

```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('/content/drive/MyDrive/Project 2/FIFA World Cup.csv')
```

```
df.head()
```

	team	team_code	association	rank	previous_rank	points	previous_points	
0	Brazil	BRA	CONMEBOL	1	1	1841.30	1837.56	
1	Belgium	BEL	UEFA	2	2	1816.71	1821.92	
2	Argentina	ARG	CONMEBOL	3	3	1773.88	1770.65	
3	France	FRA	UEFA	4	4	1759.78	1764.85	
4	England	ENG	UEFA	5	5	1728.47	1737.46	

Next steps: [Generate code with df](#) [New interactive sheet](#)

```
print(df.isnull().sum())
```

```
team          0
team_code     0
association    0
rank          0
previous_rank  0
points        0
previous_points 0
dtype: int64
```

```
df.dtypes
```

	0
team	object
team_code	object
association	object
rank	int64
previous_rank	int64
points	float64
previous_points	float64
dtype:	object

```
df.describe()
```

	rank	previous_rank	points	previous_points	
count	211.00000	211.00000	211.000000	211.000000	
mean	106.00000	106.00000	1220.687346	1220.645592	
std	61.05462	61.05462	253.420954	253.331712	
min	1.00000	1.00000	762.220000	763.820000	
25%	53.50000	53.50000	1007.675000	1008.330000	
50%	106.00000	106.00000	1192.090000	1183.900000	
75%	158.50000	158.50000	1433.465000	1431.790000	
max	211.00000	211.00000	1841.300000	1837.560000	

```
y = df['points']
X = df.drop(['team', 'team_code', 'points'], axis=1)
X = pd.get_dummies(X, columns=['association'], drop_first=True)
```

```
display(X.head())
print(f"Shape of features (X): {X.shape}")
print(f"Shape of target (y): {y.shape}")
```

	rank	previous_rank	previous_points	association_CAF	association_CONCACAF	association_CONMEBOL	association_OFC	association
0	1	1	1837.56	False	False	True	False	
1	2	2	1821.92	False	False	False	False	
2	3	3	1770.65	False	False	True	False	
3	4	4	1764.85	False	False	False	False	
4	5	5	1737.46	False	False	False	False	

Shape of features (X): (211, 8)
Shape of target (y): (211,)

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"X_train shape: {X_train.shape}")
print(f"X_test shape: {X_test.shape}")
print(f"y_train shape: {y_train.shape}")
print(f"y_test shape: {y_test.shape}")
```

X_train shape: (168, 8)
X_test shape: (43, 8)
y_train shape: (168,)
y_test shape: (43,)

```
rf_model = RandomForestRegressor(random_state=42)

param_grid = {
    'n_estimators': [50, 100, 200],
    'max_features': ['sqrt', 'log2'],
    'max_depth': [4, 6, 8, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2]
}

grid_search = GridSearchCV(estimator=rf_model, param_grid=param_grid, cv=5, n_jobs=-1, verbose=2, scoring='r2')

grid_search.fit(X_train, y_train)

print(f"Best parameters found: {grid_search.best_params_}")
print(f"Best cross-validation R-squared score: {grid_search.best_score_:.4f}")
```

Fitting 5 folds for each of 96 candidates, totalling 480 fits
Best parameters found: {'max_depth': None, 'max_features': 'log2', 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100}
Best cross-validation R-squared score: 0.9963

```
best_rf_model = grid_search.best_estimator_

y_pred = best_rf_model.predict(X_test)

r2 = metrics.r2_score(y_test, y_pred)
mae = metrics.mean_absolute_error(y_test, y_pred)
mse = metrics.mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print(f"Model Performance on Test Set:")
print(f"R-squared (R2): {r2:.4f}")
print(f"Mean Absolute Error (MAE): {mae:.4f}")
print(f"Mean Squared Error (MSE): {mse:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse:.4f}")
```

Model Performance on Test Set:
R-squared (R2): 0.9985
Mean Absolute Error (MAE): 6.5218
Mean Squared Error (MSE): 85.6344
Root Mean Squared Error (RMSE): 9.2539

```
from sklearn.feature_selection import SelectKBest, f_regression

k_best_selector = SelectKBest(score_func=f_regression, k=5)

X_new = k_best_selector.fit_transform(X, y)

selected_feature_indices = k_best_selector.get_support(indices=True)
```

```
selected_features_kbest = X.columns[selected_feature_indices]

print(f"Selected features by SelectKBest: {list(selected_features_kbest)}")
```

Selected features by SelectKBest: ['rank', 'previous_rank', 'previous_points', 'association_CONMEBOL', 'association_UEFA']

```
import matplotlib.pyplot as plt
import seaborn as sns

feature_importances = best_rf_model.feature_importances_

feature_importance_df = pd.DataFrame({
    'Feature': X.columns,
    'Importance': feature_importances
})

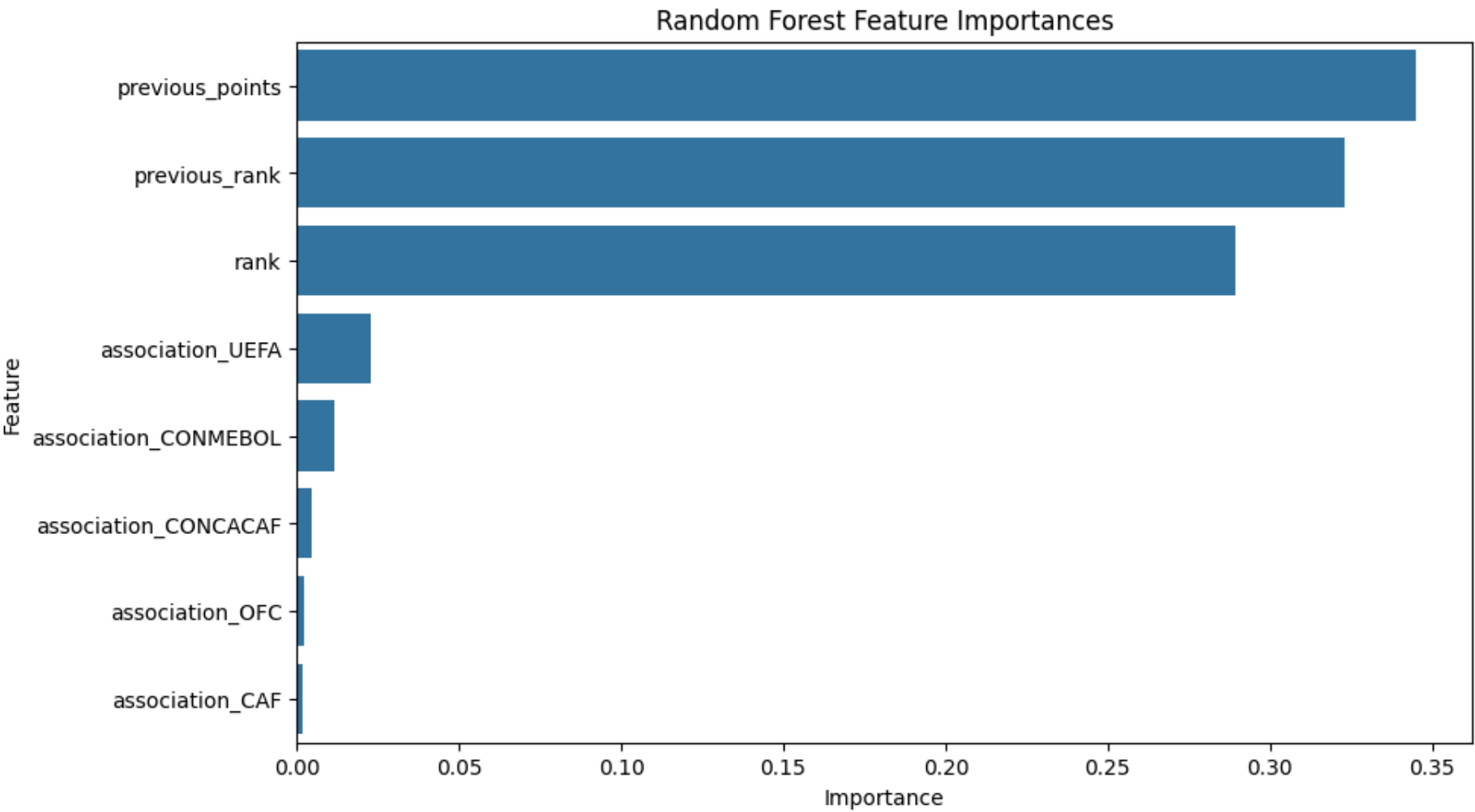
feature_importance_df = feature_importance_df.sort_values(by='Importance', ascending=False)

print("Feature Importances from Random Forest:")
display(feature_importance_df)

plt.figure(figsize=(10, 6))
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Random Forest Feature Importances')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.show()
```

Feature Importances from Random Forest:

	Feature	Importance	
2	previous_points	0.345076	
1	previous_rank	0.322797	
0	rank	0.289295	
7	association_UEFA	0.022683	
5	association_CONMEBOL	0.011664	
4	association_CONCACAF	0.004424	
6	association_OFC	0.002265	
3	association_CAF	0.001795	



Next steps: [Generate code with feature_importance_df](#) [New interactive sheet](#)

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
```

```
X_train_selected, X_test_selected, y_train_selected, y_test_selected = train_test_split(X_new, y, test_size=0.2, random_state=42)
```

```
print(f"X_train_selected shape: {X_train_selected.shape}")
print(f"X_test_selected shape: {X_test_selected.shape}")
```

```
X_train_selected shape: (168, 5)
X_test_selected shape: (43, 5)
```

```
rf_model_selected = RandomForestRegressor(n_estimators=100, max_depth=None, max_features='log2', random_state=42)
```

```
rf_model_selected.fit(X_train_selected, y_train_selected)
```

```
y_pred_selected = rf_model_selected.predict(X_test_selected)
```

```
r2_selected = metrics.r2_score(y_test_selected, y_pred_selected)
mae_selected = metrics.mean_absolute_error(y_test_selected, y_pred_selected)
mse_selected = metrics.mean_squared_error(y_test_selected, y_pred_selected)
rmse_selected = np.sqrt(mse_selected)
```

```
print(f"Model Performance on Test Set with SelectKBest Features:")
print(f"R-squared (R2): {r2_selected:.4f}")
print(f"Mean Absolute Error (MAE): {mae_selected:.4f}")
print(f"Mean Squared Error (MSE): {mse_selected:.4f}")
print(f"Root Mean Squared Error (RMSE): {rmse_selected:.4f}")
```

```
Model Performance on Test Set with SelectKBest Features:
R-squared (R2): 0.9988
Mean Absolute Error (MAE): 5.3881
Mean Squared Error (MSE): 73.4045
Root Mean Squared Error (RMSE): 8.5676
```