project1-2

October 22, 2024

[1]:
```
import pandas as pd
import numpy as np
```

[2]:
```
data=pd.read_csv(r'C:\Users\rohit\Desktop\AI &ML Project\MBA1.csv')
```

[3]:
```
data.head()
```

[3]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_industry	admission
0	5244	Male	True	3.13	Business	NaN	590.0	5.0	Investment Banking	Reject
1	3511	Female	True	3.31	Humanities	NaN	610.0	5.0	Technology	Reject
2	2154	Male	False	3.17	Humanities	Black	710.0	4.0	Consulting	Reject
3	2703	Female	False	3.08	Business	White	590.0	6.0	Other	Reject
4	5867	Male	False	3.24	Humanities	Black	660.0	3.0	Other	Reject

[4]:
```
data.head(10)
```

[4]:

	application_id	gender	international	gpa	major	race	gmat	work_exp	work_industry	admission
0	5244	Male	True	3.13	Business	NaN	590.0	5.0	Investment Banking	Reject
1	3511	Female	True	3.31	Humanities	NaN	610.0			
2	2154	Male	False	3.17	Humanities	Black	710.0			
3	2703	Female	False	3.08	Business	White	590.0			
4	5867	Male	False	3.24	Humanities	Black	660.0			
5	3282	Male	False	3.25	Humanities	Black	620.0			
6	4256	Female	False	3.21	Humanities	White	630.0			
7	443	Female	False	3.25	STEM	White	620.0			
8	5477	Male	False	3.22	Humanities	White	660.0			
9	471	Male	False	3.37	Humanities	Hispanic	720.0			

```
1       5.0          Technology    Reject
2       4.0          Consulting    Reject
3       6.0               Other    Reject
4       3.0               Other    Reject
5       7.0         Health Care    Reject
6       6.0          Technology    Reject
7       4.0          Technology     Admit
8       5.0          Technology     Admit
9       6.0       Nonprofit/Gov     Admit
```

[5]:
```
data.tail()
```

[5]:
```
      application_id  gender  international   gpa       major      race  \
3089            1641    Male          False  2.90    Business  Hispanic
3090            4275    Male          False  3.25  Humanities     Other
3091            4402    Male          False  3.31    Business     Asian
3092            5122  Female          False  3.38    Business     White
3093            3185  Female          False  3.23  Humanities     Other

       gmat  work_exp       work_industry admission
3089  570.0       6.0               Other    Reject
3090  720.0       4.0  Investment Banking     Admit
3091  740.0       6.0  Financial Services     Admit
3092  740.0       5.0          Consulting     Admit
3093  670.0       6.0          Consulting     Admit
```

[6]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3094 entries, 0 to 3093
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   application_id  3094 non-null   int64
 1   gender          3094 non-null   object
 2   international   3094 non-null   bool
 3   gpa             3094 non-null   float64
 4   major           3094 non-null   object
 5   race            2262 non-null   object
 6   gmat            3094 non-null   float64
 7   work_exp        3094 non-null   float64
 8   work_industry   3094 non-null   object
 9   admission       3094 non-null   object
dtypes: bool(1), float64(3), int64(1), object(5)
memory usage: 220.7+ KB
```

[7]:
```
data.isnull().sum()
```

[7]:
```
application_id      0
gender              0
international       0
gpa                 0
major               0
race              832
gmat                0
work_exp            0
work_industry       0
admission           0
dtype: int64
```

[8]:
```
data.describe()
```

[8]:

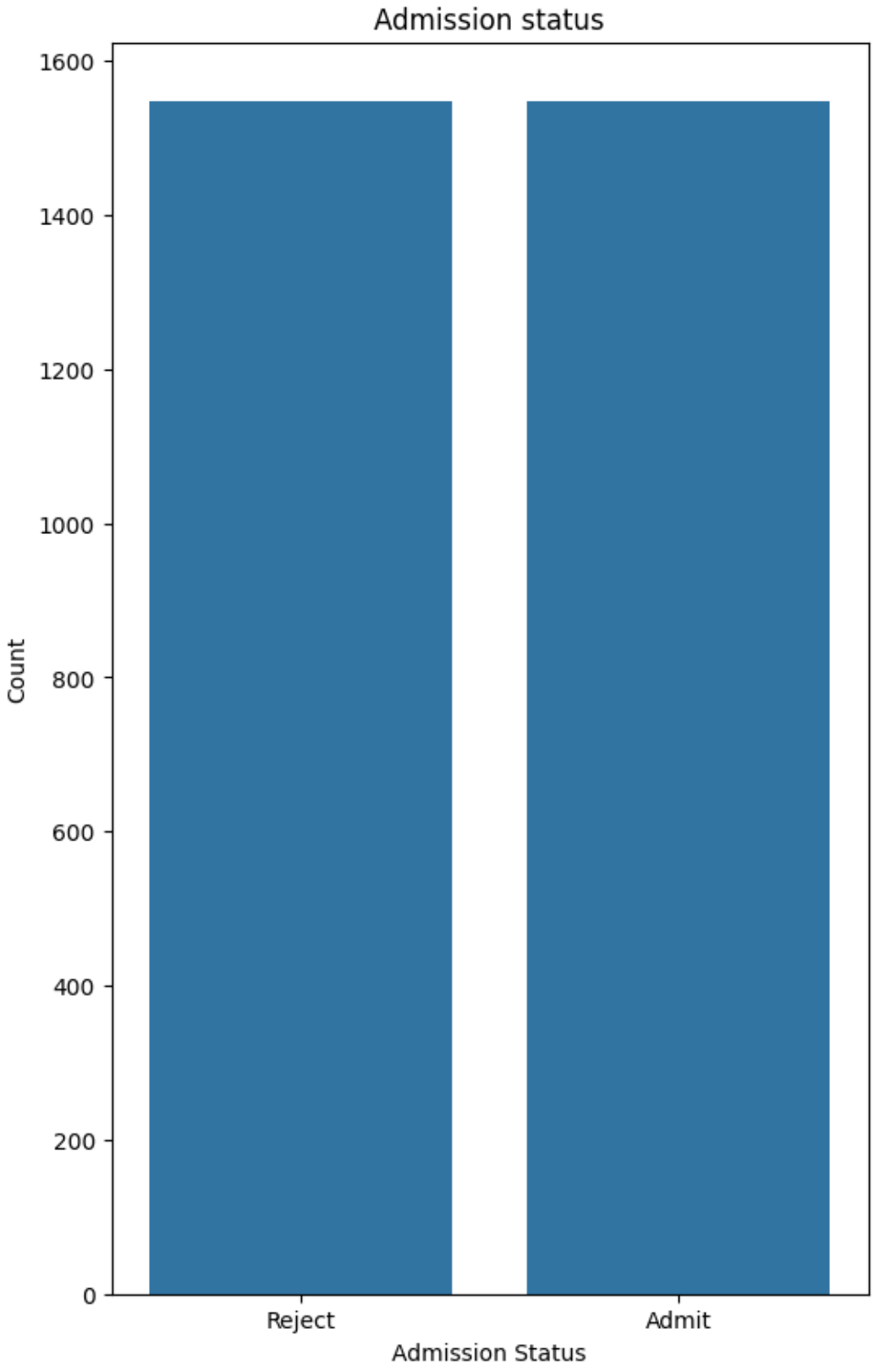
	application_id	gpa	gmat	work_exp
count	3094.000000	3094.000000	3094.000000	3094.000000
mean	3021.454105	3.288151	667.123465	5.029735
std	1825.808225	0.151341	50.577010	1.003270
min	1.000000	2.650000	570.000000	1.000000
25%	1420.250000	3.200000	630.000000	4.000000
50%	3000.500000	3.290000	670.000000	5.000000
75%	4613.750000	3.390000	700.000000	6.000000
max	6194.000000	3.760000	780.000000	8.000000

[9]:
```
data['race']=data['race'].fillna('International')

data.isnull().sum()
```

[9]:
```
application_id    0
gender            0
international     0
gpa               0
major             0
race              0
gmat              0
work_exp          0
work_industry     0
admission         0
dtype: int64
```

[10]:
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(6,10))
sns.countplot(x='admission',data=data)
plt.xlabel("Admission Status")
plt.ylabel('Count')
plt.title("Admission status")
```

```
plt.show()
```


