In [5]:
```
pip install pandas numpy scikit-learn
```

```
Requirement already satisfied: pandas in c:\users\nippu\anaconda3\lib\site-p
ackages (2.2.2)
Requirement already satisfied: numpy in c:\users\nippu\anaconda3\lib\site-pa
ckages (1.26.4)
Requirement already satisfied: scikit-learn in c:\users\nippu\anaconda3\lib
\site-packages (1.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\nippu\anac
onda3\lib\site-packages (from pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\nippu\anaconda3\lib
\site-packages (from pandas) (2024.1)
Requirement already satisfied: tzdata>=2022.7 in c:\users\nippu\anaconda3\li
b\site-packages (from pandas) (2023.3)
Requirement already satisfied: scipy>=1.6.0 in c:\users\nippu\anaconda3\lib
\site-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\nippu\anaconda3\lib
\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in c:\users\nippu\anacon
da3\lib\site-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: six>=1.5 in c:\users\nippu\anaconda3\lib\site
-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [1]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score, mean_squared_error, classificatior
import warnings
warnings.filterwarnings('ignore')
```

In [83]:
```
teams_df = pd.read_csv('teams.csv')  # Assuming the first document is saved as
games_df = pd.read_csv('games.csv')
```

In [85]:
```
# Data Preprocessing
def preprocess_data(teams_df, games_df):
    # Merge team information
    games_df = games_df.merge(teams_df[['team_name', 'team_id']],
                              left_on='team_home', right_on='team_name',
                              how='left').rename(columns={'team_id': 'home_team_
    games_df = games_df.merge(teams_df[['team_name', 'team_id']],
                              left_on='team_away', right_on='team_name',
                              how='left').rename(columns={'team_id': 'away_team_
```

In [87]:
```
games_df['home_win'] = (games_df['score_home'] > games_df['score_away']).astyp
```

In [89]:
```
games_df['total_score'] = games_df['score_home'] + games_df['score_away']
```

In [118]:
```
def preprocess_data(teams_df, games_df):
    # Merge team information
    games_df = games_df.merge(teams_df[['team_name', 'team_id']],
                              left_on='team_home', right_on='team_name',
                              how='left').rename(columns={'team_id': 'home_team_
    games_df = games_df.merge(teams_df[['team_name', 'team_id']],
                              left_on='team_away', right_on='team_name',
                              how='left').rename(columns={'team_id': 'away_team_

    # Create target variables
    games_df['home_win'] = (games_df['score_home'] > games_df['score_away']).a
    games_df['total_score'] = games_df['score_home'] + games_df['score_away']

    # Convert numeric columns to appropriate types
    # Handle over_under_line: convert to float, coerce errors to NaN
    games_df['over_under_line'] = pd.to_numeric(games_df['over_under_line'], e
    games_df['spread_favorite'] = pd.to_numeric(games_df['spread_favorite'], e
    games_df['weather_temperature'] = pd.to_numeric(games_df['weather_temperat
    games_df['weather_wind_mph'] = pd.to_numeric(games_df['weather_wind_mph'],

    # Fill missing values
    games_df['spread_favorite'] = games_df['spread_favorite'].fillna(0)
    games_df['over_under_line'] = games_df['over_under_line'].fillna(games_df[
    games_df['weather_temperature'] = games_df['weather_temperature'].fillna(g
    games_df['weather_wind_mph'] = games_df['weather_wind_mph'].fillna(games_d

    return games_df
```

In [120]:
```
# Feature Engineering
def create_features(games_df):
    def team_stats(team_id, date, df):
        team_games = df[((df['home_team_id'] == team_id) | (df['away_team_id']
                       (df['schedule_date'] < date)]
        recent_games = team_games.sort_values('schedule_date', ascending=False

        if len(recent_games) == 0:
            return pd.Series({
                'avg_points_scored': 0,
                'avg_points_allowed': 0,
                'win_rate': 0
            })

        points_scored = []
        points_allowed = []
        wins = 0

        for _, game in recent_games.iterrows():
            if game['home_team_id'] == team_id:
                points_scored.append(game['score_home'])
                points_allowed.append(game['score_away'])
                wins += 1 if game['score_home'] > game['score_away'] else 0
            else:
                points_scored.append(game['score_away'])
                points_allowed.append(game['score_home'])
                wins += 1 if game['score_away'] > game['score_home'] else 0

        return pd.Series({
            'avg_points_scored': np.mean(points_scored),
            'avg_points_allowed': np.mean(points_allowed),
            'win_rate': wins / len(recent_games)
        })

    # Apply team stats
    games_df[['home_avg_points_scored', 'home_avg_points_allowed', 'home_win_r
        team_stats(row['home_team_id'], row['schedule_date'], games_df)
        for _, row in games_df.iterrows()
    ]

    games_df[['away_avg_points_scored', 'away_avg_points_allowed', 'away_win_r
        team_stats(row['away_team_id'], row['schedule_date'], games_df)
        for _, row in games_df.iterrows()
    ]

    # Additional features
    games_df['point_differential'] = games_df['home_avg_points_scored'] - game
    games_df['is_favorite_home'] = (games_df['team_favorite_id'] == games_df['

    return games_df
```

