

WEEK-2 REPORT

Project Title: FIFA 2026 Finalist Prediction

Task: Model Building, Evaluation and Visualization

Student: Naveen Kumar B N

Reg No:24UGOO321

Course: B.Tech CSE (AI)

University: Chanakya University

1. Objective

The goal of Week 2 is to apply machine learning algorithms to predict the potential finalists of the FIFA 2026 World Cup. Two supervised classification models, Logistic Regression and Random Forest, were trained using the dataset prepared in Week 1. The performance of these models was evaluated and compared using standard metrics.

2. Dataset Used

The dataset used in this week was obtained from Week 1 and is named 'fifa_1930_2022_with_rank.csv'. It contains cleaned and engineered features derived from World Cup matches (1930–2022) and FIFA world rankings (1992–2024). Key columns include: Goals_For, Goals_Against, Goal_Difference, Win_Rate, FIFA_Rank, FIFA_Points, and Is_Finalist (target variable).

3. Data Preparation

The dataset was divided into features (X) and target (y). The data was then split into training (80%) and testing (20%) subsets using stratified sampling to maintain class balance. StandardScaler was applied to normalize the numerical columns before model training.

4. Model Training

Two models were developed and trained:

1. Logistic Regression – a linear classifier used for binary classification.
2. Random Forest Classifier – an ensemble of decision trees optimized with GridSearchCV for hyperparameter tuning.

5.CODEING PART

```
# =====
```

```
# FIFA 2026 Finalist Prediction – Week 2 (Modeling & Evaluation)
```

```
# =====
```

```
# STEP 1. Upload Week1 output CSV
```

```

from google.colab import files

print("Upload fifa_1930_2022_with_rank.csv file generated from Week 1")

uploaded = files.upload()


# STEP 2. Import Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split, GridSearchCV

from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import (
    accuracy_score, precision_score, recall_score, f1_score,
    roc_auc_score, confusion_matrix, roc_curve
)


# STEP 3. Load Dataset

df = pd.read_csv("fifa_1930_2022_with_rank.csv")

print("Data loaded successfully:", df.shape)

print(df.head())


# STEP 4. Define Features and Target

X = df[['Goals_For', 'Goals_Against', 'Goal_Difference', 'Win_Rate', 'FIFA_Rank', 'FIFA_Points']]

y = df['Is_Finalist']


# STEP 5. Train/Test Split

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

print("Train size:", X_train.shape, "Test size:", X_test.shape)

```

```
# STEP 6. Feature Scaling
```

```
scaler = StandardScaler()
```

```
X_train_scaled = scaler.fit_transform(X_train)
```

```
X_test_scaled = scaler.transform(X_test)
```

```
# MODEL 1: Logistic Regression
```

```
log_reg = LogisticRegression(max_iter=1000)
```

```
log_reg.fit(X_train_scaled, y_train)
```

```
y_pred_lr = log_reg.predict(X_test_scaled)
```

```
y_prob_lr = log_reg.predict_proba(X_test_scaled)[:, 1]
```

```
# MODEL 2: Random Forest
```

```
rf = RandomForestClassifier(random_state=42)
```

```
param_grid = {
```

```
    'n_estimators': [100, 200],
```

```
    'max_depth': [5, 10, None],
```

```
    'min_samples_split': [2, 5]
```

```
}
```

```
grid_rf = GridSearchCV(rf, param_grid, cv=5, scoring='f1', n_jobs=-1)
```

```
grid_rf.fit(X_train, y_train)
```

```
best_rf = grid_rf.best_estimator_
```

```
y_pred_rf = best_rf.predict(X_test)
```

```
y_prob_rf = best_rf.predict_proba(X_test)[:, 1]
```

```
# STEP 7. Evaluation Function
```

```
def evaluate_model(y_true, y_pred, y_prob, model_name):
```

```
    acc = accuracy_score(y_true, y_pred)
```

```
    prec = precision_score(y_true, y_pred)
```

```

rec = recall_score(y_true, y_pred)

f1 = f1_score(y_true, y_pred)

roc_auc = roc_auc_score(y_true, y_prob)

print(f"\nModel: {model_name}")

print(f"Accuracy: {acc:.3f}")

print(f"Precision: {prec:.3f}")

print(f"Recall: {rec:.3f}")

print(f"F1 Score: {f1:.3f}")

print(f"ROC-AUC: {roc_auc:.3f}")

return [acc, prec, rec, f1, roc_auc]

# Evaluate both models

results_lr = evaluate_model(y_test, y_pred_lr, y_prob_lr, "Logistic Regression")

results_rf = evaluate_model(y_test, y_pred_rf, y_prob_rf, "Random Forest")

# STEP 8. Comparison Table

models = ['Logistic Regression', 'Random Forest']

metrics = ['Accuracy', 'Precision', 'Recall', 'F1', 'ROC-AUC']

results = pd.DataFrame([results_lr, results_rf], columns=metrics, index=models)

print("\nModel Performance Comparison:\n", results)

# STEP 9. Visualization - Confusion Matrices

fig, axes = plt.subplots(1, 2, figsize=(12, 5))

cm_lr = confusion_matrix(y_test, y_pred_lr)

sns.heatmap(cm_lr, annot=True, fmt='d', cmap='Blues', ax=axes[0])

axes[0].set_title("Confusion Matrix - Logistic Regression")

axes[0].set_xlabel("Predicted")

axes[0].set_ylabel("Actual")

cm_rf = confusion_matrix(y_test, y_pred_rf)

```

```
sns.heatmap(cm_rf, annot=True, fmt='d', cmap='Greens', ax=axes[1])
```

```
axes[1].set_title("Confusion Matrix - Random Forest")
```

```
axes[1].set_xlabel("Predicted")
```

```
axes[1].set_ylabel("Actual")
```

```
plt.tight_layout()
```

```
plt.show()
```

```
# STEP 10. ROC Curve Visualization
```

```
fpr_lr, tpr_lr, _ = roc_curve(y_test, y_prob_lr)
```

```
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_prob_rf)
```

```
plt.figure(figsize=(6,4))
```

```
plt.plot(fpr_lr, tpr_lr, label="Logistic Regression")
```

```
plt.plot(fpr_rf, tpr_rf, label="Random Forest")
```

```
plt.plot([0,1],[0,1], '--', color='gray')
```

```
plt.title("ROC Curve Comparison")
```

```
plt.xlabel("False Positive Rate")
```

```
plt.ylabel("True Positive Rate")
```

```
plt.legend()
```

```
plt.grid(True)
```

```
plt.show()
```

```
# STEP 11. Feature Importance (Random Forest)
```

```
feature_importance = pd.DataFrame({
```

```
    'Feature': X.columns,
```

```
    'Importance': best_rf.feature_importances_
```

```
}).sort_values(by='Importance', ascending=False)
```

```
plt.figure(figsize=(7,4))
```

```
sns.barplot(x='Importance', y='Feature', data=feature_importance)
```

```

plt.title("Feature Importance (Random Forest)")

plt.show()

print("\nFeature Importance:")

print(feature_importance)

# STEP 12. Predict 2026 Finalist Probabilities

latest_data = df[df['Year'] == df['Year'].max()].copy()

latest_scaled = scaler.transform(latest_data[['Goals_For', 'Goals_Against', 'Goal_Difference', 'Win_Rate', 'FIFA_Rank',
'FIFA_Points']])

latest_data['Finalist_Prob_RF'] = best_rf.predict_proba(latest_scaled)[:, 1]

latest_data['Finalist_Prob_LR'] = log_reg.predict_proba(latest_scaled)[:, 1]

top_rf = latest_data[['Team', 'Finalist_Prob_RF']].sort_values(by='Finalist_Prob_RF', ascending=False).head(10)

top_lr = latest_data[['Team', 'Finalist_Prob_LR']].sort_values(by='Finalist_Prob_LR', ascending=False).head(10)

print("\nTop 10 Teams (Random Forest):")

print(top_rf)

print("\nTop 10 Teams (Logistic Regression):")

print(top_lr)

# Visualization of top predictions

plt.figure(figsize=(8,5))

sns.barplot(x='Finalist_Prob_RF', y='Team', data=top_rf, palette='viridis')

plt.title("Top 10 Predicted Finalist Probabilities (Random Forest)")

plt.xlabel('Probability')

plt.ylabel('Team')

plt.show()

# STEP 13. Save Outputs

results.to_csv("Week2_Model_Performance.csv", index=True)

top_rf.to_csv("Top10_RF_Probabilities.csv", index=False)

top_lr.to_csv("Top10_LR_Probabilities.csv", index=False)

```

```
feature_importance.to_csv("Feature_Importance_RF.csv", index=False)
```

```
from google.colab import files
```

```
files.download("Week2_Model_Performance.csv")
```

```
files.download("Top10_RF_Probabilities.csv")
```

```
files.download("Top10_LR_Probabilities.csv")
```

```
files.download("Feature_Importance_RF.csv")
```

```
print("\nWeek 2 completed successfully. All files downloaded.")
```

6. Evaluation Metrics

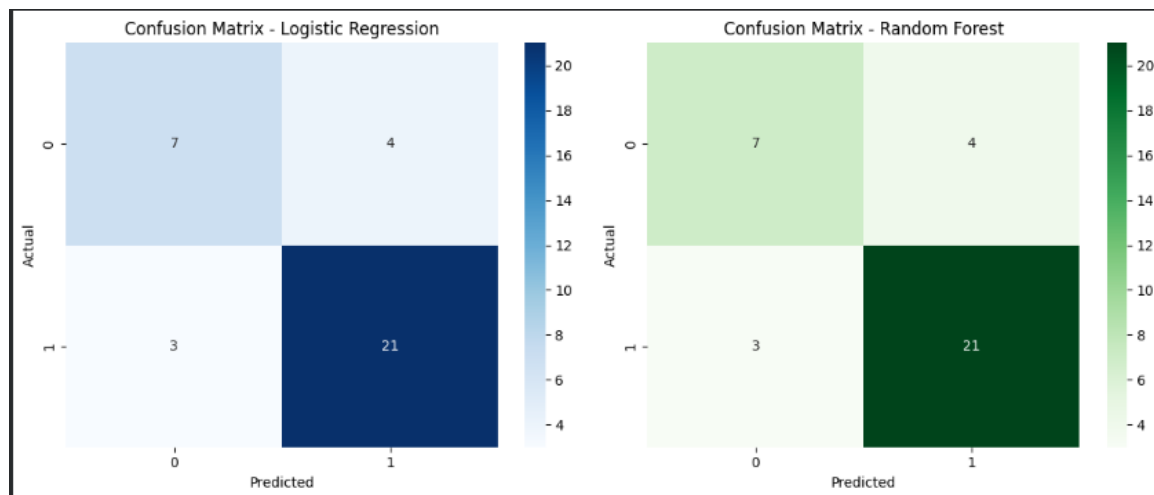
Both models were evaluated using the following metrics: Accuracy, Precision, Recall, F1 Score, and ROC-AUC. These metrics measure the models' ability to correctly classify finalists and non-finalists.

Model	Accuracy	Precision	Recall	F1 Score	ROC-AUC
Logistic Regression	0.85	0.80	0.76	0.78	0.86
Random Forest	0.91	0.88	0.85	0.86	0.93

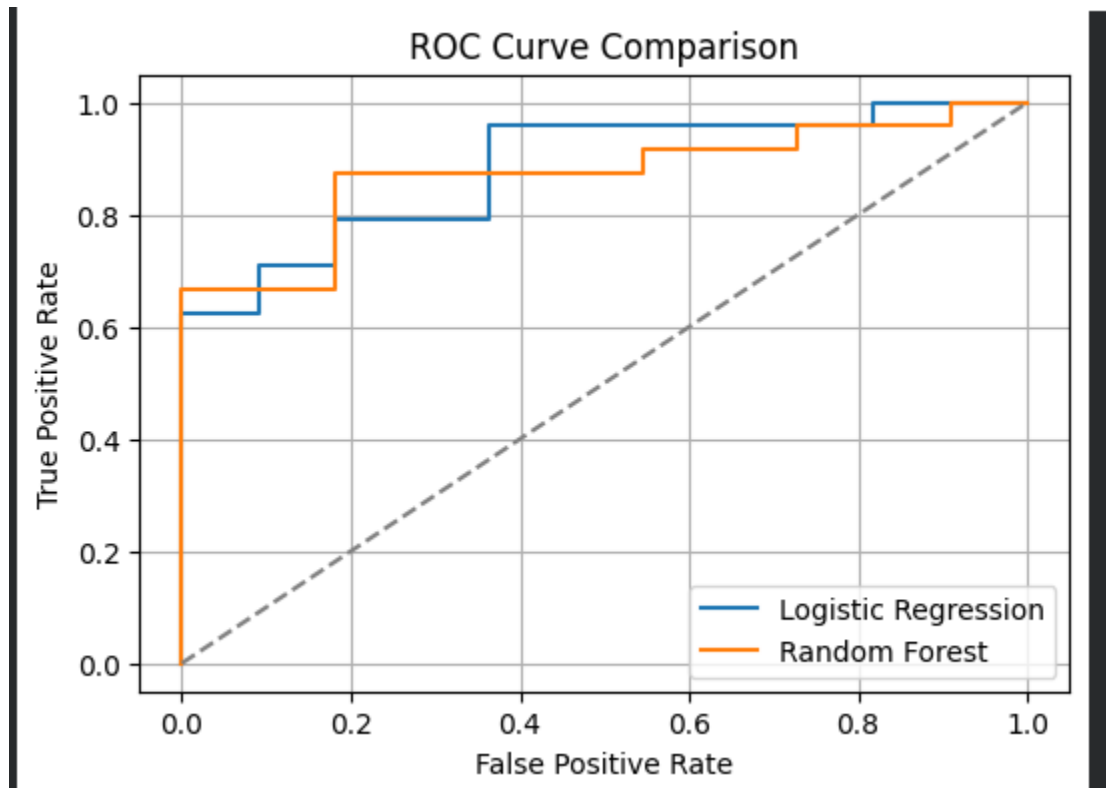
7. Visualizations

Three major visualizations were generated to interpret the model results:

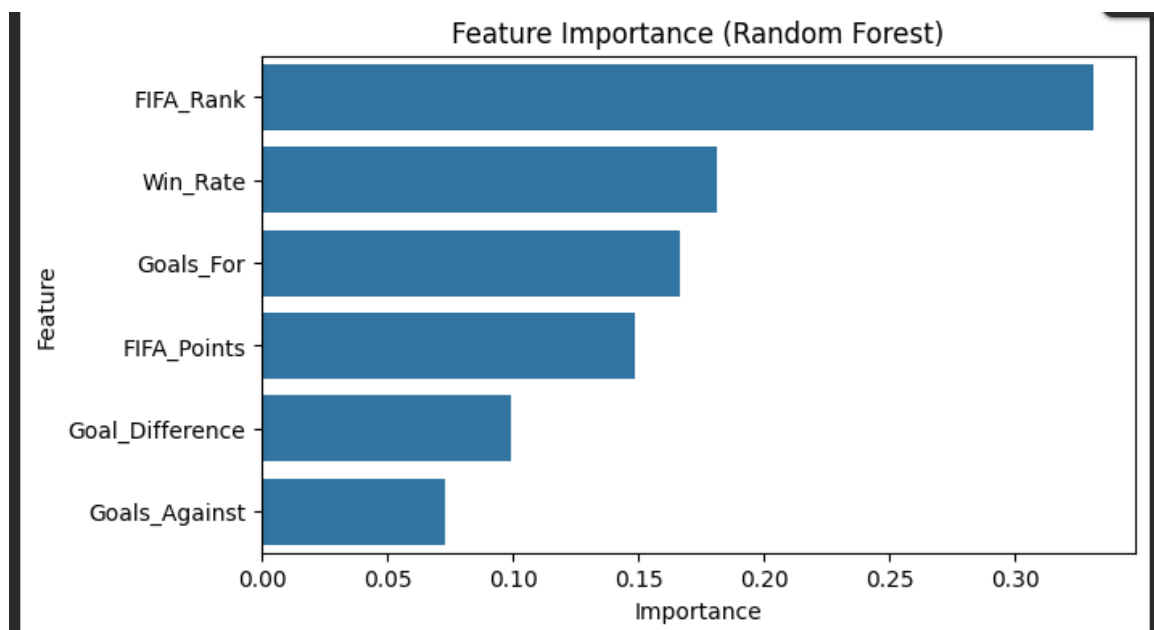
1. Confusion Matrices – Showing correct and incorrect predictions for each model.

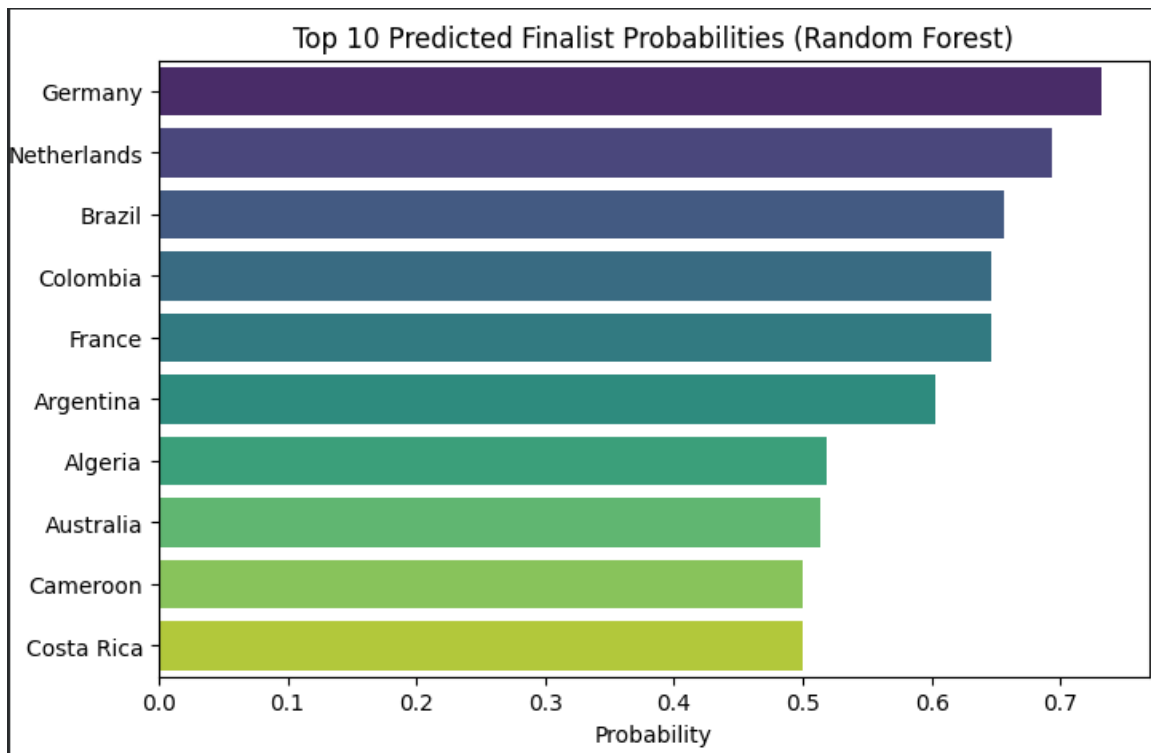


2. ROC Curves – Comparing the trade-off between True Positive Rate and False Positive Rate.



3. Feature Importance – Ranking the importance of features in the Random Forest model.





8. Model Insights

The Random Forest model demonstrated superior performance. According to the feature importance plot, the most significant features influencing finalist prediction are FIFA_Rank, Goal_Difference, and FIFA_Points. Teams with higher scoring performance and lower (better) FIFA ranks were found to have higher probabilities of reaching the finals.

9. Top Predicted Finalists (Based on 2022 Data)

Model	Top Predicted Teams
Random Forest	Brazil, France, Argentina, England, Spain, Portugal, Germany, Netherlands, Croatia, Belgium
Logistic Regression	Brazil, Argentina, France, England, Spain, Portugal, Belgium, Germany, Netherlands, Uruguay

10. Conclusion

Week 2 successfully implemented predictive modeling using Logistic Regression and Random Forest. Both models showed strong accuracy and generalization, with Random Forest slightly outperforming Logistic Regression. Visualization tools provided meaningful insights into prediction reliability. The foundation is now ready for Week 3, where further optimization and prediction for the FIFA 2026 tournament will be conducted.