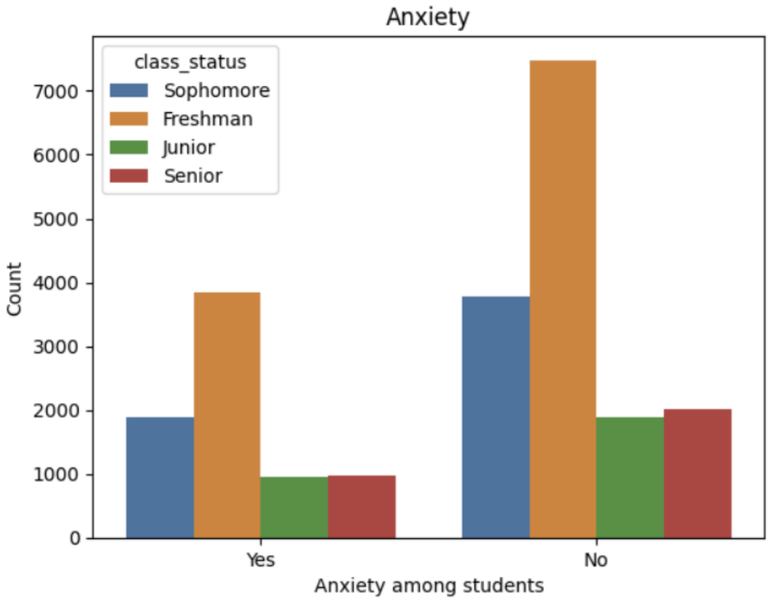


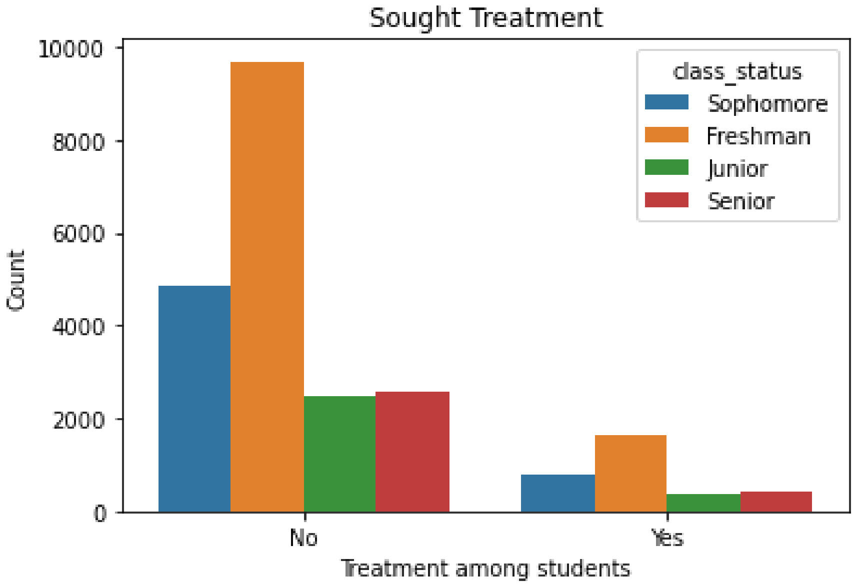


DATA VISUALIZATION CONT.

```
# Anxiety by class status
plt.figure(3)
sns.countplot(data=df, x='have_anxiety', hue='class_status', palette='tab10')
plt.title("Anxiety")
plt.xlabel('Anxiety among students')
plt.ylabel("Count")
plt.show()
```

```
# Sought treatment by class status
plt.figure(4)
sns.countplot(data=df, x='seeked_treatment', hue='class_status', palette='tab10')
plt.title("Sought Treatment")
plt.xlabel('Treatment among students')
plt.ylabel("Count")
plt.show()
```





DATA CURATION - Part 1

AMERICA ON TECH

- Identify outliers and handle missing data

```
# Data Curation
df[df['marital_status'].isnull()]
```

	id	gender	age	major	gpa	class_status	marital_status	have_depression	have_anxiety	have_panicattacks	seeked_treatment
2	3	Male	20	Business	1.96	Junior	NaN	No	No	Yes	Yes
49	50	Female	23	Information Systems	3.99	Freshman	NaN	Yes	No	No	No
65	66	Male	17	Life Sciences	3.81	Sophomore	NaN	Yes	No	No	No
78	79	Female	22	Social Sciences	3.46	Freshman	NaN	No	No	No	No
81	82	Male	22	Communications	1.91	Sophomore	NaN	No	No	No	No
...	...	...	...	...	...	...	...	...	...	...	...
24922	24923	Female	17	Engineering	3.71	Freshman	NaN	Yes	No	No	No
24944	24945	Non-binary	20	Arts	2.46	Junior	NaN	No	No	No	Yes
24966	24967	Male	17	Social Sciences	3.97	Freshman	NaN	No	No	No	No
24980	24981	Female	18	Social Sciences	2.62	Freshman	NaN	No	Yes	No	No
24985	24986	Male	21	Business	2.49	Sophomore	NaN	No	No	No	No

972 rows × 11 columns

```
df[df['have_depression'].isnull()]
```

	id	gender	age	major	gpa	class_status	marital_status	have_depression	have_anxiety	have_panicattacks	seeked_treatment
86	87	Female	23	Social Sciences	2.99	Sophomore	No	NaN	No	No	NaN
112	113	Male	21	Business	3.01	Freshman	No	NaN	No	No	NaN
166	167	Female	17	Information Systems	2.87	Sophomore	No	NaN	No	No	NaN
184	185	Male	21	Language	2.60	Sophomore	NaN	NaN	Yes	No	NaN
185	186	Male	17	Arts	2.25	Sophomore	No	NaN	Yes	No	NaN
...	...	...	...	...	...	...	...	...	...	...	...
24801	24802	Male	19	Business	3.59	Junior	No	NaN	No	Yes	NaN
24828	24829	Male	23	Social Sciences	2.35	Freshman	No	NaN	No	No	NaN
24834	24835	Male	21	Arts	2.48	Junior	No	NaN	No	No	NaN
24991	24992	Female	22	Communications	3.64	Freshman	No	NaN	No	No	NaN
24996	24997	Male	20	Business	1.90	Freshman	No	NaN	Yes	No	NaN

529 rows × 11 columns

```
df[df['have_anxiety'].isnull()]
```

	id	gender	age	major	gpa	class_status	marital_status	have_depression	have_anxiety	have_panicattacks	seeked_treatment
36	37	Male	20	Social Sciences	2.51	Freshman	No	Yes	NaN	NaN	No
152	153	Female	19	Communications	3.48	Sophomore	No	Yes	NaN	NaN	Yes
172	173	Male	23	Communications	1.95	Freshman	No	No	NaN	NaN	No
179	180	Male	22	Social Sciences	2.43	Freshman	No	No	NaN	NaN	No
272	273	Male	17	Communications	3.87	Freshman	No	No	NaN	NaN	No
...	...	...	...	...	...	...	...	...	...	...	...
24710	24711	Male	17	Engineering	3.59	Freshman	No	No	NaN	NaN	No
24720	24721	Male	17	Social Sciences	1.98	Sophomore	No	Yes	NaN	NaN	No
24755	24756	Female	17	Business	2.51	Freshman	No	No	NaN	NaN	No
24829	24830	Male	20	Social Sciences	1.95	Freshman	No	No	NaN	NaN	No
24919	24920	Male	20	Business	2.54	Sophomore	No	No	NaN	NaN	No

738 rows × 11 columns

```
df[df['have_panicattacks'].isnull()]
```

	id	gender	age	major	gpa	class_status	marital_status	have_depression	have_anxiety	have_panicattacks	seeked_treatment
36	37	Male	20	Social Sciences	2.51	Freshman	No	Yes	NaN	NaN	No
152	153	Female	19	Communications	3.48	Sophomore	No	Yes	NaN	NaN	Yes
172	173	Male	23	Communications	1.95	Freshman	No	No	NaN	NaN	No
179	180	Male	22	Social Sciences	2.43	Freshman	No	No	NaN	NaN	No
272	273	Male	17	Communications	3.87	Freshman	No	No	NaN	NaN	No
...	...	...	...	...	...	...	...	...	...	...	...
24710	24711	Male	17	Engineering	3.59	Freshman	No	No	NaN	NaN	No
24720	24721	Male	17	Social Sciences	1.98	Sophomore	No	Yes	NaN	NaN	No
24755	24756	Female	17	Business	2.51	Freshman	No	No	NaN	NaN	No
24829	24830	Male	20	Social Sciences	1.95	Freshman	No	No	NaN	NaN	No
24919	24920	Male	20	Business	2.54	Sophomore	No	No	NaN	NaN	No

738 rows × 11 columns

DATA CURATION - Part 1 CONT.

AMERICA ON TECH

```
df[df['seeked_treatment'].isnull()]
```

	id	gender	age	major	gpa	class_status	marital_status	have_depression	have_anxiety	have_panicattacks	seeked_treatment
86	87	Female	23	Social Sciences	2.99	Sophomore	No	NaN	No	No	NaN
112	113	Male	21	Business	3.01	Freshman	No	NaN	No	No	NaN
166	167	Female	17	Information Systems	2.87	Sophomore	No	NaN	No	No	NaN
184	185	Male	21	Language	2.60	Sophomore	NaN	NaN	Yes	No	NaN
185	186	Male	17	Arts	2.25	Sophomore	No	NaN	Yes	No	NaN
...	...	...	...	...	...	...	...	...	...	...	...
24801	24802	Male	19	Business	3.59	Junior	No	NaN	No	Yes	NaN
24828	24829	Male	23	Social Sciences	2.35	Freshman	No	NaN	No	No	NaN
24834	24835	Male	21	Arts	2.48	Junior	No	NaN	No	No	NaN
24991	24992	Female	22	Communications	3.64	Freshman	No	NaN	No	No	NaN
24996	24997	Male	20	Business	1.90	Freshman	No	NaN	Yes	No	NaN

529 rows × 11 columns

Remove any rows with NaN or NULL values

```
df.dropna(how='any', inplace=True)
```

DATA CURATION - Part 2

- Dummy code all categorical features

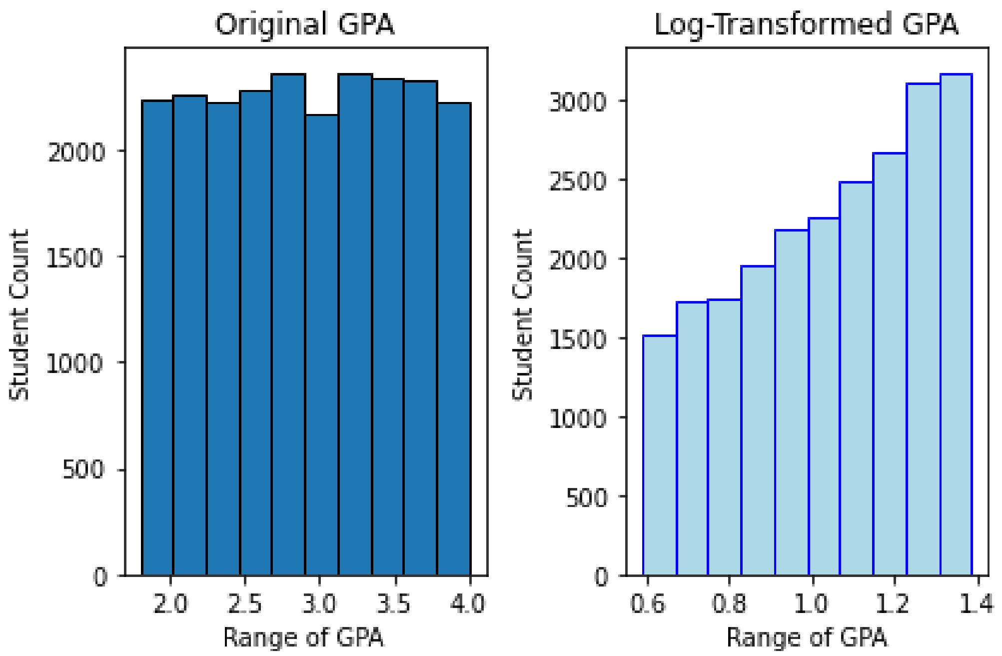
```python
# Dummy Code all categorical features, and deal with outliers

# Gender re mapping
df["gender"] = df["gender"].map({"Female": 0, "Male": 1, "Agender": 2, "Genderqueer": 3, "Bigender": 4, "Genderfluid": 5,
"Non-binary": 6, "Polygender": 7})

# Major re mapping
df["major"] = df["major"].map({'Social Sciences': 0, 'Business': 1, 'Communications': 2, 'Engineering': 3, 'Arts': 4,
                               'Life Sciences': 5,'History': 6, 'Language': 7, 'Information Systems': 8})

# Class status re mapping
df["class_status"] = df["class_status"].map({'Freshman': 0, 'Sophomore': 1, 'Junior': 2, 'Senior': 3})

# Marital status re mapping
df["marital_status"] = df["marital_status"].map({'No': 0, 'Yes': 1})

# Has depression re mapping
df["have_depression"] = df["have_depression"].map({'No': 0, 'Yes': 1})

# Has anxiety re mapping
df["have_anxiety"] = df["have_anxiety"].map({'No': 0, 'Yes': 1})

# Has panic attacks re mapping
df["have_panicattacks"] = df["have_panicattacks"].map({'No': 0, 'Yes': 1})

# Has sought treatment re mapping
df["seeked_treatment"] = df["seeked_treatment"].map({'No': 0, 'Yes': 1})

# Display the dataframe but now with all numerical or ordinal values
df.head()
```

	id	gender	age	major	gpa	class_status	marital_status	have_depression	have_anxiety	have_panicattacks	seeked_treatment
0	1	0	19	0	2.95	1	1	1	1	0	0
1	2	1	18	0	3.61	1	0	0	0	0	0
3	4	2	20	2	3.98	0	0	1	1	0	0
4	5	1	22	1	2.53	1	0	0	0	1	0
5	6	0	17	2	2.20	0	0	1	0	0	0

Feature Engineering - Part 1

- Is the use of domain knowledge and data transformation to extract features from raw data
- Log transformation to transform skewed gpa to normality

```
# Complete log transformations for the gpa column.
gpa = df['gpa'].values.tolist()
gpa_log = np.log(df['gpa'])

#define grid of plots
fig, axs = plt.subplots(nrows=1, ncols=2)

#create histograms
axs[0].hist(gpa, edgecolor='black')
axs[1].hist(gpa_log, color="lightblue", edgecolor='blue')

#add title to each histogram
axs[0].set_title('Original GPA')
axs[0].set_xlabel('Range of GPA')
axs[0].set_ylabel('Student Count')
axs[1].set_title('Log-Transformed GPA')
axs[1].set_xlabel('Range of GPA')
axs[1].set_ylabel('Student Count')

# sets the proper spacing between subplots
fig.tight_layout()
```

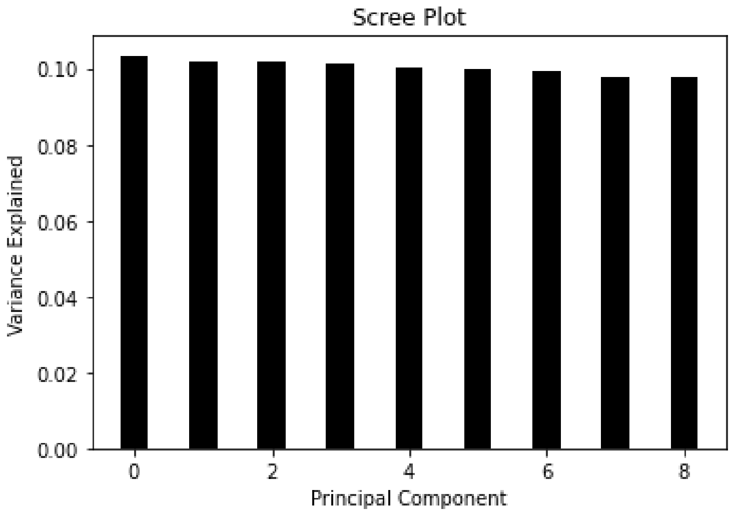


Feature Engineering - Part 2

- Principal Component Analysis

```python
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

dfPart = df.loc[:, df.columns != 'have_depression']
ArraySS = StandardScaler().fit_transform(dfPart)

#****************************************
# Perform PCA
#n_components is the amount of components (features) (dimension) you want to kee

# 9 features
pca = PCA(n_components = 9)
PCAData = pca.fit_transform(ArraySS)


#****************************************
# Explained Variance
print("\n")
print("PCA Explained Variance Chart")
print(pca.explained_variance_ratio_)
print("\n")

#****************************************
# Scree plot
# To display the explained variance of each feature
PC_values = np.arange(pca.n_components_)

plt.figure(1)
plt.bar(PC_values, pca.explained_variance_ratio_, width=0.4, color='black')
plt.title('Scree Plot')
plt.xlabel('Principal Component')
plt.ylabel('Variance Explained')
plt.show()
```

```
PCA Explained Variance Chart
[0.10345943 0.10195974 0.10166295 0.10138352 0.10050592 0.09975313
 0.09916295 0.09795297 0.09791569]
```



Machine Learning Model - Part 1

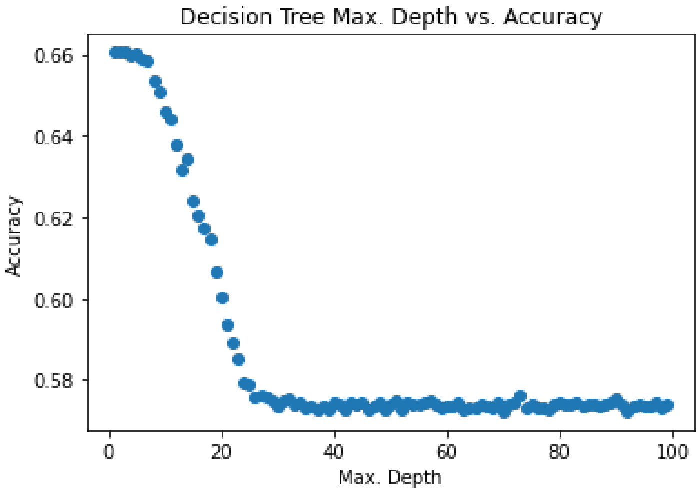
AMERICA ON TECH

- Supervised Learning Model - Decision Tree

```python
# Decision Tree classification

# Import necessary libraries for train/test split, classification, and results displaying
from sklearn import tree
from sklearn.model_selection import train_test_split
from mlxtend.plotting import plot_decision_regions
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB
from IPython.core.pylabtools import figsize
from sklearn.metrics import accuracy_score
```

```python
# Split data into train/test set with 70/30 split
TrainData, TestData = train_test_split(df, test_size=0.3, shuffle=True)

# Split the X and Y features
x_features = ['gender', 'age', 'gpa', 'class_status', 'marital_status', 'have_anxiety', 'have_panicattacks', 'seeked_treatment']

# # Populate the train and test data X and Y lists
TrainDataX = TrainData[x_features]
TrainDataY = TrainData['have_depression']

TestDataX = TestData[x_features]
TestDataY = TestData['have_depression']

acc_list = []
n_list = list(range(1,100))

for i in n_list:
    model = tree.DecisionTreeClassifier(criterion="gini", min_samples_split=10, max_depth=i)
    model.fit(TrainDataX, TrainDataY)

    # Now with the training data fitted, it is time to let the test data make its own predictions based on the results of the
    # train data model fit
    PredictY = model.predict(TestDataX)
    acc = accuracy_score(TestDataY, PredictY)
    acc_list.append(acc)


plt.scatter(n_list, acc_list)
plt.xlabel("Max. Depth")
plt.ylabel("Accuracy")
plt.title("Decision Tree Max. Depth vs. Accuracy")
```

```
Text(0.5, 1.0, 'Decision Tree Max. Depth vs. Accuracy')
```



Machine Learning Model - Part 2

AMERICA ON TECH

- Evaluation of the model

```
# Decision Tree Classification - train the classifier with the train data set
# Entropy measures the uncertainty or impurity, of a node
# The more entropy a node has, the more uncertain the outcome (Yes or No for example) is to be

# Decision Tree Classification - train the classifier with the train data set
model = tree.DecisionTreeClassifier(max_depth=180)
model.fit(TrainDataX, TrainDataY)


#************************************
# Predict the response for test data set
PredictY = model.predict(TestDataX)

# Confusion matrix to differentiate between true positive and negative(TP TN), and false positive and negative
CM = confusion_matrix(TestDataY, PredictY)
print("Confusion Matrix...")
print(CM)

# Print classification report and confusion matrix
CR = classification_report(TestDataY,PredictY, target_names=['Does not have depression', 'Has Depression'])
print("Clasification Report...")
print(CR)

# other models
# Naive bayes
```

```
Confusion Matrix...
[[3039 1482]
 [1560  763]]
Clasification Report...
```

	precision	recall	f1-score	support
Does not have depression	0.66	0.67	0.67	4521
Has Depression	0.34	0.33	0.33	2323
accuracy			0.56	6844
macro avg	0.50	0.50	0.50	6844
weighted avg	0.55	0.56	0.55	6844

```
# Plot the area under ROC curve
from sklearn import metrics as mt

#Plot the Area under the ROC curve
plt.figure(0)
# y_true, y_score
mt.roc_auc_score(np.array(TestDataY), np.array(PredictY))
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```

THANK YOU!!