In [133]:
```
# Train Models
def train_models(games_df):
    features = [
        'home_avg_points_scored', 'home_avg_points_allowed', 'home_win_rate',
        'away_avg_points_scored', 'away_avg_points_allowed', 'away_win_rate',
        'point_differential', 'spread_favorite', 'over_under_line',
        'weather_temperature', 'weather_wind_mph', 'is_favorite_home'
    ]

    # Verify features
    missing_features = [f for f in features if f not in games_df.columns]
    if missing_features:
        raise ValueError(f"Missing features in DataFrame: {missing_features}")

    # Handle missing values
    games_df[features] = games_df[features].fillna(0)

    # Prepare data
    X = games_df[features]  # Corrected line
    y_outcome = games_df['home_win']
    y_total = games_df['total_score']

    # Verify data
    if X.empty or y_outcome.empty or y_total.empty:
        raise ValueError("Empty feature or target data")

    # Split data
    X_train, X_test, y_train_outcome, y_test_outcome = train_test_split(
        X, y_outcome, test_size=0.2, random_state=42
    )

    X_train_total, X_test_total, y_train_total, y_test_total = train_test_spli
        X, y_total, test_size=0.2, random_state=42
    )

    # Outcome Prediction
    outcome_model = RandomForestClassifier(n_estimators=100, random_state=42)
    outcome_model.fit(X_train, y_train_outcome)
    outcome_pred = outcome_model.predict(X_test)

    print("Outcome Prediction Results:")
    print(f"Accuracy: {accuracy_score(y_test_outcome, outcome_pred):.3f}")
    print(classification_report(y_test_outcome, outcome_pred))

    # Over/Under Prediction
    total_model = RandomForestRegressor(n_estimators=100, random_state=42)
    total_model.fit(X_train_total, y_train_total)
    total_pred = total_model.predict(X_test_total)

    print("\nOver/Under Prediction Results:")
    print(f"RMSE: {np.sqrt(mean_squared_error(y_test_total, total_pred)):.3f}"

    # Upset Detection
    def detect_upsets(model, X, y, games_df):
        predictions = model.predict_proba(X)
        actual_results = y

        upset_games = []
        for i, (pred, actual) in enumerate(zip(predictions, actual_results)):
            if pred[0] > 0.7 and actual == 1 and games_df.iloc[i]['is_favorite
                upset_games.append({
```

```
                    'game_id': games_df.iloc[i].name,
                    'home_team': games_df.iloc[i]['team_home'],
                    'away_team': games_df.iloc[i]['team_away'],
                    'upset_probability': pred[0]
                })

        return pd.DataFrame(upset_games)

    upsets = detect_upsets(outcome_model, X_test, y_test_outcome, games_df.ilc
    print("\nPotential Upsets Detected:")
    print(upsets)

    return outcome_model, total_model
```

In [135]:
```
# Main execution
def main():
    # Preprocess data
    processed_df = preprocess_data(teams_df, games_df)

    # Create features
    featured_df = create_features(processed_df)

    # Train models
    outcome_model, total_model = train_models(featured_df)

    # Feature importance
    print("\nFeature Importance for Outcome Prediction:")
    importance = pd.DataFrame({
        'feature': [
            'home_avg_points_scored', 'home_avg_points_allowed', 'home_win_rat
            'away_avg_points_scored', 'away_avg_points_allowed', 'away_win_rat
            'point_differential', 'spread_favorite', 'over_under_line',
            'weather_temperature', 'weather_wind_mph', 'is_favorite_home'
        ],
        'importance': outcome_model.feature_importances_
    }).sort_values('importance', ascending=False)
    print(importance)

if __name__ == "__main__":
    main()
```

```
Outcome Prediction Results:
Accuracy: 0.616
              precision    recall  f1-score   support

           0       0.56      0.46      0.51      1206
           1       0.64      0.73      0.69      1612

    accuracy                           0.62      2818
   macro avg       0.60      0.60      0.60      2818
weighted avg       0.61      0.62      0.61      2818


Over/Under Prediction Results:
RMSE: 13.920

Potential Upsets Detected:
```

	game_id	home_team	away_team	upset_probability
0	12645	New York Jets	Pittsburgh Steelers	0.75
1	10710	Jacksonville Jaguars	Tennessee Titans	0.81
2	7955	St. Louis Rams	Oakland Raiders	0.76
3	7921	Detroit Lions	New Orleans Saints	0.71
4	8629	Miami Dolphins	New England Patriots	0.79
5	5524	Tampa Bay Buccaneers	Chicago Bears	0.79
6	11643	Buffalo Bills	Arizona Cardinals	0.73
7	3776	Kansas City Chiefs	Seattle Seahawks	0.71
8	12830	Houston Texans	New England Patriots	0.71
9	7499	Cleveland Browns	New England Patriots	0.75
10	8661	Washington Redskins	Minnesota Vikings	0.75
11	6319	Atlanta Falcons	San Francisco 49ers	0.87
12	10491	Kansas City Chiefs	Green Bay Packers	0.72
13	11570	Atlanta Falcons	Carolina Panthers	0.79
14	1199	Detroit Lions	Chicago Bears	0.72
15	6714	Arizona Cardinals	Philadelphia Eagles	0.71
16	7590	Atlanta Falcons	Kansas City Chiefs	0.75
17	13929	Carolina Panthers	New Orleans Saints	0.80
18	13269	Indianapolis Colts	Kansas City Chiefs	0.80
19	6750	New Orleans Saints	Seattle Seahawks	0.73
20	6695	Oakland Raiders	Denver Broncos	0.71
21	1921	Detroit Lions	New England Patriots	0.71
22	4441	Houston Oilers	Seattle Seahawks	0.74
23	13313	Atlanta Falcons	San Francisco 49ers	0.76
24	10572	St. Louis Rams	Washington Redskins	0.80
25	6810	Indianapolis Colts	Miami Dolphins	0.78
26	8046	Washington Redskins	St. Louis Rams	0.74
27	10536	San Francisco 49ers	New Orleans Saints	0.77
28	3798	Seattle Seahawks	Los Angeles Raiders	0.76
29	9019	New Orleans Saints	Philadelphia Eagles	0.75
30	2614	Baltimore Colts	New England Patriots	0.82
31	4374	Buffalo Bills	Miami Dolphins	0.73
32	9129	Tennessee Titans	Indianapolis Colts	0.87
33	5863	Los Angeles Rams	Arizona Cardinals	0.76
34	6179	Indianapolis Colts	San Francisco 49ers	0.83
35	7063	New England Patriots	San Francisco 49ers	0.71
36	10468	Arizona Cardinals	San Francisco 49ers	0.74
37	14033	Dallas Cowboys	Tampa Bay Buccaneers	0.80
38	8023	Minnesota Vikings	Green Bay Packers	0.74
39	6147	Detroit Lions	San Francisco 49ers	0.88
40	1981	New England Patriots	Buffalo Bills	0.75
41	13139	Seattle Seahawks	San Francisco 49ers	0.76
42	7631	Cincinnati Bengals	Baltimore Ravens	0.72
43	13784	New York Giants	Philadelphia Eagles	0.72

```
44     9572        St. Louis Rams        Dallas Cowboys              0.84
45     9024  Tampa Bay Buccaneers    Cincinnati Bengals              0.72

Feature Importance for Outcome Prediction:
                   feature  importance
1   home_avg_points_allowed    0.110544
4   away_avg_points_allowed    0.110116
6        point_differential    0.105966
0    home_avg_points_scored    0.105236
3    away_avg_points_scored    0.105044
7           spread_favorite    0.089707
9      weather_temperature    0.087106
8           over_under_line    0.081226
10        weather_wind_mph    0.068816
11        is_favorite_home    0.058045
2            home_win_rate    0.039511
5            away_win_rate    0.038683
```

In [ ]:

In [ ]: