Here are some points highlighting why your code can be considered clean, along with specific code lines:

1.**Modularization and Functionality Separation**:
Various parts of my code, like loading the dataset, exploratory data analysis, data cleaning, machine learning, and prediction, have been broken up into meaningful blocks by me. This makes the code simple to peruse and comprehend.
For example:

```
    # Modularization: Exploratory Data Analysis
    sns.displot(df['wickets'], kde=False, bins=10)
    plt.title("Wickets Distribution")
    plt.show()
```

2. **Comments and Documentation**:
I've included remarks all through my code to make sense of the motivation behind each part and explicit tasks.
For example:

```
  # Loading the Dataset
    df=pd.read_csv('/content/ipl_data.csv')
    df.head()
```

3.**Variable Naming**:
Variable names are clear and descriptive, making it easy to understand the purpose of each variable.
For example:

```
    # Variable Naming: Consistent Teams
    consistent_team=['Kolkata Knight Riders', 'Chennai Super Kings',
...]
    df = df[(df['bat_team'].isin(consistent_team)) &
(df['bowl_team'].isin(consistent_team))]
```

4. **Data Cleaning:**
The data cleaning section is well-structured, and irrelevant columns are dropped efficiently.
For example:

```
    # Data Cleaning: Dropping Irrelevant Columns
    df = df.drop(irrelevant, axis=1)
```

5. **Label Encoding and One-Hot Encoding:**
I've used scikit-learn's `LabelEncoder` and `OneHotEncoder` in a concise manner for encoding categorical variables.
For example:

```
    # Label Encoding and One-Hot Encoding
    le = LabelEncoder()
    for col in ['bat_team', 'bowl_team']:
      df[col] = le.fit_transform(df[col])
```

6. **Machine Learning Section**:
I've used different machine learning models and evaluated their performance.
For example:

```
# Machine Learning: Decision Tree Regressor
    tree = DecisionTreeRegressor()
    tree.fit(train_features, train_labels)
```

7. **Predict Function:**
I've encapsulated the prediction logic in a separate function, making it reusable and enhancing code readability.
For example:

```
    # Prediction Function
    def score_predict(batting_team, bowling_team, runs, wickets, overs,
runs_last_5, wickets_last_5, model=forest):
```

8. **Testing and Evaluation**:
I've included a test case to validate the prediction function.
For example:

```
    # Test 1
    batting_team='Mumbai Indians'
    bowling_team='Kings XI Punjab'
    score = score_predict(batting_team, bowling_team, overs=12.3,
runs=113, wickets=2, runs_last_5=55, wickets_last_5=0)
```

9. **Use of Libraries:**
I've utilized popular data science libraries like NumPy, Pandas, Matplotlib, Seaborn, and scikit-learn efficiently.

10. **Consistent Formatting:**
my code maintains consistent formatting throughout, enhancing readability.