merged data

January 19, 2024

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

[4]:
```
df=pd.read_excel('/Users/me2/Desktop/Methu_Projects/FT_ranking/
↪21stOct_mergefinal.xlsx')
```

[5]:
```
df.head()
```

[5]:

	Year	#	School Name	Programme name	Overall satisfaction	Career service rank	Faculty with doctorates (%)	Women on board (%)	International course experience rank	International faculty (%)	…	International students (%)	Internships (%)	International board (%)
0	2020	53	Aalto University	MSc in Economics and Business Administration	8.70	65	97	43	46	26	…	14	100	43
1	2021	56	Aalto University	MSc in Economics and Business Administration	8.74	72	96	43	44	29	…			
2	2022	59	Aalto University	Master of Science in Economics and Business Ad…	8.87	64	98	43	40	32	…			
3	2021	95	Adam Smith Business School, University of Glasgow	MSc Management	9.60	19	83	43	99	70	…			
4	2020	79	Alliance Manchester Business School	MSc Business Analysis & Strategic Management	9.03	15	94	42	85	52	…			

```
1                          17              100                          43
2                          13               37                          43
3                          98              100                           7
4                          85              100                          17

   Location Weighted salary (US$) Aims achieved (%)  Company internships (%)  \
0   Finland               64,271            86.000                       41
1   Finland               67,381            86.702                       37
2   Unknown               70,790            85.837                      100
3        UK               38,897            83.636                        0
4        UK               53,711            81.000                       72

   Salary percentage increase Salary today (US$)  \
0                      48.000              64333
1                      48.345              67417
2                      43.680              69624
3                      51.730              38897
4                      70.000              53711

  Average course length (months)
0                           24.0
1                           24.0
2                           24.0
3                           12.0
4                           14.0

[5 rows x 27 columns]
```

[6]:
```
df.describe()
```

[6]:
```
              Year           #  Overall satisfaction  Career service rank  \
count   290.000000  290.000000            290.000000           290.000000
mean   2021.034483   48.755172              8.842897            48.948276
std       0.810090   28.124760              0.433850            28.145951
min    2020.000000    1.000000              7.090000             1.000000
25%    2020.000000   25.000000              8.560000            25.000000
50%    2021.000000   49.000000              8.815000            49.000000
75%    2022.000000   73.000000              9.160000            73.000000
max    2022.000000  100.000000              9.980000           100.000000

       Faculty with doctorates (%)  Women on board (%)  \
count                   290.000000          290.000000
mean                     94.234483           37.310345
std                       7.161038           13.085832
min                      62.000000            0.000000
25%                      91.000000           27.250000
50%                      97.000000           38.000000
```

```
75%                    100.000000           50.000000
max                    100.000000           68.000000

       International course experience rank  International faculty (%)  \
count                            290.000000                 290.000000
mean                              48.789655                  42.744828
std                               27.887947                  24.851888
min                                1.000000                   0.000000
25%                               25.000000                  24.000000
50%                               49.000000                  44.000000
75%                               73.000000                  61.000000
max                               99.000000                  97.000000

       International mobility rank  Employed in three months  \
count                    290.000000                 290.000000
mean                      48.944828                  74.347414
std                       28.148710                  19.140297
min                        1.000000                  10.980000
25%                       25.000000                  61.637500
50%                       49.000000                  79.740000
75%                       73.000000                  88.150000
max                      100.000000                 100.000000

       Career progress rank  Female students (%)  Value for money rank  \
count            290.000000           290.000000            290.000000
mean              48.948276            49.068966             48.948276
std               28.145951             9.660432             28.145951
min                1.000000            19.000000              1.000000
25%               25.000000            43.000000             25.000000
50%               49.000000            50.000000             49.000000
75%               73.000000            55.000000             73.000000
max              100.000000            77.000000            100.000000

       International students (%)  Aims achieved (%)  Company internships (%)  \
count                  290.000000         290.000000               290.000000
mean                    48.917241          84.797955                76.444828
std                     29.556468           3.119818                37.869454
min                      0.000000          75.177000                 0.000000
25%                     25.000000          82.744750                59.250000
50%                     46.000000          84.984000               100.000000
75%                     76.750000          87.000000               100.000000
max                     99.000000          95.573000               100.000000

       Salary percentage increase  Salary today (US$)  \
count                  290.000000          290.000000
mean                    54.478510        72885.803448
std                     17.700076        16860.709234
```

```
min                      19.000000        31582.000000
25%                      42.796750        63954.250000
50%                      51.324000        69624.000000
75%                      61.604000        76022.750000
max                     125.000000       141007.000000

       Average course length (months)
count                      290.000000
mean                        20.159552
std                          7.226496
min                          8.000000
25%                         14.000000
50%                         21.000000
75%                         24.000000
max                         64.000000
```

[7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290 entries, 0 to 289
Data columns (total 27 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Year                                  290 non-null    int64
 1   #                                     290 non-null    int64
 2   School Name                           290 non-null    object
 3   Programme name                        290 non-null    object
 4   Overall satisfaction                  290 non-null    float64
 5   Career service rank                   290 non-null    int64
 6   Faculty with doctorates (%)           290 non-null    int64
 7   Women on board (%)                    290 non-null    int64
 8   International course experience rank  290 non-null    int64
 9   International faculty (%)             290 non-null    int64
 10  International mobility rank           290 non-null    int64
 11  Employed in three months              290 non-null    float64
 12  Career progress rank                  290 non-null    int64
 13  Female students (%)                   290 non-null    int64
 14  Location by primary campus            290 non-null    object
 15  Female faculty (%)                    290 non-null    object
 16  Value for money rank                  290 non-null    int64
 17  International students (%)             290 non-null    int64
 18  Internships (%)                       290 non-null    object
 19  International board (%)               290 non-null    object
 20  Location                              290 non-null    object
 21  Weighted salary (US$)                 290 non-null    object
 22  Aims achieved (%)                     290 non-null    float64
 23  Company internships (%)               290 non-null    int64
 24  Salary percentage increase            290 non-null    float64
```

```
 25  Salary today (US$)                    290 non-null    int64
 26  Average course length (months)        290 non-null    float64
dtypes: float64(5), int64(14), object(8)
memory usage: 61.3+ KB
```

[8]:
```
df.isnull().sum()
```

[8]:
```
Year                                  0
#                                     0
School Name                           0
Programme name                        0
Overall satisfaction                  0
Career service rank                   0
Faculty with doctorates (%)           0
Women on board (%)                    0
International course experience rank  0
International faculty (%)             0
International mobility rank           0
Employed in three months              0
Career progress rank                  0
Female students (%)                   0
Location by primary campus            0
Female faculty (%)                    0
Value for money rank                  0
International students (%)            0
Internships (%)                       0
International board (%)               0
Location                              0
Weighted salary (US$)                 0
Aims achieved (%)                     0
Company internships (%)               0
Salary percentage increase            0
Salary today (US$)                    0
Average course length (months)        0
dtype: int64
```

[64]:
```
df['Weighted salary (US$)'] = pd.to_numeric(df['Weighted salary (US$)'].
 ↪replace('[^\d.]', '', regex=True), errors='coerce')
```

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290 entries, 0 to 289
Data columns (total 27 columns):
 #   Column                                Non-Null Count  Dtype
---  ------                                --------------  -----
 0   Year                                  290 non-null    int64
 1   #                                     290 non-null    int64
```

```
 2   School Name                            290 non-null    object
 3   Programme name                         290 non-null    object
 4   Overall satisfaction                   290 non-null    float64
 5   Career service rank                    290 non-null    int64
 6   Faculty with doctorates (%)            290 non-null    int64
 7   Women on board (%)                     290 non-null    int64
 8   International course experience rank   290 non-null    int64
 9   International faculty (%)              290 non-null    int64
 10  International mobility rank            290 non-null    int64
 11  Employed in three months               290 non-null    float64
 12  Career progress rank                   290 non-null    int64
 13  Female students (%)                    290 non-null    int64
 14  Location by primary campus             290 non-null    object
 15  Female faculty (%)                     290 non-null    object
 16  Value for money rank                   290 non-null    int64
 17  International students (%)              290 non-null    int64
 18  Internships (%)                        290 non-null    object
 19  International board (%)                290 non-null    object
 20  Location                               290 non-null    object
 21  Weighted salary (US$)                  290 non-null    object
 22  Aims achieved (%)                      290 non-null    float64
 23  Company internships (%)                290 non-null    int64
 24  Salary percentage increase             290 non-null    float64
 25  Salary today (US$)                     290 non-null    int64
 26  Average course length (months)         290 non-null    float64
dtypes: float64(5), int64(14), object(8)
memory usage: 61.3+ KB
```

[65]:
```
df = df.drop(columns=['International course experience rank','Career progress␣
↪rank','Year','School Name'])
```

[66]:
```
pip install xgboost
```

```
Requirement already satisfied: xgboost in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(2.0.0)
Requirement already satisfied: numpy in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from xgboost) (1.24.0)
Requirement already satisfied: scipy in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from xgboost) (1.11.2)
Note: you may need to restart the kernel to use updated packages.
```

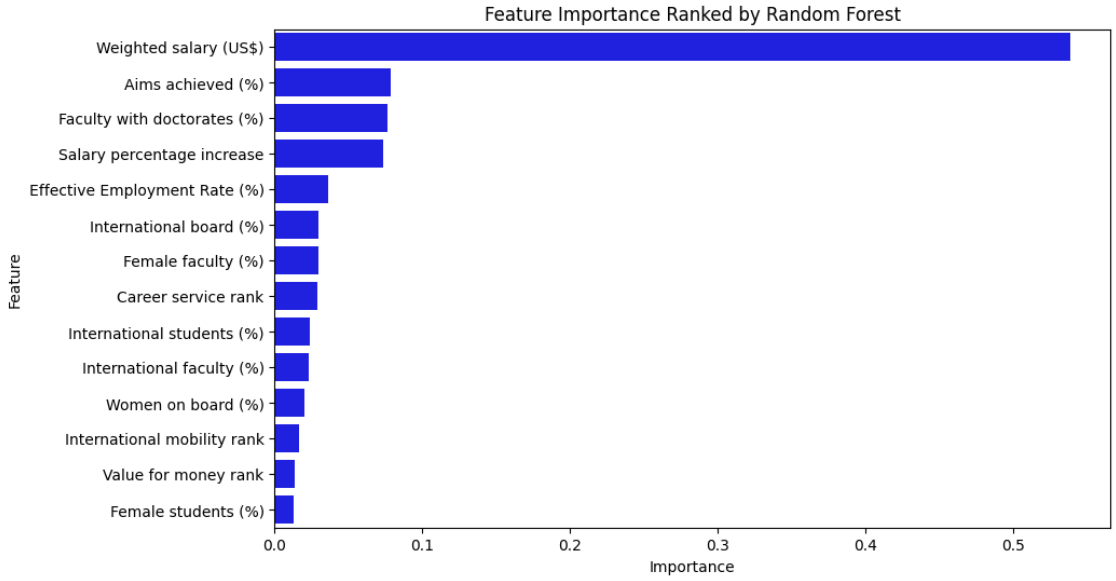
[67]:
```
import pandas as pd
from sklearn.ensemble import RandomForestRegressor

# Separate the features (X) from the target variable (y)
```

```
X = df.drop(columns=['Rank'])  # Assuming '#' is the target variable
y = df['Rank']

# Initialize the Random Forest model
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)

# Train the model on the entire dataset
rf_model.fit(X, y)

# Extract feature importance scores
feature_importances = rf_model.feature_importances_

# Create a DataFrame to hold the feature names and their corresponding␣
 ↪importance scores
feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': feature_importances
})

# Sort the features based on importance
feature_importance_df = feature_importance_df.sort_values(by='Importance',␣
 ↪ascending=False)

# Display the sorted feature importances
print(feature_importance_df)
```

```
                        Feature  Importance
0           Weighted salary (US$)    0.539087
3               Aims achieved (%)    0.078367
12    Faculty with doctorates (%)    0.076128
1      Salary percentage increase    0.073422
13  Effective Employment Rate (%)    0.036173
10        International board (%)    0.029304
5              Female faculty (%)    0.029298
4             Career service rank    0.028800
9      International students (%)    0.024031
8       International faculty (%)    0.022803
7              Women on board (%)    0.019935
11    International mobility rank    0.016430
2            Value for money rank    0.013375
6             Female students (%)    0.012847
```

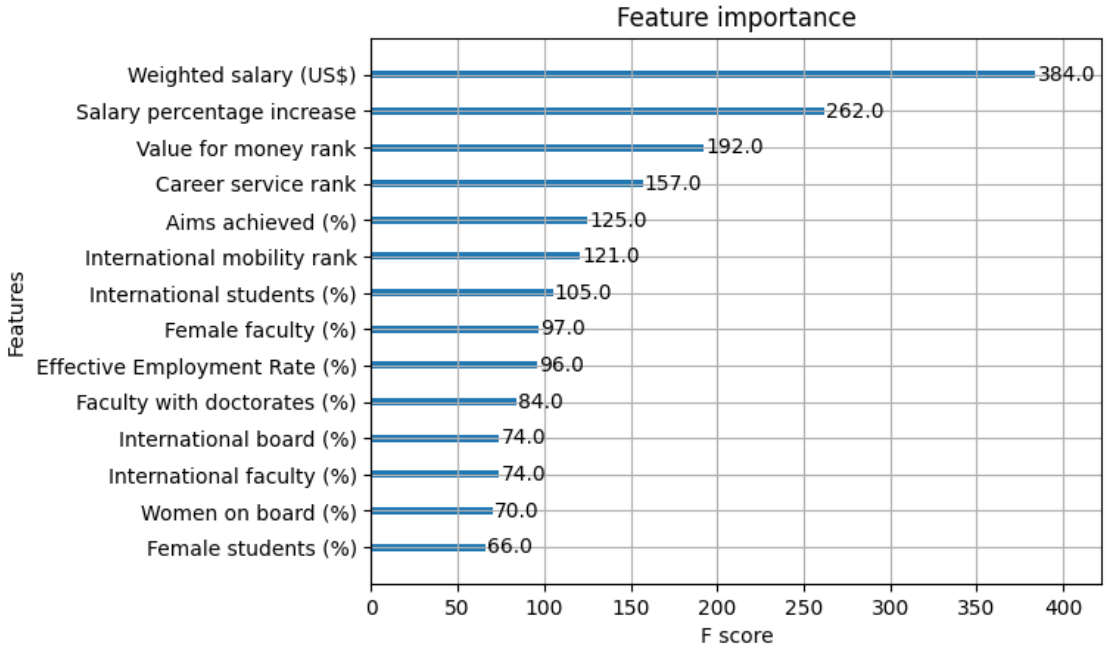
[68]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Set up the matplotlib figure
plt.figure(figsize=(10, 6))
```

```
# Draw a horizontal bar plot of feature importances
sns.barplot(x='Importance', y='Feature', data=feature_importance_df, color='b')

# Add labels and title
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.title('Feature Importance Ranked by Random Forest')

# Show the plot
plt.show()
```



[69]:
```
import xgboost as xgb
from xgboost import plot_importance
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split

X = df.drop(columns=['Rank'])  # Assuming '#' is the target variable
y = df['Rank']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

model = xgb.XGBRegressor(objective ='reg:squarederror')
model.fit(X_train, y_train)

plot_importance(model)
plt.show()
```

```
# Get feature importances
importances = model.feature_importances_

# Sort features by importance
sorted_idx = importances.argsort()
```



[70]:
```
# Install Bayesian-Optimization
!pip install bayesian-optimization
```

```
Requirement already satisfied: bayesian-optimization in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(1.4.3)
Requirement already satisfied: numpy>=1.9.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from bayesian-optimization) (1.24.0)
Requirement already satisfied: scipy>=1.0.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from bayesian-optimization) (1.11.2)
Requirement already satisfied: scikit-learn>=0.18.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from bayesian-optimization) (1.3.0)
Requirement already satisfied: colorama>=0.4.6 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from bayesian-optimization) (0.4.6)
Requirement already satisfied: joblib>=1.1.1 in
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from scikit-learn>=0.18.0->bayesian-optimization) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from scikit-learn>=0.18.0->bayesian-optimization) (3.2.0)
```

[71]:
```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error, r2_score

X_final = df.drop(columns=['Rank'])
y_final = df['Rank']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final,␣
 ↪test_size=0.2, random_state=42)

# Initialize the model
gb_model = GradientBoostingRegressor(random_state=42)

# Train the model
gb_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = gb_model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae}')
# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calculate R-squared (R²)
r_squared = r2_score(y_test, y_pred)

print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R²): {r_squared}")
```

```
Mean Absolute Error: 7.369784071785871
Mean Squared Error (MSE): 180.43624676011115
Root Mean Squared Error (RMSE): 13.432655983092515
R-squared (R²): 0.784068444259337
```

[18]:
```
Selected_columns = ['International board (%)',
       'Faculty with doctorates (%)',
       'Female students (%)', 'Female faculty (%)', 'S
       'Career service rank',
       'Weighted salary (US$)', 'Value for money rank',
       'Effective Employment Rate (%)', 'Aims achieved (%)']
```

[19]:
```
# Import important libraries
import itertools
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
import xgboost as xgb
from bayes_opt import BayesianOptimization


# Create function to caculate mae, with input are parameters of XGB: learning_rate, max_depth, n_estimators
def evaluate_mae(learning_rate, max_depth, n_estimators):
    X = df[Selected_columns]
    y = df['Rank']

    # Keep the same values for parameters
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    xgb_regressor = xgb.XGBRegressor(
        objective ='reg:squarederror',
        n_estimators=int(n_estimators),
        learning_rate=learning_rate,
        max_depth=int(max_depth),
        random_state=42
    )

    # Train the model
    xgb_regressor.fit(X_train, y_train)
    y_pred = xgb_regressor.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    return -mae  # We return negative MAE because BayesianOptimization tries to maximize the function

# Define the bounds for the hyperparameters
pbounds = {
    'learning_rate': (0.01, 0.3),
    'max_depth': (3, 10),
    'n_estimators': (50, 500)
}
```

```python
# Initialize Bayesian optimization
optimizer = BayesianOptimization(
    f=evaluate_mae,
    pbounds=pbounds,
    random_state=42,
)

# Optimize
optimizer.maximize(
    init_points=5,
    n_iter=25,
)

# Print the best hyperparameters
print(optimizer.max)
```

```
|   iter    |  target   | learni… | max_depth | n_esti… |
-------------------------------------------------------------
| 1         | -5.187    | 0.1186    | 9.655
| 379.4     |
| 2         | -5.648    | 0.1836    | 4.092
| 120.2     |
| 3         | -5.41     | 0.02684   | 9.063
| 320.5     |
| 4         | -5.145    | 0.2153    | 3.144
 | 486.5     |
| 5         | -5.536    | 0.2514    | 4.486
| 131.8     |
| 6         | -5.018    | 0.2028    | 4.822
 | 458.2     |
| 7         | -6.769    | 0.01259   | 4.306
| 459.2     |
| 8         | -5.113    | 0.2072    | 4.9
| 458.3     |
| 9         | -5.731    | 0.07384   | 5.119
| 457.6     |
| 10        | -5.237    | 0.06643   | 4.271
| 457.8     |
| 11        | -5.534    | 0.1723    | 3.155
| 485.8     |
| 12        | -5.348    | 0.07797   | 3.255
| 487.2     |
| 13        | -5.37     | 0.2798    | 3.513
| 457.7     |
| 14        | -5.635    | 0.2956    | 3.71
| 486.4     |
```

```
| 15        | -5.455    | 0.2395    | 9.767
| 378.8     |
| 16        | -5.347    | 0.0994    | 9.028
| 92.23     |
| 17        | -5.305    | 0.05869   | 9.092
| 379.8     |
| 18        | -5.115    | 0.08778   | 4.787
| 277.0     |
| 19        | -5.531    | 0.1876    | 3.431
| 241.2     |
| 20        | -5.518    | 0.2577    | 4.66
| 277.4     |
| 21        | -5.822    | 0.05068   | 4.875
| 276.9     |
| 22        | -5.55     | 0.214     | 8.91
| 92.08     |
| 23        | -5.171    | 0.1075    | 3.407
| 445.3     |
| 24        | -5.147    | 0.2653    | 8.864
| 194.3     |
| 25        | -5.491    | 0.1581    | 8.905
| 194.3     |
| 26        | -5.318    | 0.2718    | 8.978
| 194.3     |
| 27        | -5.474    | 0.1628    | 4.549
| 471.2     |
| 28        | -5.286    | 0.03039   | 8.465
| 442.6     |
| 29        | -5.227    | 0.1466    | 3.216
| 487.3     |
| 30        | -5.535    | 0.0725    | 4.851
| 277.1     |
=============================================================
{'target': -5.018198283513387, 'params': {'learning_rate': 0.20284562200803863,
'max_depth': 4.822008002962021, 'n_estimators': 458.23776381762343}}
```

[36]:
```python
import itertools
import random
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error, r2_score

# List of metrics
metrics = Selected_columns

# Function to evaluate MAE for a given set of metrics
def evaluate_mae(selected_metrics):
```

```
    X = df[Selected_columns]
    y = df['Rank']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

    xgb_regressor = xgb.XGBRegressor(objective ='reg:squarederror',␣
 ↪n_estimators=100, learning_rate=0.1, max_depth=8, random_state=42)
    xgb_regressor.fit(X_train, y_train)

    #gb_regressor = GradientBoostingRegressor(n_estimators=100, learning_rate=0.
 ↪1, max_depth=3, random_state=42)
    #gb_regressor.fit(X_train, y_train)
    y_pred = xgb_regressor.predict(X_test)
    mae = mean_absolute_error(y_test, y_pred)
    return mae

# Loop through different combinations of metrics and evaluate MAE
best_mae = float('inf')
best_combination = []

# Find combo of metrics with at least 5 metrics
for r in range(5, len(metrics) + 1):
    for subset in itertools.combinations(metrics, r):
        mae = evaluate_mae(list(subset))
        if mae < best_mae:
            best_mae = mae
            best_combination = subset

# Print the best combination and its MAE
print(f"Best combination of metrics: {best_combination}")
print(f"Lowest MAE: {best_mae}")

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calculate R-squared (R²)
r_squared = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R²): {r_squared}")
```

```
Best combination of metrics: ('International board (%)', 'Faculty with
```

```
doctorates (%)', 'Female students (%)', 'Female faculty (%)', 'Career service
rank')
Lowest MAE: 5.202130677964952
Mean Absolute Error (MAE): 5.202130677964952
Mean Squared Error (MSE): 153.62513256702798
Root Mean Squared Error (RMSE): 12.394560604032238
R-squared (R²): 0.8161538245684826
```

[28]:
```python
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.feature_selection import RFE
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

# Assuming df is your dataset and "TargetColumn" is your target metric
X = df[Selected_columns]  # This will remove the target column and keep only␣
 ↪features.
y = df["Rank"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# Initialize Gradient Boosting Regressor
gbr = GradientBoostingRegressor(random_state=42)

# Recursive Feature Elimination
selector = RFE(gbr, step=1)
selector = selector.fit(X_train, y_train)

# Using the boolean mask to subset the DataFrames
X_train_selected = X_train.loc[:, selector.support_]
X_test_selected = X_test.loc[:, selector.support_]

gbr.fit(X_train_selected, y_train)

# Predict and compute MAE
y_pred = gbr.predict(X_test_selected)
mae = mean_absolute_error(y_test, y_pred)

print(f"MAE with selected features: {mae}")
```

```
MAE with selected features: 8.801363149532705
```

[30]:
```
pip install mlxtend
```

```
Collecting mlxtend
  Obtaining dependency information for mlxtend from https://files.pythonhosted.o
rg/packages/73/da/d5d77a9a7a135c948dbf8d3b873655b105a152d69e590150c83d23c3d070/m
lxtend-0.23.0-py3-none-any.whl.metadata
```

```
  Downloading mlxtend-0.23.0-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from mlxtend) (1.11.2)
Requirement already satisfied: numpy>=1.16.2 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from mlxtend) (1.24.0)
Requirement already satisfied: pandas>=0.24.2 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from mlxtend) (2.0.3)
Requirement already satisfied: scikit-learn>=1.0.2 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from mlxtend) (1.3.0)
Requirement already satisfied: matplotlib>=3.0.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from mlxtend) (3.7.2)
Requirement already satisfied: joblib>=0.13.2 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from mlxtend) (1.3.2)
Requirement already satisfied: contourpy>=1.0.1 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (1.1.0)
Requirement already satisfied: cycler>=0.10 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (4.42.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (23.1)
Requirement already satisfied: pillow>=6.2.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (10.0.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from pandas>=0.24.2->mlxtend) (2023.3)
Requirement already satisfied: tzdata>=2022.1 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
```

```
(from pandas>=0.24.2->mlxtend) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from scikit-learn>=1.0.2->mlxtend) (3.2.0)
Requirement already satisfied: six>=1.5 in
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/site-packages
(from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
Downloading mlxtend-0.23.0-py3-none-any.whl (1.4 MB)
                         1.4/1.4 MB
15.0 MB/s eta 0:00:00a 0:00:01
Installing collected packages: mlxtend
Successfully installed mlxtend-0.23.0
Note: you may need to restart the kernel to use updated packages.
```

[31]:
```python
from mlxtend.feature_selection import SequentialFeatureSelector
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error

# Assuming df is your dataset and "TargetColumn" is your target metric
X = df[Selected_columns]  # This will remove the target column and keep only␣
 ↪features.
y = df["Rank"]

# Initialize Gradient Boosting Regressor
gbr = GradientBoostingRegressor(random_state=42)

# Sequential Forward Selection
sfs = SequentialFeatureSelector(gbr,
           k_features="best",
           forward=True,
           floating=False,
           scoring='neg_mean_absolute_error',
           cv=5)

sfs = sfs.fit(X, y)

selected_features = X.columns[list(sfs.k_feature_idx_)]
print(f"Best features: {selected_features}")
```

```
Best features: Index(['International board (%)', 'Faculty with doctorates (%)',
       'Female students (%)', 'Female faculty (%)', 'Weighted salary (US$)',
       'Value for money rank', 'Effective Employment Rate (%)',
       'Aims achieved (%)'],
      dtype='object')
```

[44]:
```python
metrics = ['International board (%)', 'Faculty with doctorates (%)',
       'Female students (%)', 'Female faculty (%)', 'Weighted salary (US$)',
       'Value for money rank', 'Effective Employment Rate (%)',
```

```
       'Aims achieved (%)']
```

[45]:
```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error


X_final = df[metrics]
y_final = df["Rank"]

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final,␣
 ↪test_size=0.2, random_state=42)

# Initialize the model
gb_model = GradientBoostingRegressor(random_state=42)

# Train the model
gb_model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = gb_model.predict(X_test)

# Evaluate the model
mae = mean_absolute_error(y_test, y_pred)
print(f'Mean Absolute Error: {mae}')
```

```
Mean Absolute Error: 7.087208483907138
```

[46]:
```
from sklearn.metrics import mean_squared_error, r2_score

# Calculate Mean Squared Error (MSE)
mse = mean_squared_error(y_test, y_pred)

# Calculate Root Mean Squared Error (RMSE)
rmse = np.sqrt(mse)

# Calculate R-squared (R²)
r_squared = r2_score(y_test, y_pred)

print(f"Mean Absolute Error (MAE): {mae}")
print(f"Mean Squared Error (MSE): {mse}")
print(f"Root Mean Squared Error (RMSE): {rmse}")
print(f"R-squared (R²): {r_squared}")
```

```
Mean Absolute Error (MAE): 7.087208483907138
Mean Squared Error (MSE): 153.62513256702798
Root Mean Squared Error (RMSE): 12.394560604032238
```

```
R-squared (R²): 0.8161538245684826
```

[47]:
```python
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

X= df[metrics]
y= df["Rank"]

# 2. Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)

# 3. Train Random Forest and SVM models
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

svm = SVR(kernel='linear')
svm.fit(X_train, y_train)

# 4. Predict on test data
y_pred_rf = rf.predict(X_test)
y_pred_svm = svm.predict(X_test)

# 5. Calculate and print metrics
metrics = {
    "MSE": mean_squared_error,
    "MAE": mean_absolute_error,
    "RMSE": lambda y_true, y_pred: np.sqrt(mean_squared_error(y_true, y_pred)),
    "R^2": r2_score
}

print("Random Forest Metrics:")
for name, func in metrics.items():
    print(f"{name}: {func(y_test, y_pred_rf)}")

print("\nSVM Metrics:")
for name, func in metrics.items():
    print(f"{name}: {func(y_test, y_pred_svm)}")
```

```
Random Forest Metrics:
MSE: 157.8473066666667
MAE: 7.523111111111111
RMSE: 12.563729807133974
R^2: 0.8111010669418235
```

```
SVM Metrics:
MSE: 1609468.9298711508
MAE: 1102.9756358583459
RMSE: 1268.6484658372274
R^2: -1925.0826811886827
```

[ ]: