Machine Learning Model (Total Completed)

This code block prepares the sample data for training a machine learning model to predict the 'user_completed'.

```
sample = user_country_stats.sample(n=1000, random_state=42)

x_sample = sample.drop(labels="user_completed", axis=1)
y_sample = np.log1p(sample["user_completed"])
```

This code block splits the data into training and testing sets.

```
x_sample_train, x_sample_test, y_sample_train, y_sample_test = train_test_split(x_sample, y_sample, test_size=0.2, random_state=42
```

This code block splits the main dataset into features and the target variable, and then further divides these into training and testing sets.

```
X = user_country_stats.drop('user_completed', axis=1)
y = np.log1p(user_country_stats['user_completed'])

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

This code block focuses on cleaning the column names of training and testing feature sets, X_train and X_test.

```
X_train.columns = [re.sub('[^A-Za-z0-9_]+', '_', col) for col in X_train.columns]
X_test.columns = [re.sub('[^A-Za-z0-9_]+', '_', col) for col in X_test.columns]
```

This code block trains a baseline machine learning model, specifically a Random Forest Regressor, on a sample of the sample data and then evaluates its performance.

```
np.random.seed(42)

base_model_sample = RandomForestRegressor(n_jobs=-1)

base_model_sample.fit(X=x_sample_train, y=y_sample_train)

base_model_sample_scores = show_scores(model=base_model_sample,
                                       train_features=x_sample_train,
                                       train_labels=y_sample_train,
                                       test_features=x_sample_test,
                                       test_labels=y_sample_test)

print(f"Model score on {len(sample)} samples:")
base_model_sample_scores
```

```
Model score on 1000 samples:
{'Training MAE': 0.130003875443669,
 'Test MAE': 0.3813792642886928,
 'Training RMSLE': 0.04109369523920908,
 'Test RMSLE': 0.1327517882889829,
 'Training R^2': 0.9818933262318,
 'Test R^2': 0.8587871454038378}
```

This code block tunes the hyperparameters of a Random Forest Regressor model using a technique called Randomized Search Cross-Validation.

```
np.random.seed(42)

rf_grid = {"n_estimators": np.arange(10, 200, 10),
           "max_depth": [None, 10, 20],
           "min_samples_split": np.arange(2, 10, 1),
           "min_samples_leaf": np.arange(1, 10, 1),
           "max_features": [0.5, 1.0, "sqrt"], }

sample_rs_model = RandomizedSearchCV(estimator=RandomForestRegressor(),
                              param_distributions=rf_grid,
                              n_iter=20,
                              cv=3,
                              verbose=3)

sample_rs_model.fit(X=x_sample_train,
                    y=y_sample_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
[CV 1/3] END max_depth=10, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=130;, score=0.820 total time=
[CV 2/3] END max_depth=10, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=130;, score=0.805 total time=
[CV 3/3] END max_depth=10, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=130;, score=0.821 total time=
[CV 1/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=7, n_estimators=60;, score=0.880 total time=
[CV 2/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=7, n_estimators=60;, score=0.861 total time=
[CV 3/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=7, n_estimators=60;, score=0.859 total time=
[CV 1/3] END max_depth=10, max_features=0.5, min_samples_leaf=9, min_samples_split=5, n_estimators=140;, score=0.882 total time=
[CV 2/3] END max_depth=10, max_features=0.5, min_samples_leaf=9, min_samples_split=5, n_estimators=140;, score=0.859 total time=
[CV 3/3] END max_depth=10, max_features=0.5, min_samples_leaf=9, min_samples_split=5, n_estimators=140;, score=0.860 total time=
[CV 1/3] END max_depth=10, max_features=0.5, min_samples_leaf=8, min_samples_split=3, n_estimators=50;, score=0.882 total time=   0
[CV 2/3] END max_depth=10, max_features=0.5, min_samples_leaf=8, min_samples_split=3, n_estimators=50;, score=0.860 total time=   0
[CV 3/3] END max_depth=10, max_features=0.5, min_samples_leaf=8, min_samples_split=3, n_estimators=50;, score=0.855 total time=   0
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=7, min_samples_split=7, n_estimators=140;, score=0.799 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=7, min_samples_split=7, n_estimators=140;, score=0.784 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=7, min_samples_split=7, n_estimators=140;, score=0.815 total time=
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=3, n_estimators=180;, score=0.822 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=3, n_estimators=180;, score=0.808 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=3, n_estimators=180;, score=0.824 total time=
[CV 1/3] END max_depth=10, max_features=1.0, min_samples_leaf=2, min_samples_split=7, n_estimators=160;, score=0.878 total time=
[CV 2/3] END max_depth=10, max_features=1.0, min_samples_leaf=2, min_samples_split=7, n_estimators=160;, score=0.861 total time=
[CV 3/3] END max_depth=10, max_features=1.0, min_samples_leaf=2, min_samples_split=7, n_estimators=160;, score=0.846 total time=
[CV 1/3] END max_depth=10, max_features=1.0, min_samples_leaf=6, min_samples_split=3, n_estimators=150;, score=0.882 total time=
[CV 2/3] END max_depth=10, max_features=1.0, min_samples_leaf=6, min_samples_split=3, n_estimators=150;, score=0.860 total time=
[CV 3/3] END max_depth=10, max_features=1.0, min_samples_leaf=6, min_samples_split=3, n_estimators=150;, score=0.851 total time=
[CV 1/3] END max_depth=None, max_features=0.5, min_samples_leaf=4, min_samples_split=2, n_estimators=110;, score=0.883 total time=
[CV 2/3] END max_depth=None, max_features=0.5, min_samples_leaf=4, min_samples_split=2, n_estimators=110;, score=0.862 total time=
[CV 3/3] END max_depth=None, max_features=0.5, min_samples_leaf=4, min_samples_split=2, n_estimators=110;, score=0.859 total time=
[CV 1/3] END max_depth=10, max_features=0.5, min_samples_leaf=3, min_samples_split=2, n_estimators=190;, score=0.884 total time=
[CV 2/3] END max_depth=10, max_features=0.5, min_samples_leaf=3, min_samples_split=2, n_estimators=190;, score=0.865 total time=
[CV 3/3] END max_depth=10, max_features=0.5, min_samples_leaf=3, min_samples_split=2, n_estimators=190;, score=0.860 total time=
[CV 1/3] END max_depth=10, max_features=1.0, min_samples_leaf=1, min_samples_split=7, n_estimators=120;, score=0.876 total time=
[CV 2/3] END max_depth=10, max_features=1.0, min_samples_leaf=1, min_samples_split=7, n_estimators=120;, score=0.867 total time=
[CV 3/3] END max_depth=10, max_features=1.0, min_samples_leaf=1, min_samples_split=7, n_estimators=120;, score=0.846 total time=
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=5, min_samples_split=6, n_estimators=90;, score=0.809 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=5, min_samples_split=6, n_estimators=90;, score=0.799 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=5, min_samples_split=6, n_estimators=90;, score=0.811 total time=
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=20;, score=0.817 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=20;, score=0.791 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=20;, score=0.799 total time=
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=160;, score=0.824 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=160;, score=0.823 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=160;, score=0.823 total time=
[CV 1/3] END max_depth=20, max_features=0.5, min_samples_leaf=1, min_samples_split=8, n_estimators=10;, score=0.885 total time=   0
[CV 2/3] END max_depth=20, max_features=0.5, min_samples_leaf=1, min_samples_split=8, n_estimators=10;, score=0.851 total time=   0
[CV 3/3] END max_depth=20, max_features=0.5, min_samples_leaf=1, min_samples_split=8, n_estimators=10;, score=0.842 total time=   0
[CV 1/3] END max_depth=None, max_features=1.0, min_samples_leaf=3, min_samples_split=2, n_estimators=140;, score=0.879 total time=
[CV 2/3] END max_depth=None, max_features=1.0, min_samples_leaf=3, min_samples_split=2, n_estimators=140;, score=0.861 total time=
[CV 3/3] END max_depth=None, max_features=1.0, min_samples_leaf=3, min_samples_split=2, n_estimators=140;, score=0.847 total time=
[CV 1/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=2, n_estimators=100;, score=0.884 total time=
[CV 2/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=2, n_estimators=100;, score=0.865 total time=
[CV 3/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=2, n_estimators=100;, score=0.860 total time=
[CV 1/3] END max_depth=20, max_features=1.0, min_samples_leaf=7, min_samples_split=7, n_estimators=10;, score=0.882 total time=   0
[CV 2/3] END max_depth=20, max_features=1.0, min_samples_leaf=7, min_samples_split=7, n_estimators=10;, score=0.854 total time=   0
[CV 3/3] END max_depth=20, max_features=1.0, min_samples_leaf=7, min_samples_split=7, n_estimators=10;, score=0.834 total time=   0
[CV 1/3] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=7, n_estimators=150;, score=0.818 total time=
[CV 2/3] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=7, n_estimators=150;, score=0.814 total time=
[CV 3/3] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=7, n_estimators=150;, score=0.824 total time=
[CV 1/3] END max_depth=None, max_features=1.0, min_samples_leaf=8, min_samples_split=2, n_estimators=20;, score=0.880 total time=
[CV 2/3] END max_depth=None, max_features=1.0, min_samples_leaf=8, min_samples_split=2, n_estimators=20;, score=0.863 total time=
[CV 3/3] END max_depth=None, max_features=1.0, min_samples_leaf=8, min_samples_split=2, n_estimators=20;, score=0.850 total time=
```

RandomizedSearchCV

best_estimator_: RandomForestRegressor

RandomForestRegressor

```
sample_rs_model.best_params_
```

```
{'n_estimators': np.int64(100),
 'min_samples_split': np.int64(2),
 'min_samples_leaf': np.int64(6),
 'max_features': 0.5,
 'max_depth': None}
```

```
sample_rs_model_scores = show_scores(model=sample_rs_model,
                                     train_features=x_sample_train,
                                     train_labels=y_sample_train,
                                     test_features=x_sample_test,
                                     test_labels=y_sample_test)
sample_rs_model_scores
```

```
{'Training MAE': 0.23123880501433647,
 'Test MAE': 0.37506713463342095,
```

```
 'Training RMSLE': 0.07391614223660027,
 'Test RMSLE': 0.13168306425054546,
 'Training R^2': 0.9402052980423313,
 'Test R^2': 0.8627250829412689}
```

This code block performs a Grid Search to find the best hyperparameters for a RandomForestRegressor model using the sample training data.

```
np.random.seed(42)

rf_grid = {"n_estimators": np.arange(50, 150, 50),
           "max_depth": [None, 10, 20],
           "min_samples_split": np.arange(2, 8, 2),
           "min_samples_leaf": np.arange(2, 6, 2),
           "max_features": [0.5, "sqrt"],
           "oob_score": [True, False]
}

sample_gs_model = GridSearchCV(estimator=RandomForestRegressor(n_jobs=-1),
                               param_grid=rf_grid,
                               cv=3,
                               verbose=3,
                               n_jobs=-1)

sample_gs_model.fit(X=x_sample_train,
                    y=y_sample_train)
```

```
Fitting 3 folds for each of 144 candidates, totalling 432 fits
```

GridSearchCV

best_estimator_: RandomForestRegressor

RandomForestRegressor

```
sample_gs_model.best_params_
```

```
{'max_depth': 10,
 'max_features': 0.5,
 'min_samples_leaf': np.int64(4),
 'min_samples_split': np.int64(6),
 'n_estimators': np.int64(100),
 'oob_score': False}
```

```
sample_gs_model_scores = show_scores(model=sample_rs_model,
                                     train_features=x_sample_train,
                                     train_labels=y_sample_train,
                                     test_features=x_sample_test,
                                     test_labels=y_sample_test)
sample_gs_model_scores
```

```
{'Training MAE': 0.23123880501433647,
 'Test MAE': 0.37506713463342095,
 'Training RMSLE': 0.07391614223660027,
 'Test RMSLE': 0.13168306425054546,
 'Training R^2': 0.9402052980423313,
 'Test R^2': 0.8627250829412689}
```

This code block sets up and runs a HalvingRandomSearchCV for a RandomForestRegressor model. The goal is to find the best combination of hyperparameters for the model efficiently.

```
np.random.seed(42)

rf_grid = {"n_estimators": np.arange(10, 200, 10),
           "max_depth": [None, 10, 20],
           "min_samples_split": np.arange(2, 10, 1),
           "min_samples_leaf": np.arange(1, 10, 1),
           "max_features": [0.5, 1.0, "sqrt"],
           "oob_score": [True, False]}

sample_hrs_model = HalvingRandomSearchCV(estimator=RandomForestRegressor(n_jobs=-1),
                                         param_distributions=rf_grid,
                                         cv=3,
                                         verbose=3,
                                         n_jobs=-1)

```

```
sample_hrs_model.fit(X=x_sample_train,
```

```
n_iterations: 5
n_required_iterations: 5
n_possible_iterations: 5
min_resources_: 6
max_resources_: 800
aggressive_elimination: False
factor: 3
----------
iter: 0
n_candidates: 133
n_resources: 6
Fitting 3 folds for each of 133 candidates, totalling 399 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan nan
 nan nan nan nan nan nan nan]
  warnings.warn(
----------
iter: 1
n_candidates: 45
n_resources: 18
Fitting 3 folds for each of 45 candidates, totalling 135 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan  0.10786252  0.13421061 -0.24447727  0.26945767  0.1427421
 -0.36087642  0.26545823  0.09048162 -0.25401627 -0.24968605  0.02695616
 -0.23411552 -0.25570336  0.09131135 -0.24427581  0.24359104 -0.25845888
 -0.25625096 -0.25177534  0.07310319 -0.25924247 -0.2637557  -0.24320169
  0.29481288 -0.19327707  0.22760937 -0.08905875  0.24598839 -0.25340653
 -0.26013536 -0.25973454 -0.23370457 -0.25224212 -0.2248123   0.01595762
  0.27578645  0.15987199  0.29780949  0.29580015 -0.17629565 -0.26387836
 -0.2251651  -0.26009043 -0.27193243 -0.1913232 ]
  warnings.warn(
----------
iter: 2
n_candidates: 15
n_resources: 54
Fitting 3 folds for each of 15 candidates, totalling 45 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan         nan         nan         nan         nan         nan
        nan  0.10786252  0.13421061 -0.24447727  0.26945767  0.1427421
 -0.36087642  0.26545823  0.09048162 -0.25401627 -0.24968605  0.02695616
 -0.23411552 -0.25570336  0.09131135 -0.24427581  0.24359104 -0.25845888
 -0.25625096 -0.25177534  0.07310319 -0.25924247 -0.2637557  -0.24320169
  0.29481288 -0.19327707  0.22760937 -0.08905875  0.24598839 -0.25340653
 -0.26013536 -0.25973454 -0.23370457 -0.25224212 -0.2248123   0.01595762
```

```
  0.27578645  0.15987199  0.29780949  0.29580015 -0.17629565 -0.26387836
 -0.2251651  -0.26009043 -0.27193243 -0.1913232   0.76701239  0.74910461
  0.52336839  0.57459067  0.52383073  0.76590687  0.75135845  0.753656
  0.76473236  0.77646417  0.75596013  0.75774724  0.7604571   0.77712615
  0.76715118]
  warnings.warn(
----------
iter: 3
n_candidates: 5
n_resources: 162
Fitting 3 folds for each of 5 candidates, totalling 15 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan  0.10786252  0.13421061 -0.24447727  0.26945767  0.1427421
 -0.36087642  0.26545823  0.09048162 -0.25401627 -0.24968605  0.02695616
 -0.23411552 -0.25570336  0.09131135 -0.24427581  0.24359104 -0.25845888
 -0.25625096 -0.25177534  0.07310319 -0.25924247 -0.2637557  -0.24320169
  0.29481288 -0.19327707  0.22760937 -0.08905875  0.24598839 -0.25340653
 -0.26013536 -0.25973454 -0.23370457 -0.25224212 -0.2248123   0.01595762
  0.27578645  0.15987199  0.29780949  0.29580015 -0.17629565 -0.26387836
 -0.2251651  -0.26009043 -0.27193243 -0.1913232   0.76701239  0.74910461
  0.52336839  0.57459067  0.52383073  0.76590687  0.75135845  0.753656
  0.76473236  0.77646417  0.75596013  0.75774724  0.7604571   0.77712615
  0.76715118  0.80702066  0.8134364   0.8205332   0.81353329  0.81282868]
  warnings.warn(
----------
iter: 4
n_candidates: 2
n_resources: 486
Fitting 3 folds for each of 2 candidates, totalling 6 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan         nan         nan         nan         nan         nan
         nan  0.10786252  0.13421061 -0.24447727  0.26945767  0.1427421
 -0.36087642  0.26545823  0.09048162 -0.25401627 -0.24968605  0.02695616
 -0.23411552 -0.25570336  0.09131135 -0.24427581  0.24359104 -0.25845888
 -0.25625096 -0.25177534  0.07310319 -0.25924247 -0.2637557  -0.24320169
  0.29481288 -0.19327707  0.22760937 -0.08905875  0.24598839 -0.25340653
 -0.26013536 -0.25973454 -0.23370457 -0.25224212 -0.2248123   0.01595762
  0.27578645  0.15987199  0.29780949  0.29580015 -0.17629565 -0.26387836
 -0.2251651  -0.26009043 -0.27193243 -0.1913232   0.76701239  0.74910461
  0.52336839  0.57459067  0.52383073  0.76590687  0.75135845  0.753656
  0.76473236  0.77646417  0.75596013  0.75774724  0.7604571   0.77712615
  0.76715118  0.80702066  0.8134364   0.8205332   0.81353329  0.81282868
  0.86584171  0.86855859]
  warnings.warn(
```

▸ **HalvingRandomSearchCV**

▸ **best_estimator_: RandomForestRegressor**

RandomForestRegressor

```
sample_hrs_model.best_params_
```

```
{'oob_score': True,
 'n_estimators': np.int64(120),
 'min_samples_split': np.int64(7),
 'min_samples_leaf': np.int64(3),
 'max_features': 1.0,
 'max_depth': None}
```

```
sample_hrs_model_scores = show_scores(model=sample_hrs_model,
                                     train_features=x_sample_train,
                                      train_labels=y_sample_train,
                                      test_features=x_sample_test,
                                      test_labels=y_sample_test)
sample_hrs_model_scores
```

```
{'Training MAE': 0.1714868669221164,
 'Test MAE': 0.37873365950620896,
 'Training RMSLE': 0.05815846916238737,
 'Test RMSLE': 0.13302086010875291,
 'Training R^2': 0.9643432868723696,
 'Test R^2': 0.8582603323812968}
```

This code block trains a baseline machine learning model, specifically a Random Forest Regressor, on the full training dataset and then evaluates its performance.

```
base_model = RandomForestRegressor(n_jobs=-1)

base_model.fit(X=X_train, y=y_train)

base_model_scores = show_scores(model=base_model,
                                train_features=X_train,
                                train_labels=y_train,
                                test_features=X_test,
                                test_labels=y_test)
base_model_scores
```

```
{'Training MAE': 0.1147896729302036,
 'Test MAE': 0.30908671413580363,
 'Training RMSLE': 0.03708440936342342,
 'Test RMSLE': 0.09498849454251997,
 'Training R^2': 0.9857995697546297,
 'Test R^2': 0.8990780678226931}
```

This code block focuses on tuning the hyperparameters of a RandomForestRegressor model using Randomized Search Cross-Validation.

```
np.random.seed(42)

rf_grid = {"n_estimators": np.arange(10, 200, 10),
           "max_depth": [None, 10, 20],
           "min_samples_split": np.arange(2, 10, 1),
           "min_samples_leaf": np.arange(1, 10, 1),
           "max_features": [0.5, 1.0, "sqrt"], }

rs_model = RandomizedSearchCV(estimator=RandomForestRegressor(),
                              param_distributions=rf_grid,
                              n_iter=20,
                              cv=3,
                              verbose=3)

rs_model.fit(X_train, y_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
[CV 1/3] END max_depth=10, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=130;, score=0.879 total time=
[CV 2/3] END max_depth=10, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=130;, score=0.881 total time=
[CV 3/3] END max_depth=10, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=130;, score=0.882 total time=
[CV 1/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=7, n_estimators=60;, score=0.898 total time=
[CV 2/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=7, n_estimators=60;, score=0.897 total time=
[CV 3/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=7, n_estimators=60;, score=0.898 total time=
[CV 1/3] END max_depth=10, max_features=0.5, min_samples_leaf=9, min_samples_split=5, n_estimators=140;, score=0.896 total time=  5
[CV 2/3] END max_depth=10, max_features=0.5, min_samples_leaf=9, min_samples_split=5, n_estimators=140;, score=0.895 total time=  4
[CV 3/3] END max_depth=10, max_features=0.5, min_samples_leaf=9, min_samples_split=5, n_estimators=140;, score=0.897 total time=  5
[CV 1/3] END max_depth=10, max_features=0.5, min_samples_leaf=8, min_samples_split=3, n_estimators=50;, score=0.896 total time=  18
[CV 2/3] END max_depth=10, max_features=0.5, min_samples_leaf=8, min_samples_split=3, n_estimators=50;, score=0.894 total time=  18
[CV 3/3] END max_depth=10, max_features=0.5, min_samples_leaf=8, min_samples_split=3, n_estimators=50;, score=0.896 total time=  18
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=7, min_samples_split=7, n_estimators=140;, score=0.890 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=7, min_samples_split=7, n_estimators=140;, score=0.890 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=7, min_samples_split=7, n_estimators=140;, score=0.891 total time=
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=3, n_estimators=180;, score=0.893 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=3, n_estimators=180;, score=0.892 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=3, n_estimators=180;, score=0.893 total time=
[CV 1/3] END max_depth=10, max_features=1.0, min_samples_leaf=2, min_samples_split=7, n_estimators=160;, score=0.896 total time= 1.
[CV 2/3] END max_depth=10, max_features=1.0, min_samples_leaf=2, min_samples_split=7, n_estimators=160;, score=0.894 total time= 1.
[CV 3/3] END max_depth=10, max_features=1.0, min_samples_leaf=2, min_samples_split=7, n_estimators=160;, score=0.897 total time= 2.
[CV 1/3] END max_depth=10, max_features=1.0, min_samples_leaf=6, min_samples_split=3, n_estimators=150;, score=0.896 total time= 1.
[CV 2/3] END max_depth=10, max_features=1.0, min_samples_leaf=6, min_samples_split=3, n_estimators=150;, score=0.894 total time= 1.
[CV 3/3] END max_depth=10, max_features=1.0, min_samples_leaf=6, min_samples_split=3, n_estimators=150;, score=0.897 total time= 1.
[CV 1/3] END max_depth=None, max_features=0.5, min_samples_leaf=4, min_samples_split=2, n_estimators=110;, score=0.899 total time=
[CV 2/3] END max_depth=None, max_features=0.5, min_samples_leaf=4, min_samples_split=2, n_estimators=110;, score=0.897 total time=
[CV 3/3] END max_depth=None, max_features=0.5, min_samples_leaf=4, min_samples_split=2, n_estimators=110;, score=0.899 total time=
[CV 1/3] END max_depth=10, max_features=0.5, min_samples_leaf=3, min_samples_split=2, n_estimators=190;, score=0.896 total time= 1.
[CV 2/3] END max_depth=10, max_features=0.5, min_samples_leaf=3, min_samples_split=2, n_estimators=190;, score=0.895 total time= 1.
[CV 3/3] END max_depth=10, max_features=0.5, min_samples_leaf=3, min_samples_split=2, n_estimators=190;, score=0.897 total time= 1.
[CV 1/3] END max_depth=10, max_features=1.0, min_samples_leaf=1, min_samples_split=7, n_estimators=120;, score=0.896 total time= 1.
[CV 2/3] END max_depth=10, max_features=1.0, min_samples_leaf=1, min_samples_split=7, n_estimators=120;, score=0.894 total time= 1.
[CV 3/3] END max_depth=10, max_features=1.0, min_samples_leaf=1, min_samples_split=7, n_estimators=120;, score=0.896 total time= 1.
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=5, min_samples_split=6, n_estimators=90;, score=0.891 total time=  1
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=5, min_samples_split=6, n_estimators=90;, score=0.891 total time=  1
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=5, min_samples_split=6, n_estimators=90;, score=0.892 total time=  1
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=20;, score=0.888 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=20;, score=0.889 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=3, min_samples_split=8, n_estimators=20;, score=0.888 total time=
[CV 1/3] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=160;, score=0.893 total time=
[CV 2/3] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=160;, score=0.894 total time=
[CV 3/3] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=160;, score=0.894 total time=
[CV 1/3] END max_depth=20, max_features=0.5, min_samples_leaf=1, min_samples_split=8, n_estimators=10;, score=0.890 total time=   6
[CV 2/3] END max_depth=20, max_features=0.5, min_samples_leaf=1, min_samples_split=8, n_estimators=10;, score=0.888 total time=   5
[CV 3/3] END max_depth=20, max_features=0.5, min_samples_leaf=1, min_samples_split=8, n_estimators=10;, score=0.890 total time=   6
[CV 1/3] END max_depth=None, max_features=1.0, min_samples_leaf=3, min_samples_split=2, n_estimators=140;, score=0.898 total time=
[CV 2/3] END max_depth=None, max_features=1.0, min_samples_leaf=3, min_samples_split=2, n_estimators=140;, score=0.897 total time=
[CV 3/3] END max_depth=None, max_features=1.0, min_samples_leaf=3, min_samples_split=2, n_estimators=140;, score=0.899 total time=
[CV 1/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=2, n_estimators=100;, score=0.899 total time=
[CV 2/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=2, n_estimators=100;, score=0.897 total time=
[CV 3/3] END max_depth=None, max_features=0.5, min_samples_leaf=6, min_samples_split=2, n_estimators=100;, score=0.899 total time=
[CV 1/3] END max_depth=20, max_features=1.0, min_samples_leaf=7, min_samples_split=7, n_estimators=10;, score=0.893 total time=   8
[CV 2/3] END max_depth=20, max_features=1.0, min_samples_leaf=7, min_samples_split=7, n_estimators=10;, score=0.891 total time=   9
[CV 3/3] END max_depth=20, max_features=1.0, min_samples_leaf=7, min_samples_split=7, n_estimators=10;, score=0.894 total time=  10
[CV 1/3] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=7, n_estimators=150;, score=0.877 total time=
[CV 2/3] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=7, n_estimators=150;, score=0.881 total time=
[CV 3/3] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=7, n_estimators=150;, score=0.879 total time=
[CV 1/3] END max_depth=None, max_features=1.0, min_samples_leaf=8, min_samples_split=2, n_estimators=20;, score=0.896 total time=
[CV 2/3] END max_depth=None, max_features=1.0, min_samples_leaf=8, min_samples_split=2, n_estimators=20;, score=0.894 total time=
[CV 3/3] END max_depth=None, max_features=1.0, min_samples_leaf=8, min_samples_split=2, n_estimators=20;, score=0.896 total time=
```

▸ **RandomizedSearchCV**

▸ **best_estimator_: RandomForestRegressor**

▸ RandomForestRegressor

```
rs_model.best_params_
```

```
{'n_estimators': np.int64(100),
 'min_samples_split': np.int64(2),
 'min_samples_leaf': np.int64(6),
 'max_features': 0.5,
 'max_depth': None}
```

```
rs_model_scores = show_scores(model=rs_model,
                              train_features=X_train,
                              train_labels=y_train,
                              test_features=X_test,
                              test_labels=y_test)
rs_model_scores
```

```
{'Training MAE': 0.20600592840252008,
 'Test MAE': 0.3070391800429582,
```

```
 'Training RMSLE': 0.06703012962055793,
 'Test RMSLE': 0.09451263235674658,
 'Training R^2': 0.9522656114002397,
 'Test R^2': 0.9004190995040829}
```

This code block takes the best hyperparameters found during a previous randomized search on a sample of the data (sample_rs_model.best_params_) and uses them to train a new RandomForestRegressor model on the full training dataset (X_train, y_train).

```
np.random.seed(42)

rf_best_model_rs = RandomForestRegressor(**sample_rs_model.best_params_, n_jobs=-1)

rf_best_model_rs.fit(X=X_train,
                  y=y_train)

rf_best_model_rs_scores = show_scores(model=rf_best_model_rs,
                                   train_features=X_train,
                                   train_labels=y_train,
                                   test_features=X_test,
                                   test_labels=y_test)

rf_best_model_rs_scores
```

```
{'Training MAE': 0.20628827753240644,
 'Test MAE': 0.30714110904178904,
 'Training RMSLE': 0.06723179714995035,
 'Test RMSLE': 0.0946891465476195,
 'Training R^2': 0.9520839078508789,
 'Test R^2': 0.9001482904069469}
```

This code block trains a machine learning model, specifically a Random Forest Regressor, using the optimal hyperparameters found during a previous Grid Search performed on a sample of the data. It then evaluates the performance of this trained model.

```
np.random.seed(42)

rf_best_model_gs = RandomForestRegressor(**sample_gs_model.best_params_, n_jobs=-1)

rf_best_model_gs.fit(X=X_train,
                  y=y_train)

rf_best_model_gs_scores = show_scores(model=rf_best_model_gs,
                                   train_features=X_train,
                                   train_labels=y_train,
                                   test_features=X_test,
                                   test_labels=y_test)

rf_best_model_gs_scores
```

```
{'Training MAE': 0.2813504894202222,
 'Test MAE': 0.3125488510529735,
 'Training RMSLE': 0.084329907243666,
 'Test RMSLE': 0.0962194210575584,
 'Training R^2': 0.9170215795969648,
 'Test R^2': 0.8966511339215021}
```

This code block trains a machine learning model, specifically a Random Forest Regressor, using the optimal hyperparameters found during a previous Halving Randomized Search performed on a sample of the data. It then evaluates the performance of this trained model.

```
np.random.seed(42)

rf_best_model_hrs = RandomForestRegressor(**sample_hrs_model.best_params_, n_jobs=-1)

rf_best_model_hrs.fit(X=X_train,
                  y=y_train)

rf_best_model_hrs_scores = show_scores(model=rf_best_model_hrs,
                                   train_features=X_train,
                                   train_labels=y_train,
                                   test_features=X_test,
                                   test_labels=y_test)

rf_best_model_hrs_scores
```

```
{'Training MAE': 0.15360517978759128,
 'Test MAE': 0.3075155880619641,
 'Training RMSLE': 0.05238095852765688,
 'Test RMSLE': 0.09486459280397695,
 'Training R^2': 0.9720017938797965,
 'Test R^2': 0.899669019082767}
```

This code block is setting up and running a HalvingRandomSearchCV for a RandomForestRegressor model. The goal is to efficiently find the best combination of hyperparameters for the model by exploring a range of possible values.

```
np.random.seed(42)

rf_grid = {"n_estimators": np.arange(50, 300, 25),
           "max_depth": [10, 20, 30, None],
           "min_samples_split": np.arange(5, 20, 2),
           "min_samples_leaf": np.arange(3, 15, 2),
           "max_features": [0.5, 0.7, 1.0, "sqrt"],
           "bootstrap": [True, False],
           "oob_score": [True, False]}

halving_rs_model = HalvingRandomSearchCV(estimator=RandomForestRegressor(n_jobs=-1),
                                         param_distributions=rf_grid,
                                         cv=5,
                                         verbose=3,
                                         n_jobs=-1)

halving_rs_model.fit(X_train, y_train)
```

```
n_iterations: 8
n_required_iterations: 8
n_possible_iterations: 8
min_resources_: 10
max_resources_: 48739
aggressive_elimination: False
factor: 3
----------
iter: 0
n_candidates: 4873
n_resources: 10
Fitting 5 folds for each of 4873 candidates, totalling 24365 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validation.py:528: FitFailedWarning:
6055 fits failed out of a total of 24365.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
6055 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/ensemble/_forest.py", line 448, in fit
    raise ValueError("Out of bag estimation only available if bootstrap=True")
ValueError: Out of bag estimation only available if bootstrap=True

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are no
 -7.00721460e-04  4.44089210e-17]
  warnings.warn(
----------
iter: 1
n_candidates: 1625
n_resources: 30
Fitting 5 folds for each of 1625 candidates, totalling 8125 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validation.py:528: FitFailedWarning:
2005 fits failed out of a total of 8125.
The score on these train-test partitions for these parameters will be set to nan.
If these failures are not expected, you can try to debug them by setting error_score='raise'.

Below are more details about the failures:
--------------------------------------------------------------------------------
2005 fits failed with the following error:
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_validation.py", line 866, in _fit_and_score
    estimator.fit(X_train, y_train, **fit_params)
  File "/usr/local/lib/python3.11/dist-packages/sklearn/base.py", line 1389, in wrapper
    return fit_method(estimator, *args, **kwargs)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/sklearn/ensemble/_forest.py", line 448, in fit
    raise ValueError("Out of bag estimation only available if bootstrap=True")
ValueError: Out of bag estimation only available if bootstrap=True

  warnings.warn(some_fits_failed_message, FitFailedWarning)
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  0.42957136]
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are no
  6.30447677e-01  6.11430823e-01]
  warnings.warn(
----------
iter: 2
n_candidates: 542
n_resources: 90
Fitting 5 folds for each of 542 candidates, totalling 2710 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are no
 9.44236228e-01 9.41721233e-01]
  warnings.warn(
----------
iter: 3
n_candidates: 181
n_resources: 270
Fitting 5 folds for each of 181 candidates, totalling 905 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are no
 9.56622974e-01 9.28866445e-01]
  warnings.warn(
----------
iter: 4
n_candidates: 61
n_resources: 810
```

```
Fitting 5 folds for each of 61 candidates, totalling 305 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are nc
 9.57169947e-01 9.63778872e-01]
  warnings.warn(
----------
iter: 5
n_candidates: 21
n_resources: 2430
Fitting 5 folds for each of 21 candidates, totalling 105 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are nc
 9.66195706e-01 9.56312011e-01]
  warnings.warn(
----------
iter: 6
n_candidates: 7
n_resources: 7290
Fitting 5 folds for each of 7 candidates, totalling 35 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are nc
 9.55223592e-01 9.69503262e-01]
  warnings.warn(
----------
iter: 7
n_candidates: 3
n_resources: 21870
Fitting 5 folds for each of 3 candidates, totalling 15 fits
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the test scores are nor
  warnings.warn(
/usr/local/lib/python3.11/dist-packages/sklearn/model_selection/_search.py:1108: UserWarning: One or more of the train scores are nc
 9.65005180e-01 9.68384938e-01]
  warnings.warn(
```

HalvingRandomSearchCV

best_estimator_: RandomForestRegressor

RandomForestRegressor

```
halving_rs_model.best_params_
```

```
{'oob_score': False,
 'n_estimators': np.int64(275),
 'min_samples_split': np.int64(9),
 'min_samples_leaf': np.int64(3),
 'max_features': 0.5,
 'max_depth': 30,
 'bootstrap': True}
```

```
halving_rs_model_scores = show_scores(model=halving_rs_model,
                              train_features=X_train,
                              train_labels=y_train,
                              test_features=X_test,
                              test_labels=y_test)
halving_rs_model_scores
```

```
{'Training MAE': 0.17213128569220962,
 'Test MAE': 0.3060477646387894,
 'Training RMSLE': 0.05773162601307981,
 'Test RMSLE': 0.09424251744864517,
 'Training R^2': 0.9656421524008112,
 'Test R^2': 0.9010002768665863}
```

This code block trains a Ridge Regression model. It then evaluates the trained model's performance using show_scores function and prints the resulting scores.

```
ridge_model = Ridge()

ridge_model.fit(X_train, y_train)

ridge_scores = show_scores(model=ridge_model,
                           train_features=X_train,
                           train_labels=y_train,
                           test_features=X_test,
                           test_labels=y_test)

print("Ridge Regression Scores:")
ridge_scores
```

```
Ridge Regression Scores:
/usr/local/lib/python3.11/dist-packages/sklearn/linear_model/_ridge.py:215: LinAlgWarning: Ill-conditioned matrix (rcond=2.93358e-2
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
{'Training MAE': 0.5053638001571152,
 'Test MAE': 0.5116727139965214,
 'Training RMSLE': 0.16089547197896378,
 'Test RMSLE': 0.16337398324994443,
 'Training R^2': 0.715090531088409,
 'Test R^2': 0.713378505293234}
```

This code block trains and evaluates a Support Vector Regression (SVR) model with a Radial Basis Function (RBF) kernel.

```
svr_model = SVR(kernel="rbf")

svr_model.fit(X_train, y_train)

svr_scores = show_scores(model=svr_model,
                         train_features=X_train,
                         train_labels=y_train,
                         test_features=X_test,
                         test_labels=y_test)

print("SVR (RBF Kernel) Scores:")
svr_scores
```

```
SVR (RBF Kernel) Scores:
{'Training MAE': 0.9779449940783711,
 'Test MAE': 0.99666998420428,
 'Training RMSLE': 0.2744357301503668,
 'Test RMSLE': 0.27800511623576585,
 'Training R^2': -0.001117245209311335,
 'Test R^2': -0.003981500634482105}
```

This code block trains and evaluates a Gradient Boosting Regressor.

```
gbr_model = GradientBoostingRegressor(random_state=42)

gbr_model.fit(X_train, y_train)

gbr_scores = show_scores(model=gbr_model,
                         train_features=X_train,
                         train_labels=y_train,
                         test_features=X_test,
                         test_labels=y_test)

print("Gradient Boosting Regressor Scores:")
gbr_scores
```

```
Gradient Boosting Regressor Scores:
{'Training MAE': 0.30389162740478387,
 'Test MAE': 0.30984062347150193,
 'Training RMSLE': 0.09282702678541804,
 'Test RMSLE': 0.09590938220808858,
 'Training R^2': 0.9018795942785915,
 'Test R^2': 0.8984747976960277}
```

This code block focuses on training and evaluating a Lasso Regression model.

```
lasso_model = Lasso(random_state=42)

lasso_model.fit(X_train, y_train)

lasso_scores = show_scores(model=lasso_model,
                           train_features=X_train,
                           train_labels=y_train,
                           test_features=X_test,
                           test_labels=y_test)
```

```
                              test_labels=y_test)

print("Lasso Regression Scores:")
lasso_scores
```

```
Lasso Regression Scores:
{'Training MAE': 0.5209360283357984,
 'Test MAE': 0.5283622787436629,
 'Training RMSLE': 0.16764985050268763,
 'Test RMSLE': 0.17008630946811817,
 'Training R^2': 0.6900910758890415,
 'Test R^2': 0.6863303337304982}
```

This code block focuses on training and evaluating a Linear Regression model.

```
linear_model = LinearRegression()

linear_model.fit(X_train, y_train)

linear_scores = show_scores(model=linear_model,
                            train_features=X_train,
                            train_labels=y_train,
                            test_features=X_test,
                            test_labels=y_test)

print("Linear Regression Scores:")
linear_scores
```

```
Linear Regression Scores:
{'Training MAE': 0.5059397048393599,
 'Test MAE': 0.5116965593521995,
 'Training RMSLE': 0.16118324740043635,
 'Test RMSLE': 0.16342822192065326,
 'Training R^2': 0.7138698122672583,
 'Test R^2': 0.7127963300318897}
```

This code block trains and evaluates a XGBoost (Extreme Gradient Boosting) Regressor model.

```
base_xgb_model  = xgb.XGBRegressor(objective='reg:squarederror', random_state=42)

base_xgb_model .fit(X_train, y_train)

base_xgb_scores  = show_scores(model=base_xgb_model ,
                         train_features=X_train,
                         train_labels=y_train,
                         test_features=X_test,
                         test_labels=y_test)

print("XGBoost Regressor Scores:")
base_xgb_scores
```

```
XGBoost Regressor Scores:
{'Training MAE': 0.2536544976611989,
 'Test MAE': 0.30436832038810135,
 'Training RMSLE': 0.0752648563822963,
 'Test RMSLE': 0.09283392402772897,
 'Training R^2': 0.9317988288515144,
 'Test R^2': 0.9031307536741923}
```

This code block trains and evaluates a LightGBM Regressor model.

```
base_lgb_model  = lgb.LGBMRegressor(objective='regression', random_state=42)

base_lgb_model .fit(X_train, y_train)

base_lgb_scores = show_scores(model=base_lgb_model ,
                          train_features=X_train,
                          train_labels=y_train,
                          test_features=X_test,
                          test_labels=y_test)

print("LightGBM Regressor Scores:")
base_lgb_scores
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.045962 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7622
[LightGBM] [Info] Number of data points in the train set: 48739, number of used features: 64
[LightGBM] [Info] Start training from score 4.661177
LightGBM Regressor Scores:
```

```
{'Training MAE': 0.2870195235970119,
 'Test MAE': 0.3023409923759415,
 'Training RMSLE': 0.086228159451186,
 'Test RMSLE': 0.09277016927780582,
 'Training R^2': 0.913294810653629,
 'Test R^2': 0.9034944103523949}
```

This code block trains and evaluates a Decision Tree Regressor model.

```
dt_model = DecisionTreeRegressor(random_state=42)

dt_model.fit(X_train, y_train)

dt_scores = show_scores(model=dt_model,
                        train_features=X_train,
                        train_labels=y_train,
                        test_features=X_test,
                        test_labels=y_test)

print("Decision Tree Regression Scores:")
dt_scores
```

```
Decision Tree Regression Scores:
{'Training MAE': 2.733473459755407e-19,
 'Test MAE': 0.4369118147038949,
 'Training RMSLE': 4.609051551627045e-18,
 'Test RMSLE': 0.13364573426272688,
 'Training R^2': 1.0,
 'Test R^2': 0.7966488688870226}
```

Because LightGBM Regressor and XGBoost regressor produced the best score, we will try to tune them to get an even better score.

This code block focuses on tuning the hyperparameters of a LightGBM Regressor model using Randomized Search Cross-Validation.

```
np.random.seed(42)

lgb_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'num_leaves': [31, 50, 100],
    'max_depth': [-1, 10, 20],
    'min_child_samples': [20, 50, 100],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
}

tuned_lgb_model = RandomizedSearchCV(estimator=lgb.LGBMRegressor(random_state=42),
                                     param_distributions=lgb_grid,
                                     n_iter=20,
                                     cv=3,
                                     verbose=3,
                                     n_jobs=-1)

tuned_lgb_model.fit(X=X_train, y=y_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.157310 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 7622
[LightGBM] [Info] Number of data points in the train set: 48739, number of used features: 64
[LightGBM] [Info] Start training from score 4.661177
```

```
▸ RandomizedSearchCV
▸ best_estimator_: LGBMRegressor
  ▸ LGBMRegressor
```

```
tuned_lgb_model.best_params_
```

```
{'subsample': 0.8,
 'num_leaves': 50,
 'n_estimators': 200,
 'min_child_samples': 100,
 'max_depth': 20,
 'learning_rate': 0.1,
 'colsample_bytree': 1.0}
```

```
tuned_lgb_scores = show_scores(model=tuned_lgb_model,
                               train_features=X_train,
                               train_labels=y_train,
                               test_features=X_test,
                               test_labels=y_test)
tuned_lgb_scores
```

```
{'Training MAE': 0.2632752187154248,
 'Test MAE': 0.3004680227127976,
 'Training RMSLE': 0.07818204511923915,
 'Test RMSLE': 0.0914804596951764,
 'Training R^2': 0.9273158272823236,
 'Test R^2': 0.9053268766949052}
```

This code block focuses on tuning the hyperparameters of an XGBoost Regressor model using Randomized Search Cross-Validation.

```
np.random.seed(42)

xgb_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'min_child_weight': [1, 3, 5],
    'gamma': [0, 0.1, 0.2],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0],
    'reg_alpha': [0, 0.1, 0.5],
    'reg_lambda': [0, 0.1, 0.5],
}

tuned_xgb_model = RandomizedSearchCV(estimator=xgb.XGBRegressor(random_state=42),
                                     param_distributions=xgb_grid,
                                     n_iter=20,
                                     cv=3,
                                     verbose=3,
                                     n_jobs=-1)

tuned_xgb_model.fit(X=X_train, y=y_train)
```

```
Fitting 3 folds for each of 20 candidates, totalling 60 fits
RandomizedSearchCV
best_estimator_:
XGBRegressor
XGBRegressor
```

```
tuned_xgb_model.best_params_
```

```
{'subsample': 1.0,
 'reg_lambda': 0.5,
 'reg_alpha': 0.1,
 'n_estimators': 200,
 'min_child_weight': 1,
 'max_depth': 5,
 'learning_rate': 0.1,
 'gamma': 0,
 'colsample_bytree': 1.0}
```

```
tuned_xgb_scores = show_scores(model=tuned_xgb_model,
                               train_features=X_train,
                               train_labels=y_train,
                               test_features=X_test,
                               test_labels=y_test)
tuned_xgb_scores
```

```
{'Training MAE': 0.27987266362220287,
 'Test MAE': 0.3017201220262196,
 'Training RMSLE': 0.08396133552799324,
 'Test RMSLE': 0.09261907676734066,
 'Training R^2': 0.9173263065995793,
 'Test R^2': 0.904209000164871}
```

This code block is used to organize and compare the performance scores of various machine learning models that were trained and evaluated in the preceding cells.

```
base_model_scores["model_name"] = "RandomForestRegressor (Base)"
rs_model_scores["model_name"] = "RandomForestRegressor (Tuned)"
```

```
ridge_scores["model_name"] = "Ridge"
svr_scores["model_name"] = "SVR"
gbr_scores["model_name"] = "GradientBoostingRegressor"
lasso_scores["model_name"] = "Lasso"
linear_scores["model_name"] = "LinearRegression"
base_xgb_scores["model_name"] = "XGBoost (Base)"
base_lgb_scores["model_name"] = "LightGBM (Base)"
tuned_xgb_scores["model_name"] = "XGBoost (Tuned)"
tuned_lgb_scores["model_name"] = "LightGBM (Tuned)"
dt_scores["model_name"] = "DecisionTreeRegressor"
rf_best_model_rs_scores["model_name"] = "RandomForestRegressor (Random Search Tuned)"
rf_best_model_gs_scores["model_name"] = "RandomForestRegressor (Grid Search Tuned)"
halving_rs_model_scores["model_name"] = "RandomForestRegressor (Halving Random Search Tuned)"


all_model_scores = [base_model_scores,
                    rs_model_scores,
                    rf_best_model_rs_scores,
                    rf_best_model_gs_scores,
                    halving_rs_model_scores,
                    ridge_scores,
                    svr_scores,
                    gbr_scores,
                    lasso_scores,
                    linear_scores,
                    base_xgb_scores,
                    base_lgb_scores,
                    tuned_xgb_scores,
                    tuned_lgb_scores,
                    dt_scores]

model_comparison_df = pd.DataFrame(all_model_scores).sort_values(by="Test R^2", ascending=True)
model_comparison_df
```

	Training MAE	Test MAE	Training RMSLE	Test RMSLE	Training R^2	Test R^2	model_name
6	9.779450e-01	0.996670	2.744357e-01	0.278005	-0.001117	-0.003982	SVR
8	5.209360e-01	0.528362	1.676499e-01	0.170086	0.690091	0.686330	Lasso
9	5.059397e-01	0.511697	1.611832e-01	0.163428	0.713870	0.712796	LinearRegression
5	5.053638e-01	0.511673	1.608955e-01	0.163374	0.715091	0.713379	Ridge
14	2.733473e-19	0.436912	4.609052e-18	0.133646	1.000000	0.796649	DecisionTreeRegressor
3	2.813505e-01	0.312549	8.432991e-02	0.096219	0.917022	0.896651	RandomForestRegressor (Grid Search Tuned)
7	3.038916e-01	0.309841	9.282703e-02	0.095909	0.901880	0.898475	GradientBoostingRegressor
0	1.147897e-01	0.309087	3.708441e-02	0.094988	0.985800	0.899078	RandomForestRegressor (Base)
2	2.062883e-01	0.307141	6.723180e-02	0.094689	0.952084	0.900148	RandomForestRegressor (Random Search Tuned)
1	2.060059e-01	0.307039	6.703013e-02	0.094513	0.952266	0.900419	RandomForestRegressor (Tuned)
4	1.721313e-01	0.306048	5.773163e-02	0.094243	0.965642	0.901000	RandomForestRegressor (Halving Random Search T...
10	2.536545e-01	0.304368	7.526486e-02	0.092834	0.931799	0.903131	XGBoost (Base)
11	2.870195e-01	0.302341	8.622816e-02	0.092770	0.913295	0.903494	LightGBM (Base)
12	2.798727e-01	0.301720	8.396134e-02	0.092619	0.917326	0.904209	XGBoost (Tuned)
13	2.632752e-01	0.300468	7.818205e-02	0.091480	0.927316	0.905327	LightGBM (Tuned)

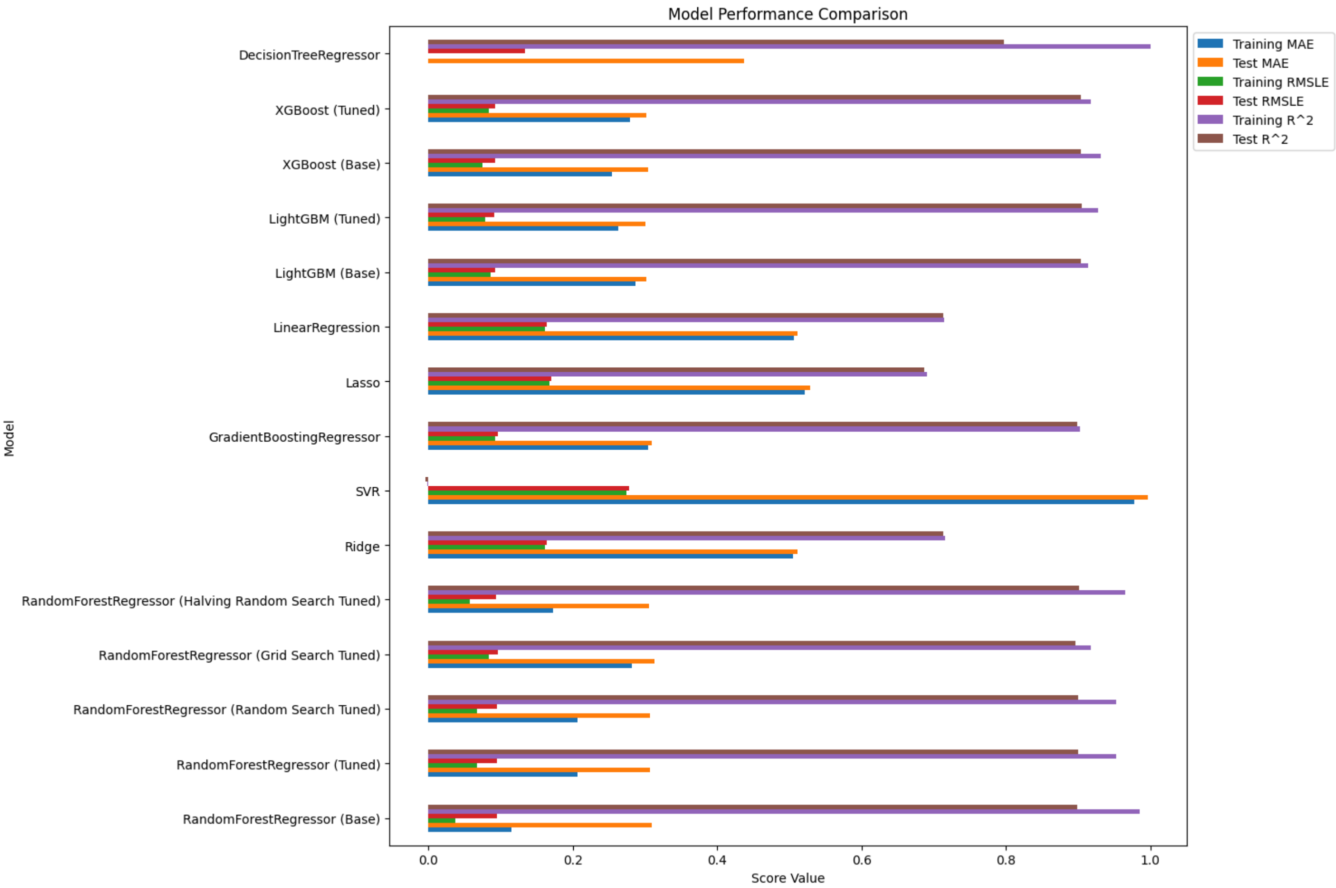
This code block focuses on visualizing the performance of the various machine learning models trained in the previous steps. It creates a bar chart to easily compare the scores of each model.

```
model_scores = pd.DataFrame({
    "RandomForestRegressor (Base)": base_model_scores,
    "RandomForestRegressor (Tuned)": rs_model_scores,
    "RandomForestRegressor (Random Search Tuned)": rf_best_model_rs_scores,
    "RandomForestRegressor (Grid Search Tuned)": rf_best_model_gs_scores,
    "RandomForestRegressor (Halving Random Search Tuned)": halving_rs_model_scores,
    "Ridge": ridge_scores,
    "SVR": svr_scores,
    "GradientBoostingRegressor": gbr_scores,
    "Lasso": lasso_scores,
    "LinearRegression": linear_scores,
    "LightGBM (Base)": base_lgb_scores,
    "LightGBM (Tuned)": tuned_lgb_scores,
    "XGBoost (Base)": base_xgb_scores,
    "XGBoost (Tuned)": tuned_xgb_scores,
    "DecisionTreeRegressor": dt_scores
```

```
})

model_scores = model_scores.transpose()

fig, ax = plt.subplots(figsize=(15, 10))
model_scores.plot(kind="barh", ax=ax)
ax.set_title("Model Performance Comparison")
ax.set_xlabel("Score Value")
ax.set_ylabel("Model")
plt.legend(loc="upper left", bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```



This code block focuses on visualizing the performance of the machine learning models trained and evaluated previously. It specifically creates a horizontal bar chart to compare the Test R^2 score for each model.

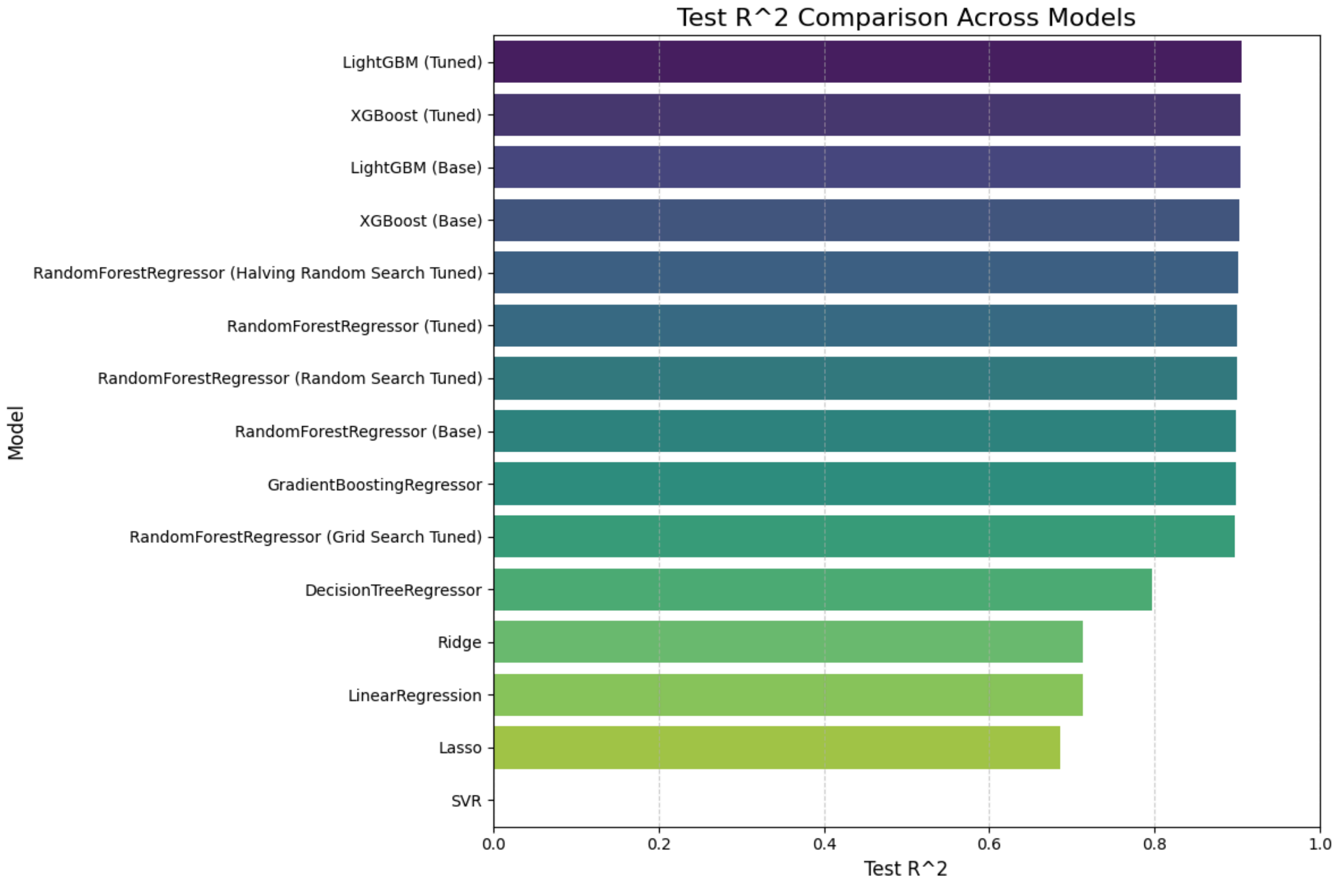
```
model_comparison_df_sorted = model_comparison_df.sort_values(by="Test R^2", ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(x="Test R^2", y="model_name", data=model_comparison_df_sorted, palette="viridis")
plt.title("Test R^2 Comparison Across Models", fontsize=16)
plt.xlabel("Test R^2", fontsize=12)
plt.ylabel("Model", fontsize=12)
plt.xlim(0, 1)
plt.grid(axis='x', linestyle='--', alpha=0.6)
plt.tight_layout()
plt.show()
```

```
<ipython-input-87-43a2ba204d6a>:4: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `le

  sns.barplot(x="Test R^2", y="model_name", data=model_comparison_df_sorted, palette="viridis")
```



Conclusion

In this project, we successfully integrated multiple rich datasets — including the World Happiness Report, World Bank indicators, and MyAnimeList user data — to construct a unified, clean, and feature-rich dataset suitable for machine learning applications. The comprehensive data preprocessing phase involved merging diverse datasets on country-level attributes, handling extensive missing values, encoding categorical variables, and engineering new features from user behavior data.

The machine learning workflow involved setting up models to predict and analyze key user engagement metrics such as stats_mean_score (average anime rating given by a user) and user_completed (number of completed anime). A variety of regression models were employed, including:

- Random Forest Regressor
- Ridge Regression
- Gradient Boosting Regressor
- Support Vector Regression (SVR)
- XGBoost
- LightGBM
- Linear Regression
- Decision Tree Regressor

Hyperparameter tuning techniques, such as GridSearchCV, RandomizedSearchCV, and HalvingRandomSearchCV, were implemented to optimize model performance efficiently. Additionally, key performance metrics like Mean Absolute Error (MAE) and Root Mean Squared Log

Error (RMSLE) were used to evaluate and compare models

After comparing model performances based on evaluation metrics such as R^2, Mean Absolute Error (MAE) and Root Mean Squared Log Error (RMSLE), it was found that the tuned LightGBM model consistently outperformed other models for both key targets. It's score was 0.268188 for stats_mean_sccore and 0.905327 for user_completed.

For future work, deploying these models and further refining feature importance analysis could uncover deeper insights into what drives digital engagement across different global contexts.

Introduction

This project investigates the evolving preferences of anime fans within the rapidly expanding global anime market. Once a niche interest, anime has achieved mainstream popularity, driven by its distinctive art styles and compelling narratives. The anime market, valued at USD 31.41 billion in 2023 and projected to reach USD 72.86 billion by 2032 (CAGR 9.8%), necessitates a deeper understanding of consumer trends. This research will analyze data from MyAnimeList, alongside with relevant economic and demographic data from World Bank, to identify key drivers of anime fan preferences. Specifically, this study will explore how factors such as gender. happiness score, country, age, and gdp per capita correlate with user ratings and engagement.

Preperation

Most of the data we use for the machine learning is in google drive. This cell allows us to use the data on Google Drive

```
import os
from os.path import join
from google.colab import drive
drive.mount("/content/drive", force_remount=True)

# to be used a prefix for file I/O
path_prefix = "/content/drive/My Drive/Colab/DSA 210/Project/Data"
```

```
Mounted at /content/drive
```

This cell lists the files in a specified directory and creates two lists based on these filenames.

exceptions_2015 and exceptions_2014 created here will be used in data processing. The year is last year with data on the datalists.

```
worldbank_fname_list_csv = os.listdir(path_prefix)
worldbank_fname_list = [item[:-4] if item.endswith(".csv") else item for item in  os.listdir(path_prefix)]

exceptions_2015 = ["Renewable electricity output (% of total electricity output).csv",]
exceptions_2014 = ["Electric power consumption (kWh per capita).csv"]
len(os.listdir(path_prefix)), len(worldbank_fname_list_csv), len(worldbank_fname_list)
```

```
(55, 55, 55)
```

Check to see if there is any missing data

```
n = os.listdir(path_prefix)
for item in n:
  if item not in worldbank_fname_list_csv:
    print(item)
```

This code block is used to set up authentication for accessing datasets from the Kaggle platform within the Google Colab environment.

```
from google.colab import userdata
os.environ["KAGGLE_KEY"] = userdata.get('KAGGLE_KEY')
os.environ["KAGGLE_USERNAME"] = userdata.get('KAGGLE_USERNAME')
```

This code block imports various Python libraries and modules that are essential for data manipulation, analysis, visualization, and machine learning tasks.

```
### Basic libraries
import math
import datetime
import numpy as np
```