Admission status
1600
1400
1200
1000
800
600
400
200
0
Count
Reject
Admit
Admission Status

[11]:
```
from sklearn.preprocessing import LabelEncoder
```

[12]:
```
le=LabelEncoder()
data['gender']=le.fit_transform(data['gender'])
data['international']=le.fit_transform(data['international'])
data['major']=le.fit_transform(data['major'])
data['race']=le.fit_transform(data['race'])
data['work_industry']=le.fit_transform(data['work_industry'])
data['admission']=le.fit_transform(data['admission'])
```

[13]:
```
data.head()
```

[13]:

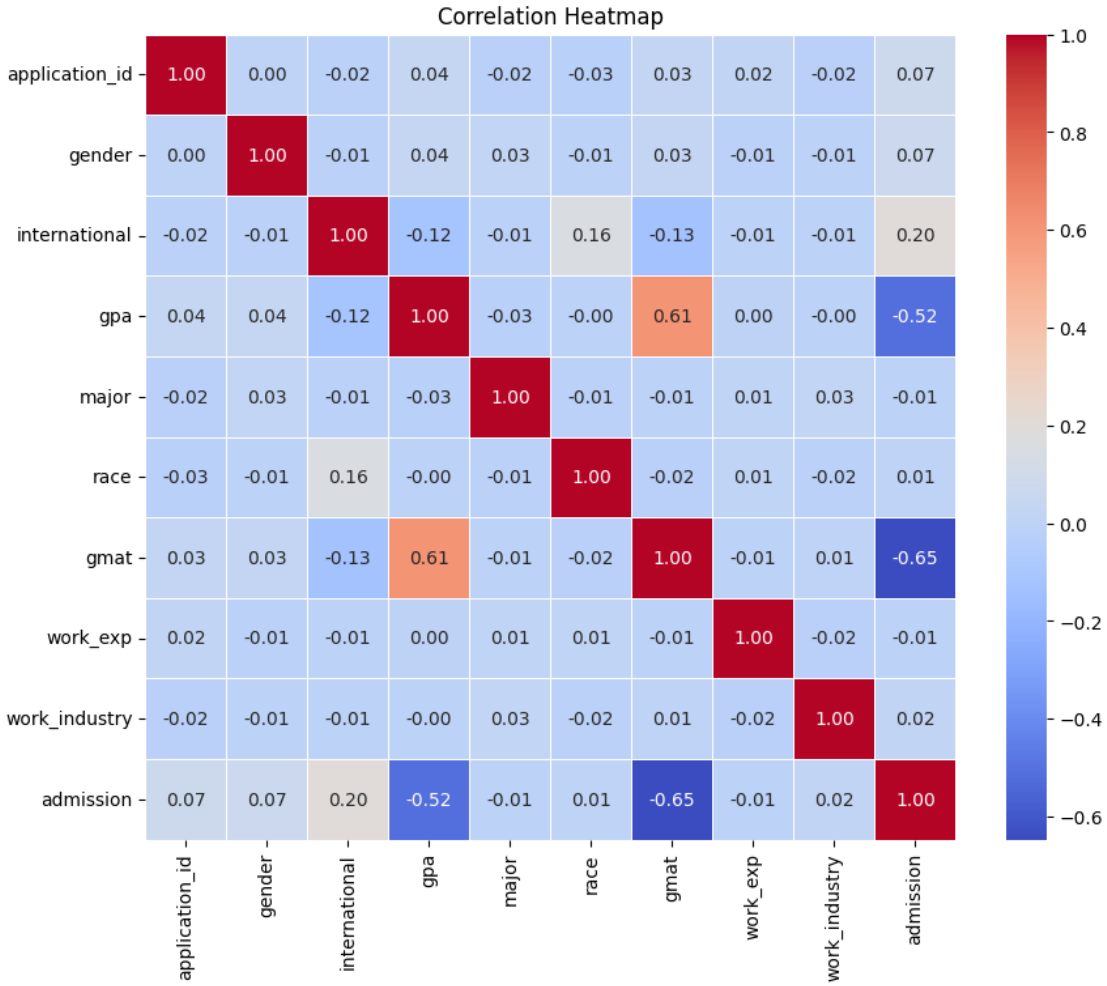
	application_id	gender	international	gpa	major	race	gmat	work_exp	work_industry	admission
0	5244	1	1	3.13	0	3	590.0	5.0	5	1
1	3511	0	1	3.31	1	3	610.0	5.0	13	1
2	2154	1	0	3.17	1	1	710.0	4.0	1	1
3	2703	0	0	3.08	0	5	590.0	6.0	9	1
4	5867	1	0	3.24	1	1	660.0	3.0	9	1

[14]:
```
data.corr
```

[14]:
```
<bound method DataFrame.corr of       application_id  gender  international
gpa  major  race   gmat  \
0               5244       1              1  3.13      0     3  590.0
1               3511       0              1  3.31      1     3  610.0
2               2154       1              0  3.17      1     1  710.0
3               2703       0              0  3.08      0     5  590.0
4               5867       1              0  3.24      1     1  660.0
…                …     …            …   …    …   …    …     …
3089            1641       1              0  2.90      0     2  570.0
3090            4275       1              0  3.25      1     4  720.0
3091            4402       1              0  3.31      0     0  740.0
3092            5122       0              0  3.38      0     5  740.0
3093            3185       0              0  3.23      1     4  670.0

      work_exp  work_industry  admission
0          5.0              5          1
1          5.0             13          1
```

```
2           4.0                 1           1
3           6.0                 9           1
4           3.0                 9           1
…           …                   …           …
3089        6.0                 9           1
3090        4.0                 5           0
3091        6.0                 3           0
3092        5.0                 1           0
3093        6.0                 1           0

[3094 rows x 10 columns]>
```

[15]:
```
plt.figure(figsize=(10, 8))  # Set the figure size
sns.heatmap(data.corr(), annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)
plt.title('Correlation Heatmap')
plt.show()
```



[19]:
```
Y=data[['admission']]
```

[20]:
```
X = data.drop(columns=['admission'])
```

[21]:
```
from sklearn.model_selection import train_test_split
```

[22]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size = 0.2,␣
 ↪random_state = 42)
```

[23]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score,␣
 ↪f1_score
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
```

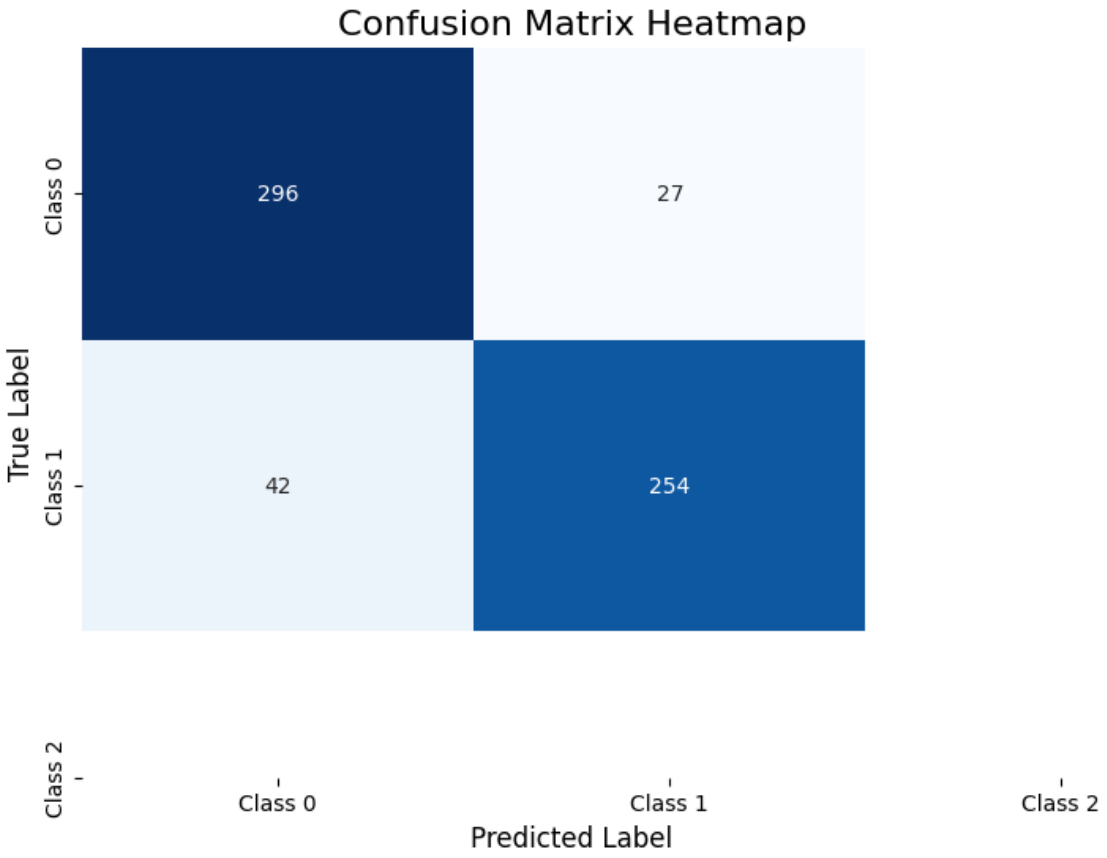
[24]:
```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)


model = DecisionTreeClassifier(max_depth=10, min_samples_leaf=3,␣
 ↪min_samples_split=2)
model.fit(X_train, Y_train)


y_pred = model.predict(X_test)
```

[33]:
```
cm = confusion_matrix(Y_test, y_pred)

# If it's a multi-class classification, print the confusion matrix and␣
 ↪classification report
print("Confusion Matrix:")
print(cm)
```

```
Confusion Matrix:
[[296  27]
 [ 42 254]]
```

[34]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import confusion_matrix

# Assuming Y_test and y_pred are already defined
cm = confusion_matrix(Y_test, y_pred)

# Plot the heatmap
plt.figure(figsize=(8, 6))
```

```
sns.heatmap(cm, annot=True, fmt='g', cmap='Blues', cbar=False)  # fmt='g'␣
 ↪ensures numbers are shown as integers
plt.title('Confusion Matrix Heatmap', fontsize=16)
plt.xlabel('Predicted Label', fontsize=12)
plt.ylabel('True Label', fontsize=12)

# Customize the tick marks and labels
class_labels = ['Class 0', 'Class 1', 'Class 2']  # Adjust according to your␣
 ↪class names
plt.xticks(ticks=[0.5, 1.5, 2.5], labels=class_labels, fontsize=10)
plt.yticks(ticks=[0.5, 1.5, 2.5], labels=class_labels, fontsize=10)

plt.show()
```



[40]:
```
y_pred = model.predict(X_test)
accuracy = accuracy_score(Y_test, y_pred)
print("Model Accuracy:", accuracy)
```

```
Model Accuracy: 0.8885298869143781
```

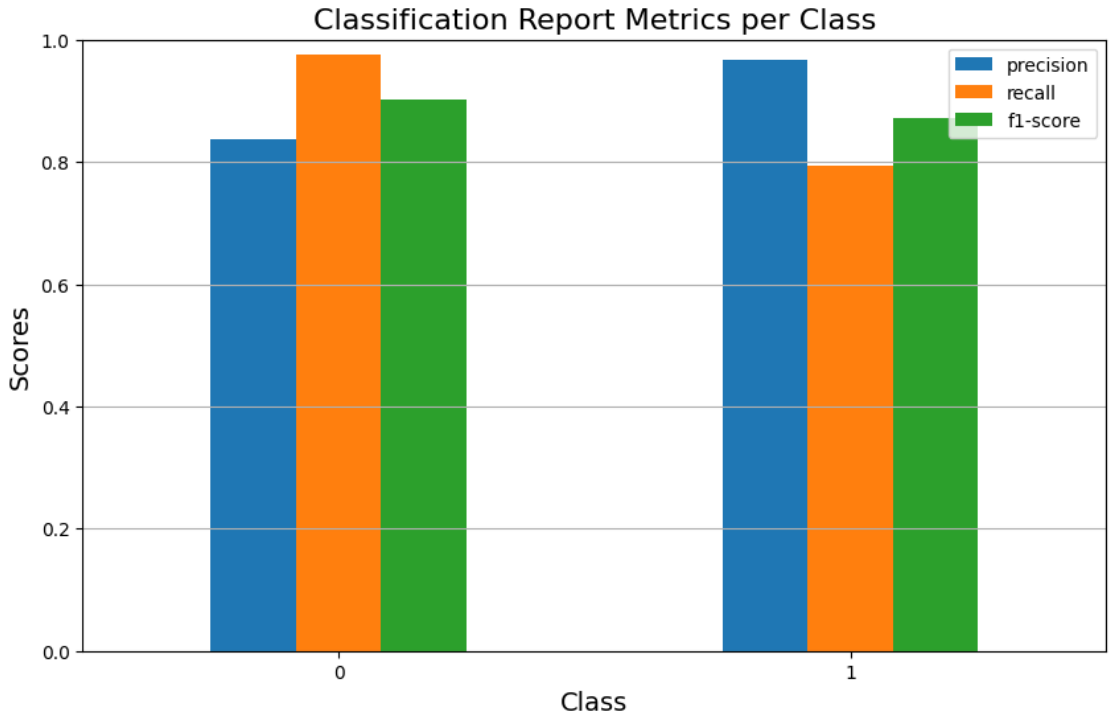
[25]:
```
from sklearn.metrics import classification_report
```

[37]:
```
print("\nClassification Report:")
print(classification_report(Y_test, y_pred))
```

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

           0       0.88      0.92      0.90       323
           1       0.90      0.86      0.88       296

    accuracy                           0.89       619
   macro avg       0.89      0.89      0.89       619
weighted avg       0.89      0.89      0.89       619
```

[31]:
```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
import pandas as pd

# Assuming Y_test and y_pred are already defined
report = classification_report(Y_test, y_pred, output_dict=True)
df_report = pd.DataFrame(report).transpose()

# Remove 'accuracy', 'macro avg', and 'weighted avg' rows as they are not␣
 ↪class-specific
df_report = df_report.drop(['accuracy', 'macro avg', 'weighted avg'])

# Plot the precision, recall, and f1-score for each class
plt.figure(figsize=(10, 6))
df_report[['precision', 'recall', 'f1-score']].plot(kind='bar', figsize=(10, 6))

# Customize the plot
plt.title('Classification Report Metrics per Class', fontsize=16)
plt.xlabel('Class', fontsize=14)
plt.ylabel('Scores', fontsize=14)
plt.xticks(rotation=0)  # Keep x-axis labels horizontal
plt.ylim(0, 1)  # Scores range from 0 to 1
plt.legend(loc='upper right')
plt.grid(axis='y')

plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



[27]:
```
new_data = np.random.rand(9, 9)
predictions = model.predict(new_data)
```

[28]:
```
print("Predictions for new random data:", predictions)
```

```
Predictions for new random data: [0 1 0 0 0 0 0 0 0]
```

[32]:
```
X
```

[32]:

	application_id	gender	international	gpa	major	race	gmat	\
0	5244	1	1	3.13	0	3	590.0	
1	3511	0	1	3.31	1	3	610.0	
2	2154	1	0	3.17	1	1	710.0	
3	2703	0	0	3.08	0	5	590.0	
4	5867	1	0	3.24	1	1	660.0	
…	…	…	…	…	…	…	…	
3089	1641	1	0	2.90	0	2	570.0	
3090	4275	1	0	3.25	1	4	720.0	
3091	4402	1	0	3.31	0	0	740.0	
3092	5122	0	0	3.38	0	5	740.0	
3093	3185	0	0	3.23	1	4	670.0	

	work_exp	work_industry
0	5.0	5

```
1          5.0             13
2          4.0              1
3          6.0              9
4          3.0              9
…          …              …
3089       6.0              9
3090       4.0              5
3091       6.0              3
3092       5.0              1
3093       6.0              1

[3094 rows x 9 columns]
```

[36]:
```
# Example data point:
new_data_single = np.array([[2345,1, 1, 3.13, 0, 3, 590.0, 5.0, 5]])

# Step 2: Standardize the new data using the same scaler that was fit on␣
 ↪training data
new_data_single_scaled = scaler.transform(new_data_single)  # Scale the single␣
 ↪data point
y_pred_single = model.predict(new_data_single_scaled)

# Print the prediction
print("Predicted output for the single data point:", y_pred_single)
```

```
Predicted output for the single data point: [0]
```

[ ]:

[ ]: