```
!pip install pandas scikit-learn matplotlib seaborn
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages (2.2.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.11/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas) (2025.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.3
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (4.
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (1.
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (24.2
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (11.1.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib) (3.2
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->
```

```
# Import libraries
import pandas as pd
from google.colab import files


# Upload the dataset
uploaded = files.upload()
```

Choose files odi_Matches_Data.csv

- **odi_Matches_Data.csv**(text/csv) - 2577297 bytes, last modified: 09/05/2024 - 100% done

```
Saving odi_Matches_Data.csv to odi_Matches_Data (1).csv
```

```
# Load the dataset into a DataFrame
df = pd.read_csv('odi_Matches_Data.csv')


# Display the first few rows to confirm loading
df.head()
```

	ODI Match No	Match ID	Match Name	Series ID	Series Name	Match Date	Match Format	Team1 ID	Team1 Name	Team1 Captain	...	Umpire 2	Match Referee	Toss Winner	T Wir Cho
0	488	65425	Australia Vs New Zealand 4Th Match	60879.0	Benson & Hedges World Series Cup Australia, Ne...	1988-01-07	ODI	2.0	Australia	1572.0	...	RC Bailhache	NaN	Australia	
1	492	65428	New Zealand Vs Sri Lanka 7Th Match	60879.0	Benson & Hedges World Series Cup Australia, Ne...	1988-01-12	ODI	5.0	New Zealand	1698.0	...	SG Randell	NaN	Sri Lanka	
2	495	65431	Australia Vs New Zealand 10Th Match	60879.0	Benson & Hedges World Series Cup Australia, Ne...	1988-01-17	ODI	5.0	New Zealand	1698.0	...	AR Crafter	NaN	Australia	
3	496	65432	Australia Vs Sri Lanka 11Th Match	60879.0	Benson & Hedges World Series Cup Australia, Ne...	1988-01-19	ODI	8.0	Sri Lanka	1664.0	...	TA Prue	NaN	Australia	
4	508	64326	New Zealand Vs England 3Rd Odi	60882.0	England tour of New Zealand - 1988 (1987/88)	1988-03-16	ODI	1.0	England	1543.0	...	SJ Woodward	NaN	New Zealand	

5 rows × 33 columns

```
# Import preprocessing tools
from sklearn.preprocessing import LabelEncoder


# Select relevant columns
columns_to_keep = ['Toss Winner', 'Toss Winner Choice', 'Match Winner', 'Team1 Name', 'Team2 Name',
                   'Team1 Runs Scored', 'Team2 Runs Scored', 'Match Venue (Stadium)']
df = df[columns_to_keep].copy()


# Handle missing values (drop rows with missing critical data)
df.dropna(subset=['Toss Winner', 'Toss Winner Choice', 'Match Winner'], inplace=True)


# Create target variable: 1 if toss winner won the match, 0 otherwise
df['Toss Winner Won Match'] = (df['Toss Winner'] == df['Match Winner']).astype(int)


# Encode categorical variables
le_toss_winner = LabelEncoder()
le_toss_choice = LabelEncoder()
le_team1 = LabelEncoder()
le_team2 = LabelEncoder()
le_venue = LabelEncoder()

df['Toss Winner'] = le_toss_winner.fit_transform(df['Toss Winner'])
df['Toss Winner Choice'] = le_toss_choice.fit_transform(df['Toss Winner Choice'])
df['Team1 Name'] = le_team1.fit_transform(df['Team1 Name'])
df['Team2 Name'] = le_team2.fit_transform(df['Team2 Name'])
df['Match Venue (Stadium)'] = le_venue.fit_transform(df['Match Venue (Stadium)'])
```

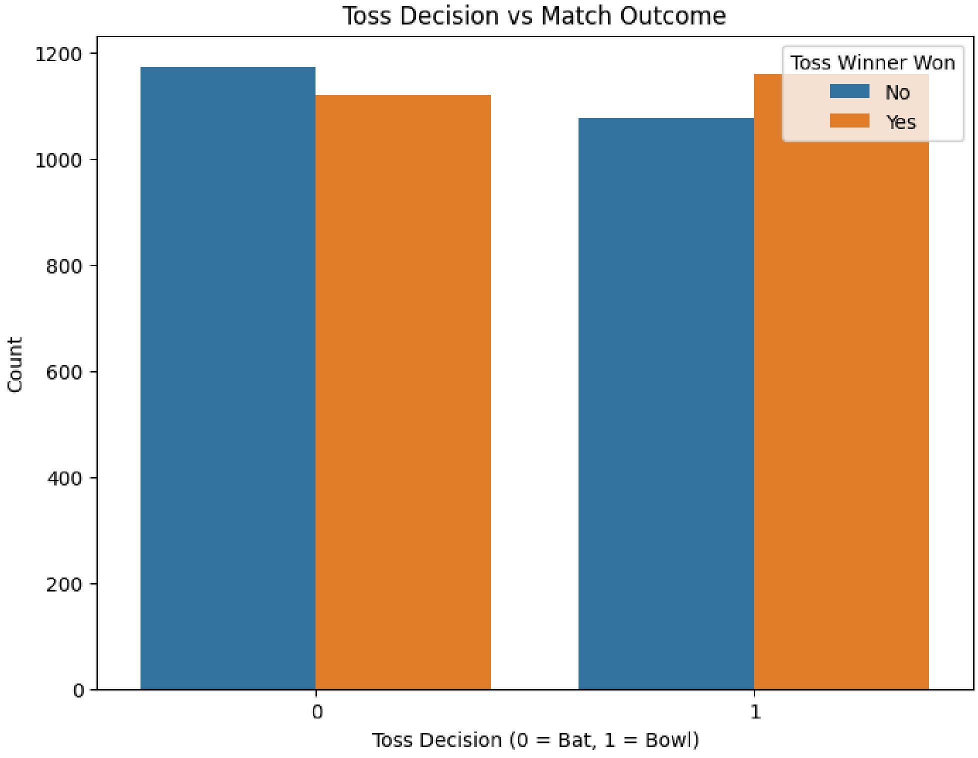
```
# Features and target
X = df[['Toss Winner', 'Toss Winner Choice', 'Team1 Name', 'Team2 Name',
        'Team1 Runs Scored', 'Team2 Runs Scored', 'Match Venue (Stadium)']]
y = df['Toss Winner Won Match']


# Display preprocessed data
print("Preprocessed Data Sample:")
print(X.head())
print("\nTarget Sample:")
print(y.head())
```

```
Preprocessed Data Sample:
   Toss Winner  Toss Winner Choice  Team1 Name  Team2 Name  Team1 Runs Scored  \
0            2                   0           3          18              216.0
1           23                   1          18          24              199.0
2            2                   1          18           3              176.0
3            2                   1          24           3              188.0
4           17                   1           8          18              219.0

   Team2 Runs Scored  Match Venue (Stadium)
0              210.0                    129
1              200.0                     23
2              177.0                     33
3              189.0                    215
4              223.0                    128

Target Sample:
0    1
1    1
2    1
3    1
4    1
Name: Toss Winner Won Match, dtype: int64
```

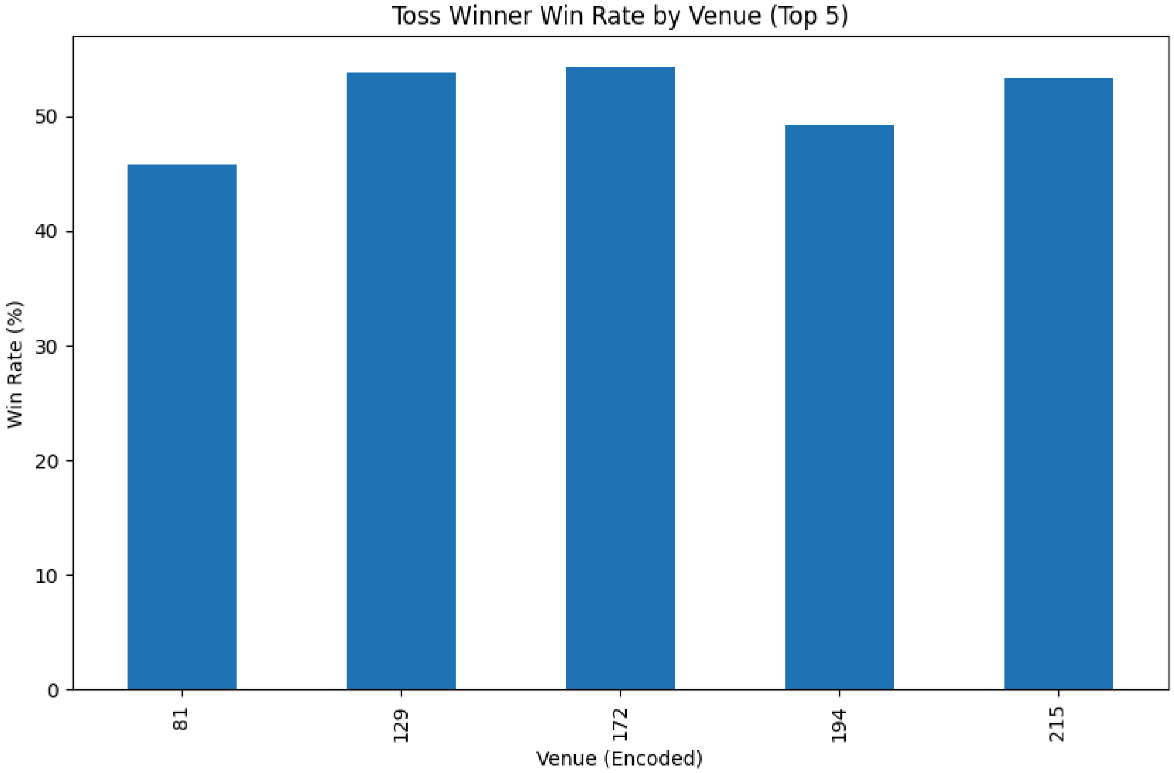
```
# Import visualization libraries
import matplotlib.pyplot as plt
import seaborn as sns


# Percentage of matches won by toss winner
toss_win_rate = y.mean() * 100
print(f"Percentage of matches won by toss winner: {toss_win_rate:.2f}%")
```

```
Percentage of matches won by toss winner: 50.34%
```

```
# Bar plot: Toss decision vs. match outcome
plt.figure(figsize=(8, 6))
sns.countplot(x='Toss Winner Choice', hue='Toss Winner Won Match', data=df)
plt.title('Toss Decision vs Match Outcome')
plt.xlabel('Toss Decision (0 = Bat, 1 = Bowl)')
plt.ylabel('Count')
plt.legend(title='Toss Winner Won', labels=['No', 'Yes'])
plt.show()
```



```
# Venue-based win rate (top 5 venues)
top_venues = df['Match Venue (Stadium)'].value_counts().index[:5]
venue_win_rates = df[df['Match Venue (Stadium)'].isin(top_venues)].groupby('Match Venue (Stadium)')['Toss Winner Won M
plt.figure(figsize=(10, 6))
venue_win_rates.plot(kind='bar')
plt.title('Toss Winner Win Rate by Venue (Top 5)')
plt.xlabel('Venue (Encoded)')
plt.ylabel('Win Rate (%)')
plt.show()
```



```
# Import modeling tools
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, classification_report
```

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

# Initialize and train the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

```
▾     RandomForestClassifier     ⓘ ?
RandomForestClassifier(random_state=42)
```

```
# Make predictions
y_pred = rf_model.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)

print("Model Evaluation Results:")
print(f"Accuracy: {accuracy:.2f}")
print(f"Precision: {precision:.2f}")
print(f"Recall: {recall:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Model Evaluation Results:
Accuracy: 0.86
Precision: 0.86
Recall: 0.87

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.85      0.86       452
           1       0.86      0.87      0.86       455

    accuracy                           0.86       907
   macro avg       0.86      0.86      0.86       907
weighted avg       0.86      0.86      0.86       907
```

```
# Import joblib for saving models
import joblib
```

```
# Save the trained Random Forest model
joblib.dump(rf_model, 'random_forest_model.pkl')
```