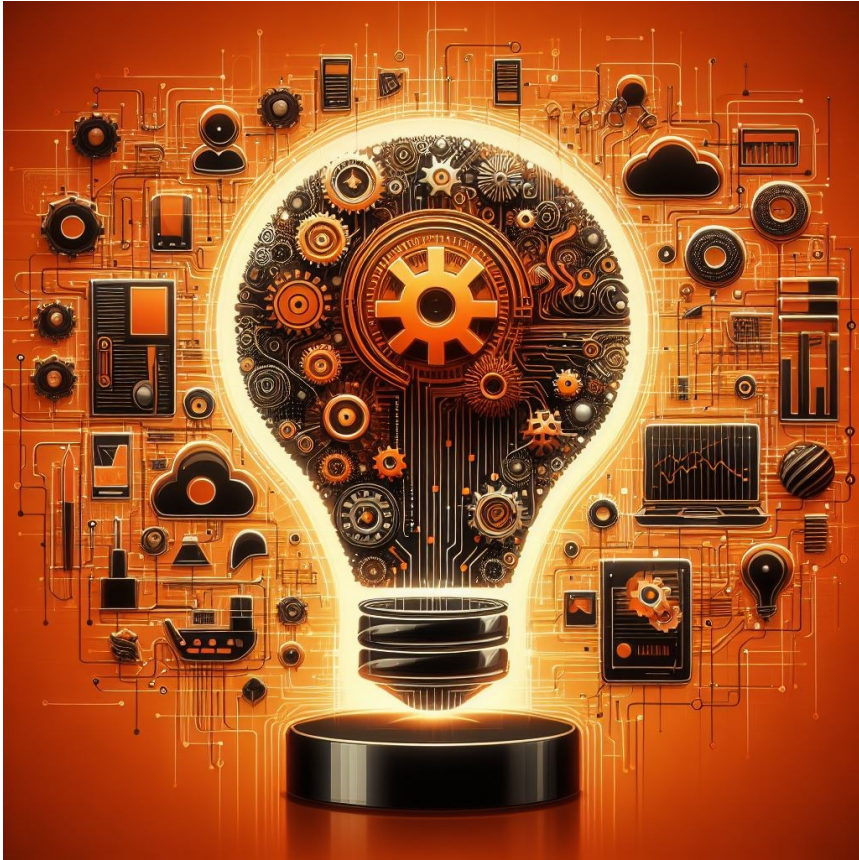


# Generative AI in Higher Ed:

## AI & Python for Intelligent IR Progress

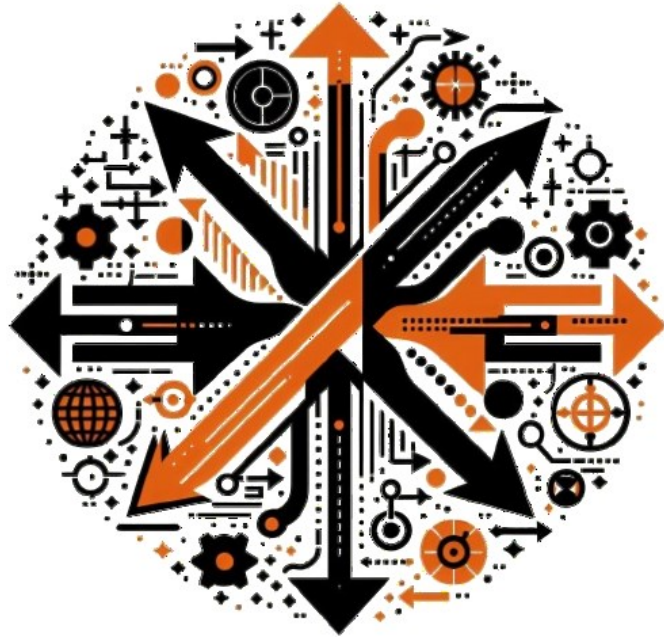
Tara L. Webb, Ph.D.  
April 2024



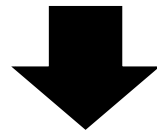
## Learning outcome

Understand how to use AI  
to code simple & complex  
IR tasks

# Using AI for simple to complex Python tasks



Simple | Calculations



| Summary Tables

Complex | Machine Learning

Important first steps - Free!

Create ChatGPT account

Download Python

Install Jupyter Notebook



# Common Data Set Table B1

	FULL-TIME			PART-TIME		
	Men	Women	Another Gender	Men	Women	Another Gender
<b>Undergraduate Students</b>		<b>Undergraduate Students</b>				
Degree-seeking, first-time, first-year students						
Other first-year, degree-seeking students						
All other degree-seeking undergraduate students						
<b>Total degree-seeking undergraduate students</b>	0	0	0	0	0	0
All other undergraduates enrolled in credit courses						
<b>Total Undergraduate Students</b>	0	0	0	0	0	0
<b>Graduate Students</b>		<b>Graduate Students</b>				
Degree-seeking, first-time						
All other degree-seeking						
All other graduates enrolled in credit courses						
<b>Total Graduate Students</b>	0	0	0	0	0	0
<b>Total All Students</b>	0	0	0	0	0	0

Source: [Home - Common Data Set](#)



# Common Data Set prompt

Be specific!

How to calculate values

Table structure



TA

**You**

I have Jupyter Notebook open and want to use Python to create a Common Data Set table. My spreadsheet contains one row per student with the following columns: ID, Student Federal Cohort Group, Person Gender, Load, Person IPEDS Race/Ethnicity Ctgy, Reg. Units for Term, Completed Units, Unit Standing, On Study Away / Study Abroad?, and more.

For all calculations, students with a value of "Y" in the 'On Study Away / Study Abroad?' column should be excluded.

I need to create a pivot table with columns by load status of full-time (Load value = O or F) and part-time (Load value = P) and by Person Gender (M = Men, F = Women).

The rows of the pivot table will be 1) "Degree-seeking, first-time" which includes students with a Student Federal Cohort Group of 2023FT,

2) "Other first-year, degree-seeking" which includes all other students with a Completed Units value less than 9 from any Student Federal Cohort Group except 2023FT or NFC,

3) "All other degree-seeking" which includes all students with a Student Federal Cohort Group except 2023FT, NFC, or any student with a Completed Units value less than 9,

4) "All other undergraduates enrolled" includes students with a Student Federal Cohort Group value of NFC,

Write the Python code that will create the calculations and requested table.



ChatGPT

To create the desired pivot table in Python using pandas, you can follow these steps:

1. Load your spreadsheet data into a pandas DataFrame.
2. Exclude students with "Y" in the 'On Study Away / Study Abroad?' column.
3. Create a new column for load status based on the 'Load' column.
4. Create a new column for student category based on the conditions mentioned.
5. Use the pandas pivot\_table function to create the pivot table with the required columns and rows.

Here's an example code snippet to achieve this:

```
python
import pandas as pd

# Load your spreadsheet data into a pandas DataFrame
# Replace 'file_path' with the actual path to your spreadsheet file
file_path = 'path_to_your_spreadsheet.xlsx'
df = pd.read_excel(file_path)

# Exclude students with "Y" in the 'On Study Away / Study Abroad?' column
df_filtered = df[df['On Study Away / Study Abroad?'] != 'Y']
```

**You**  
Add code to rename Person Gender M to Men and Person Gender F to Women. Also reorder the pivot table rows so degree-seeking, first-time comes first, then other first-year, degree-seeking, then all other degree-seeking, then all other undergraduates enrolled.

**You**  
This worked but only after I removed "\_filtered" from all df\_filtered, so rewrite the code that way please



ChatGPT

Certainly! Here's the rewritten code without the "\_filtered" suffix in the DataFrame variable names:

```
python
import pandas as pd

# Load your spreadsheet data into a pandas DataFrame
# Replace 'file_path' with the actual path to your spreadsheet file
file_path = 'path_to_your_spreadsheet.xlsx'
df = pd.read_excel(file_path)

# Exclude students with "Y" in the 'On Study Away / Study Abroad?' column
df = df[df['On Study Away / Study Abroad?'] != 'Y']

# Create a new column for load status based on the 'Load' column
df['Load_Status'] = df['Load'].apply(lambda x: 'Full-time' if x in ['0', 'F'] else 'Part-time')

# Create a new column for student category based on the conditions mentioned
def categorize_student(row):
    if row['Student Federal Cohort Group'] == '2023FT':
        return 'Degree-seeking, first-time'
    elif row['Completed Units'] < 9 and row['Student Federal Cohort Group'] not in ['2023FT', '2023FT']:
        return 'Other first-year, degree-seeking'
    elif row['Completed Units'] >= 9 and row['Student Federal Cohort Group'] not in ['2023FT', '2023FT']:
        return 'All other degree-seeking'
    elif row['Student Federal Cohort Group'] == 'NFC':
        return 'All other undergraduates enrolled'
    else:
        return 'Other'

df['Student_Category'] = df.apply(categorize_student, axis=1)

# Rename 'Person Gender' values
df['Person Gender'] = df['Person Gender'].replace({'M': 'Men', 'F': 'Women'})

# Create the pivot table
pivot_table = pd.pivot_table(df, values='ID', index=['Student_Category'], columns=['Load_Status'])

# Reorder the pivot table rows
pivot_table = pivot_table.reindex(['Degree-seeking, first-time', 'Other first-year, degree-seeking', 'All other degree-seeking', 'All other undergraduates enrolled', 'Other'])
```

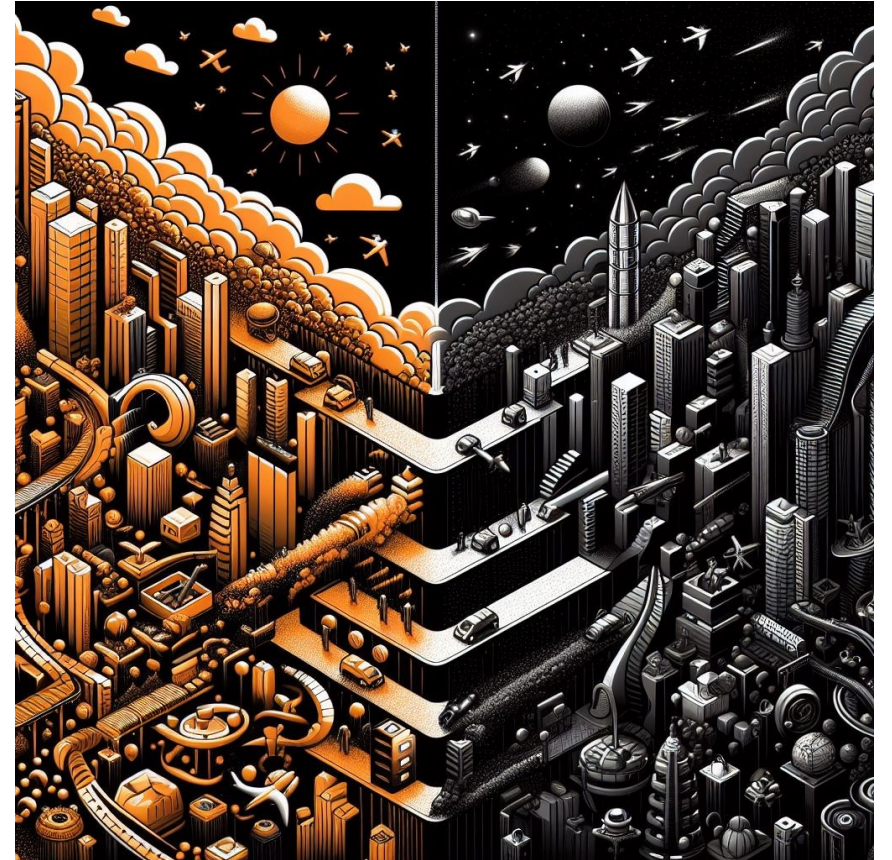


Enter code in Jupyter

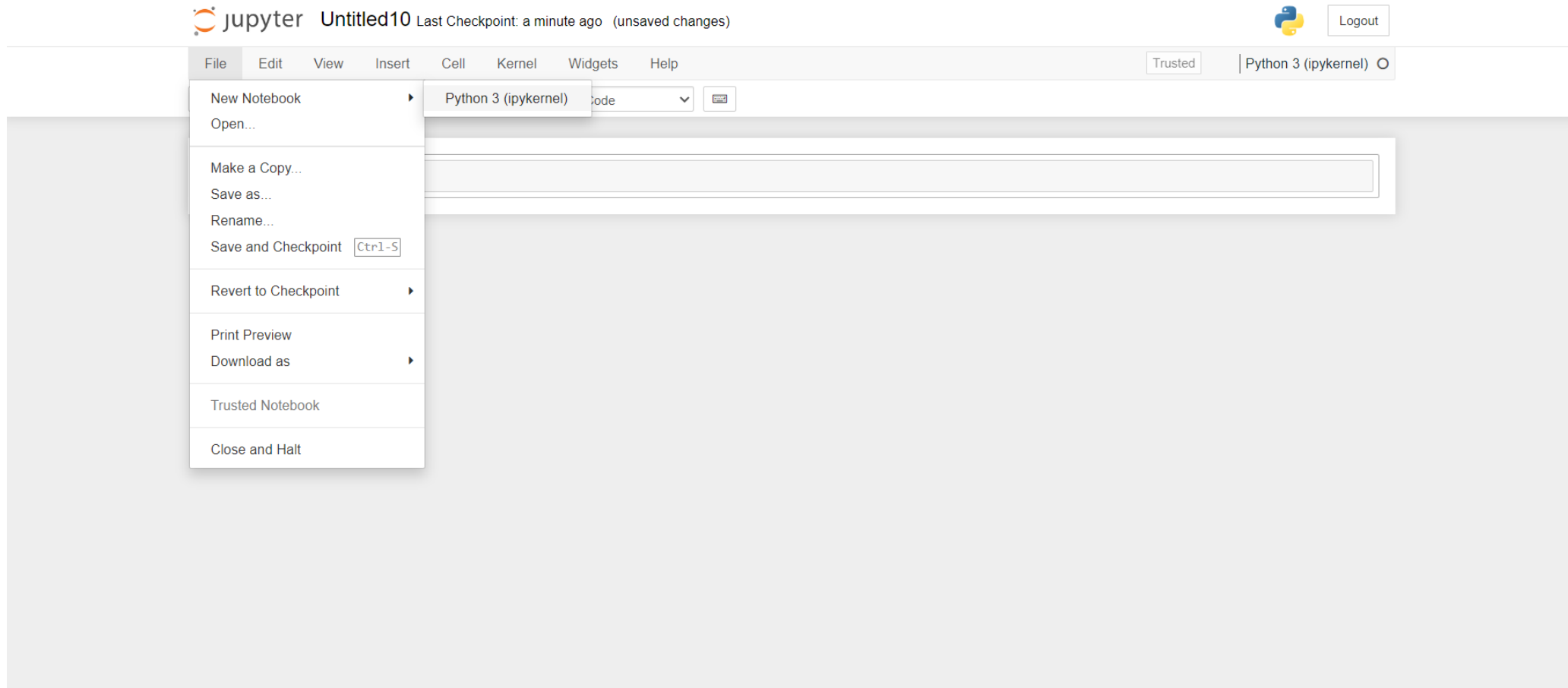
Snippets at a time

Enter the #notes too

Save code for future use



# Start a new notebook in Jupyter Notebook



# Final lines and output in Jupyter Notebook

```
▶ # Rename 'Person Gender' values
df['Person Gender'] = df['Person Gender'].replace({'M': 'Men', 'F': 'Women'})
```

```
▶ # Create the pivot table
pivot_table = pd.pivot_table(df, values='ID', index=['Student_Category'], columns=['Load Status', 'Person Gender'], aggfunc='sum')
```

```
▶ # Reorder the pivot table rows
pivot_table = pivot_table.reindex(['Degree-seeking, first-time', 'Other first-year, degree-seeking', 'All other degree-seeking', 'All other undergraduates enrolled'])
```

```
▶ # Display the pivot table
print(pivot_table)
```

Load Status	Full-time		Part-time	
Person Gender	Men	Women	Men	Women
Degree-seeking, first-time	161	208	0	1
Other first-year, degree-seeking	29	21	1	1
All other degree-seeking	316	428	5	4
All other undergraduates enrolled	8	13	0	0

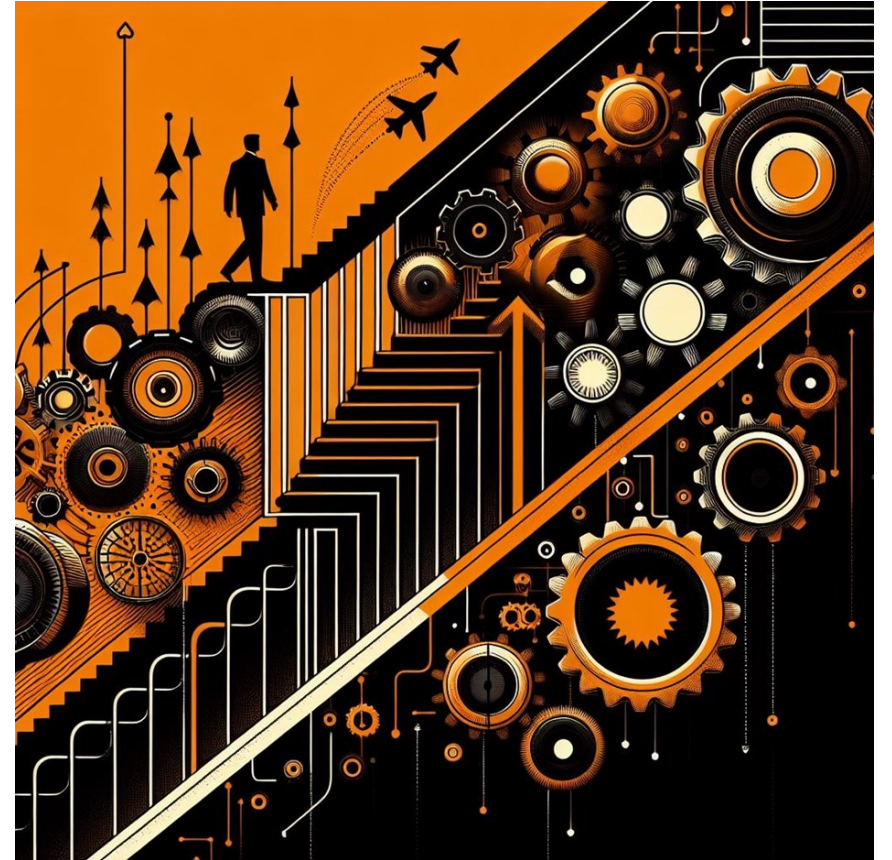
\* Remember to save for future use!

# Machine Learning prompt

Be specific!

Column names, features

Predictors, outcome



# ChatGPT prompt to start Python ML process

TA You

Let's complete machine learning analyses in Python on a dataset for academic retention forecasting. The dataset includes one categorical column (Person First Generation populated with Y or N values) and the following numerical columns: FYS GPA, 2022FA DFWNC, 2022FA FWNC, FY DFWNC, FY FWNC, 2022FA GPA, FY GPA, FTF Retention (binary, 0s and 1s), Moodle Max, and Moodle Avg. The FYS GPA column contains a few blank values that need to be addressed prior to analysis. The predictor variables include all but the FTF Retention variable which will be the criterion variable.

Write the required Python code to load, preprocess, split, train, and evaluate the appropriate machine learning models.





## ChatGPT

To complete machine learning analyses in Python for academic retention forecasting, we'll follow these steps:

1. Load the dataset.
2. Preprocess the data (handling missing values, encoding categorical variables).
3. Split the data into training and testing sets.
4. Train machine learning models (Logistic Regression, Random Forest, and Gradient Boosting).
5. Evaluate the models using appropriate metrics.

Let's start by importing the necessary libraries and loading the dataset:

python

Copy code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv('your_dataset.csv')

# Display the first few rows of the dataset to understand its structure
print(data.head())
```

python

Copy code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report

# Load the dataset
data = pd.read_csv('your_dataset.csv')

# Fill missing values in FYS GPA column with the mean
data['FYS GPA'].fillna(data['FYS GPA'].mean(), inplace=True)

# Encode categorical variable 'Person First Generation' (Y/N) using LabelEncoder
encoder = LabelEncoder()
data['Person First Generation'] = encoder.fit_transform(data['Person First Generation'])

# Separate predictor variables (X) and criterion variable (y)
X = data.drop(['FTF Retention'], axis=1)
y = data['FTF Retention']

# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Scale the numerical features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Initialize and train the Logistic Regression model with scaled data
log_reg_model = LogisticRegression(max_iter=1000, random_state=42)
log_reg_model.fit(X_train_scaled, y_train)

# Make predictions on the scaled test set
y_pred_log_reg = log_reg_model.predict(X_test_scaled)

# Evaluate the model
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
print("Logistic Regression Accuracy: ", accuracy_log_reg)
```

# Initial lines and output in Jupyter Notebook

```
In [ ]: import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score, classification_report
```

```
In [ ]: file_path =
```

```
In [ ]: data = pd.read_excel(file_path)
```

```
In [ ]: # Display the first few rows of the dataset to understand its structure
print(data.head())
```

```
Student ID Person First Generation FYS GPA 2022FA DFwNC 2022FA FwNC
0
1
2
3
4
```

```
FY DFwNC FY FwNC 2022FA GPA FY GPA FTF Retention Moodle Max
0
1
2
3
4
```

```
Moodle Avg
0
1
2
3
4
```

```
In [ ]: # Fill missing values in FYS GPA column with the mean
data['FYS GPA'].fillna(data['FYS GPA'].mean(), inplace=True)
```

# More lines and output in Jupyter Notebook

```
# Encode categorical variable 'Person First Generation' (Y/N) using LabelEncoder
encoder = LabelEncoder()
data['Person First Generation'] = encoder.fit_transform(data['Person First Generation'])
```

```
# Separate predictor variables (X) and criterion variable (y)
X = data.drop(['FTF Retention', 'Student ID'], axis=1)
y = data['FTF Retention']
```

```
# Split the data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
# Scale the numerical features using StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
# Initialize and train the Logistic Regression model with scaled data
log_reg_model = LogisticRegression(max_iter=1000, random_state=42)
log_reg_model.fit(X_train_scaled, y_train)
```

```
! LogisticRegression
LogisticRegression(max_iter=1000, random_state=42)
```

```
# Make predictions on the scaled test set
y_pred_log_reg = log_reg_model.predict(X_test_scaled)
```

```
# Evaluate the model
accuracy_log_reg = accuracy_score(y_test, y_pred_log_reg)
print("Logistic Regression Accuracy:", accuracy_log_reg)
```

Logistic Regression Accuracy: XXXXXXXXXX

\* Remember to save for future use!

TA You

Now, I want to create a plot that shows the variables in order of importance to the model. I also want to create an ROC curve for the models. Write code for both of these.

ChatGPT

Sure, I can help with that. Below is the code to create a plot showing the variables in order of importance for the Random Forest model, as well as code to create an ROC curve for both the Logistic Regression and Random Forest models:

#### 1. Plotting Variable Importance for Random Forest:

```
python Copy code

import matplotlib.pyplot as plt

# Get feature importances from the trained Random Forest model
feature_importances = rf_model.feature_importances_

# Get column names of the features
feature_names = X.columns

# Create a DataFrame to store feature importances and their corresponding names
feature_importances_df = pd.DataFrame({'Feature': feature_names, 'Importance': feature_importances})

# Sort the features by importance in descending order
feature_importances_df = feature_importances_df.sort_values(by='Importance', ascending=False)

# Plotting the variable importance
plt.figure(figsize=(10, 6))
plt.barh(feature_importances_df['Feature'], feature_importances_df['Importance'])
plt.xlabel('Importance')
plt.title('Variable Importance - Random Forest')
plt.gca().invert_yaxis()
plt.show()
```

#### 1. Creating ROC Curves for Logistic Regression and Random Forest:

```
python Copy code

from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Compute predicted probabilities for both models
y_prob_log_reg = log_reg_model.predict_proba(X_test_scaled)[:, 1]
y_prob_rf = rf_model.predict_proba(X_test_scaled)[:, 1]

# Compute ROC curve and AUC for Logistic Regression
fpr_log_reg, tpr_log_reg, _ = roc_curve(y_test, y_prob_log_reg)
roc_auc_log_reg = roc_auc_score(y_test, y_prob_log_reg)

# Compute ROC curve and AUC for Random Forest
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)
roc_auc_rf = roc_auc_score(y_test, y_prob_rf)

# Plotting ROC curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_log_reg, tpr_log_reg, label=f'Logistic Regression (AUC = {roc_auc_log_reg:.2f})')
plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {roc_auc_rf:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve')
plt.legend()
plt.grid(True)
plt.show()
```

In the code above:

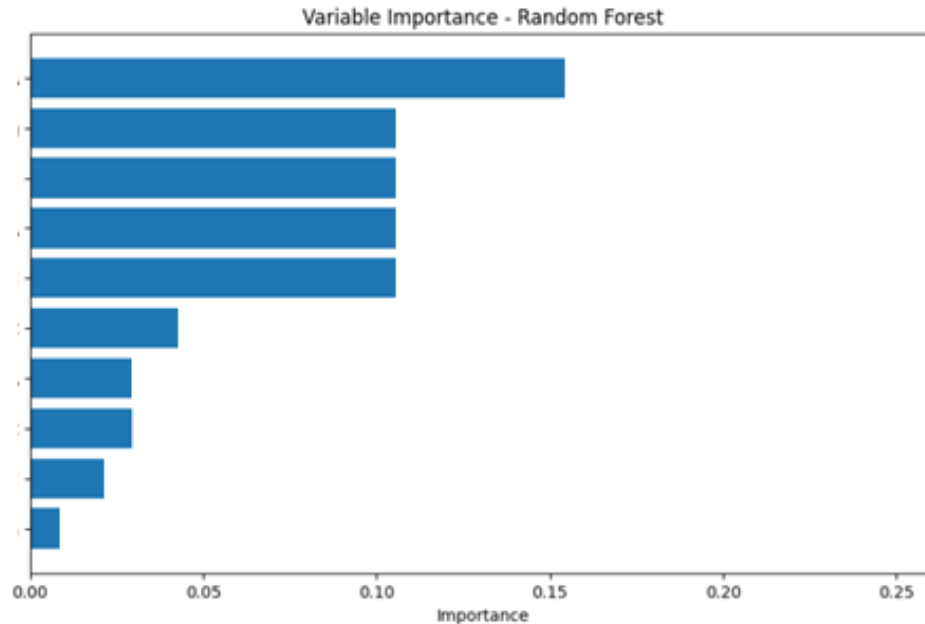
- The first part creates a bar plot showing the variable importance for the Random Forest model.
- The second part computes the ROC curves and AUC scores for both the Logistic Regression and Random Forest models, and then plots the ROC curves with their respective AUC scores.

You can run both sections of code after training your models to visualize the variable importance and ROC curves. Adjust the plot settings as needed for better visualization.

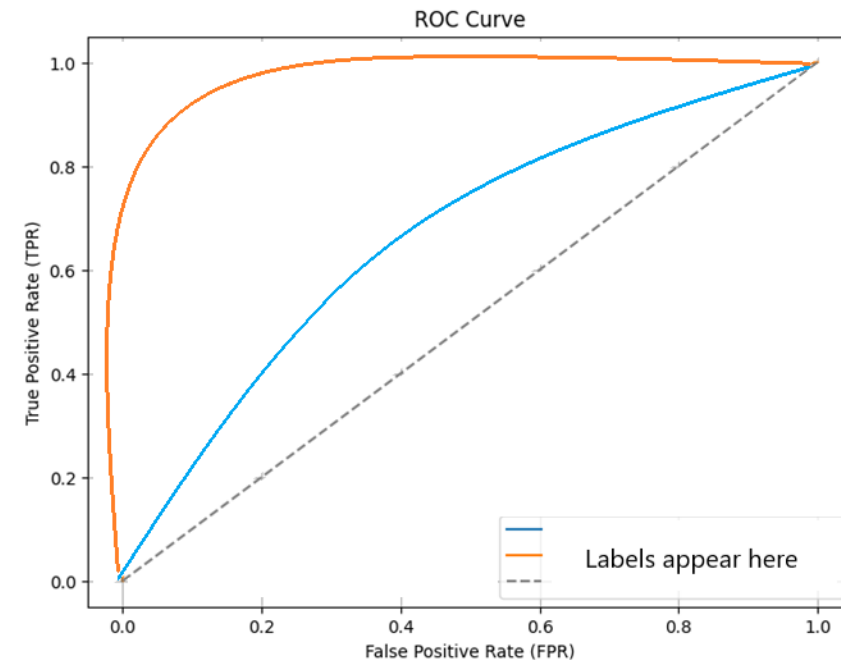
# More lines and output in Jupyter Notebook

```
# Plotting the variable importance
plt.figure(figsize=(10, 6))
plt.barh(feature_importances_df['Feature'], feature_importances_df['Importance'])
plt.xlabel('Importance')
plt.title('Variable Importance - Random Forest')
plt.gca().invert_yaxis()
plt.show()
```

Variable  
names  
would  
appear  
here



```
# Plotting ROC curves
plt.figure(figsize=(8, 6))
plt.plot(fpr_log_reg, tpr_log_reg, label=f'Logistic Regression (AUC = {roc_auc_log_reg:.2f})')
plt.plot(fpr_rf, tpr_rf, label=f'Random Forest (AUC = {roc_auc_rf:.2f})')
plt.plot([0, 1], [0, 1], linestyle='--', color='gray', label='Random Guess')
plt.xlabel('False Positive Rate (FPR)')
plt.ylabel('True Positive Rate (TPR)')
plt.title('ROC Curve')
plt.legend()
plt.grid(True)
plt.show()
```





Pros outweigh cons!

Enhanced IR efficiency

Iterative teaching tool

Imperfect but helpful

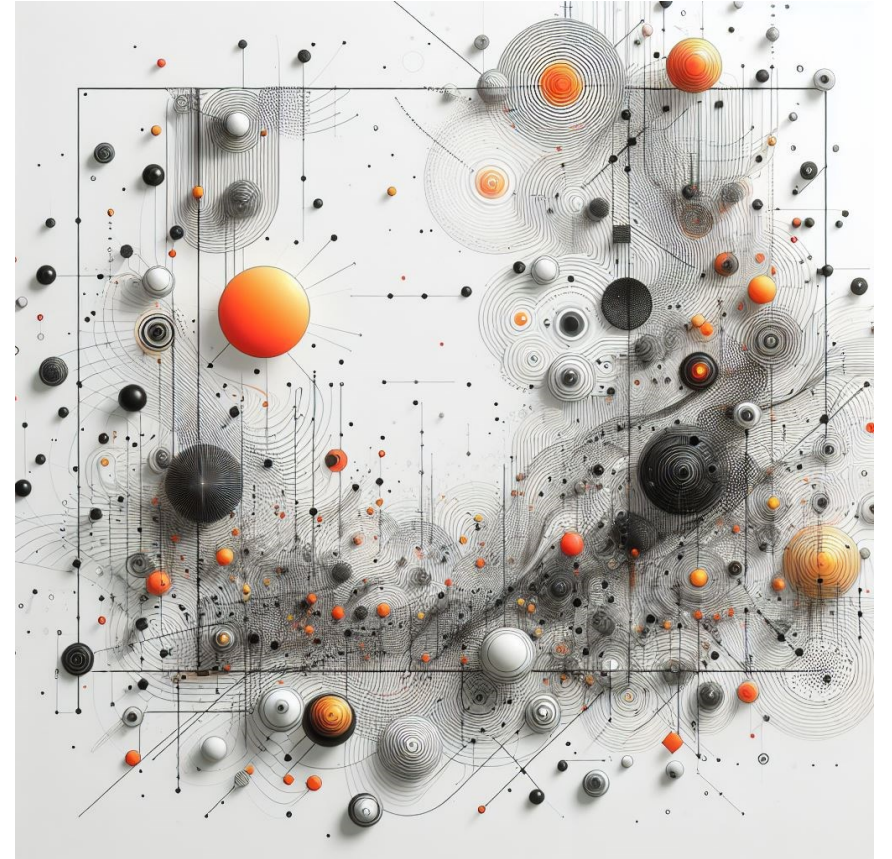


# Other AI versions?

Copilot similar output

Gemini similar output

All imperfect but helpful



# Thanks for listening!

# Questions? Comments?

tara.webb@kzoo.edu



Note: All images in this presentation were generated with AI assistance