MyModel

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
```

In [2]:
```
def get_initial_training_dataframes(training_data):
    ball_by_ball, matches_result = training_data

    ball_by_ball = ball_by_ball.rename(columns={
        'ID': 'match_id',
        'ballnumber': 'ball_number',
        'non-striker': 'non_striker',
        'BattingTeam': 'batting_team',
    }).loc[:, [
        'match_id',
        'innings',
        'batting_team',
        'overs',
        'ball_number',
        'batter',
        'bowler',
        'total_run',
    ]]

    matches_result = matches_result.rename(columns={
        'ID': 'match_id',
        'Team1': 'team_1',
        'Team2': 'team_2',
        'Venue': 'venue',
    }).loc[:, [
        'match_id',
        'team_1',
        'team_2',
        'venue',
    ]]

    return ball_by_ball, matches_result
```

In [3]:
```
venue_mapping = {
 'Arun Jaitley Stadium, Delhi': 'Arun Jaitley Stadium',
 'Arun Jaitley Stadium': 'Arun Jaitley Stadium',
 'Brabourne Stadium, Mumbai': 'Brabourne Stadium',
 'Brabourne Stadium': 'Brabourne Stadium',
 'Dr DY Patil Sports Academy, Mumbai': 'Dr DY Patil Sports Academy',
 'Dr DY Patil Sports Academy': 'Dr DY Patil Sports Academy',
 'Eden Gardens, Kolkata': 'Eden Gardens',
 'Eden Gardens': 'Eden Gardens',
 'M Chinnaswamy Stadium': 'M.Chinnaswamy Stadium',
```

```
 'M.Chinnaswamy Stadium': 'M.Chinnaswamy Stadium',
 'Maharashtra Cricket Association Stadium, Pune': 'Maharashtra Cricket Association
 'Maharashtra Cricket Association Stadium': 'Maharashtra Cricket Association Stadiu
 'Narendra Modi Stadium, Ahmedabad': 'Narendra Modi Stadium',
 'Narendra Modi Stadium': 'Narendra Modi Stadium',
 'Rajiv Gandhi International Stadium, Uppal': 'Rajiv Gandhi International Stadium',
 'Rajiv Gandhi International Stadium': 'Rajiv Gandhi International Stadium',
 'Wankhede Stadium, Mumbai': 'Wankhede Stadium',
 'Wankhede Stadium': 'Wankhede Stadium',
 'Himachal Pradesh Cricket Association Stadium': 'Himachal Pradesh Cricket Associat
 'Sawai Mansingh Stadium': 'Sawai Mansingh Stadium',
 'MA Chidambaram Stadium, Chepauk': 'MA Chidambaram Stadium',
 'MA Chidambaram Stadium, Chepauk, Chennai': 'MA Chidambaram Stadium',
 'MA Chidambaram Stadium': 'MA Chidambaram Stadium',
 'Punjab Cricket Association IS Bindra Stadium, Mohali': 'Punjab Cricket Associatio
 'Punjab Cricket Association Stadium, Mohali': 'Punjab Cricket Association IS Bindr
 'Punjab Cricket Association IS Bindra Stadium': 'Punjab Cricket Association IS Bin
}

venue_mapping.update({
    'Wankhede Stadium , Mumbai': 'Wankhede Stadium',
    'Rajiv Gandhi International Stadium, Hyderabad': 'Rajiv Gandhi International St
    'Sawai Mansingh Stadium, Jaipur': 'Sawai Mansingh Stadium',
    'Punjab Cricket Association IS Bindra Stadium,Chandigarh': 'Punjab Cricket Asso
    'M Chinnaswamy Stadium, Bangalore': 'M.Chinnaswamy Stadium',
    'M.Chinnaswamy Stadium, Bangalore': 'M.Chinnaswamy Stadium',
    'M Chinnaswamy Stadium, Bengaluru': 'M.Chinnaswamy Stadium',
    'MA Chidambaram Stadium, Chennai': 'MA Chidambaram Stadium',
    ' Bharat Ratna Shri Atal Bihari Vajpayee Ekana Cricket Stadium, Lucknow':
})
```

In [4]:
```
team_mapping = {
 'Rajasthan Royals': 'Rajasthan Royals',
 'Gujarat Titans': 'Gujarat Titans',
 'Royal Challengers Bangalore': 'Royal Challengers Bangalore',
 'Lucknow Super Giants': 'Lucknow Super Giants',
 'Sunrisers Hyderabad': 'Sunrisers Hyderabad',
 'Mumbai Indians': 'Mumbai Indians',
 'Chennai Super Kings': 'Chennai Super Kings',
 'Kolkata Knight Riders': 'Kolkata Knight Riders',
 'Kings XI Punjab': 'Punjab Kings',
 'Punjab Kings': 'Punjab Kings',
 'Delhi Daredevils': 'Delhi Capitals',
 'Delhi Capitals': 'Delhi Capitals'
}
```

In [5]:
```
def do_venue_mapping(df):
    df.venue = df.venue.map(venue_mapping)
    return df
```

In [6]:
```
def do_team_mapping(ball_by_ball, matches_result):
    matches_result.team_1 = matches_result.team_1.map(team_mapping)
    matches_result.team_2 = matches_result.team_2.map(team_mapping)
    ball_by_ball.batting_team = ball_by_ball.batting_team.map(team_mapping)
    return ball_by_ball, matches_result
```

In [7]:
```
def remove_unnecessary_rows(ball_by_ball, matches_result):
    ball_by_ball = ball_by_ball.dropna(subset=['batting_team'])
    matches_result = matches_result.dropna(subset=['team_1', 'team_2', 'venue'])
    return ball_by_ball, matches_result
```

In [8]:
```
def select_innings_and_overs(ball_by_ball):
    ball_by_ball = ball_by_ball.loc[(ball_by_ball.overs <= 5) & (ball_by_ball.innin
    ball_by_ball.innings = ball_by_ball.innings.replace({1: 0, 2: 1})
    return ball_by_ball
```

In [9]:
```
def get_final_training_dataframe(ball_by_ball, matches_result):
    gb = ball_by_ball.groupby(['match_id', 'innings', 'batting_team'])

    total_runs = gb['total_run'].sum()
    batsmen = gb['batter'].unique()
    bowlers = gb['bowler'].unique()

    total_runs = total_runs.to_frame(name = 'total_runs').reset_index()
    batsmen = batsmen.to_frame(name = 'batsmen').reset_index()
    bowlers = bowlers.to_frame(name = 'bowlers').reset_index()

    data = total_runs.merge(batsmen, how='right', on=['match_id','innings','batting
    data = data.merge(bowlers, how='right', on=['match_id','innings','batting_team'
    data = data.merge(matches_result, on=['match_id'])

    mask = data['batting_team'] == data['team_1']
    data.loc[mask, 'bowling_team'] = data['team_2']
    data.loc[~mask, 'bowling_team'] = data['team_1']

    # match_id == 829763, data for one innings is missing
    # match_id == 829813, total_runs for one innings is 2 (probably a mistake in da
    data = data.drop(data[(data['match_id'] == 829763) | (data['match_id'] == 82981

    data['count_batsmen'] = [len(x) for x in data['batsmen']]
    data['count_bowlers'] = [len(x) for x in data['bowlers']]

    data = data.drop(columns=['match_id', 'batsmen', 'bowlers', 'team_1', 'team_2']
    data = data[['venue', 'innings', 'batting_team', 'bowling_team', 'count_batsmen

    return data
```

In [10]:
```
def preprocess(training_data):
    ball_by_ball, matches_result = get_initial_training_dataframes(training_data)
    matches_result = do_venue_mapping(matches_result)
    ball_by_ball, matches_result = do_team_mapping(ball_by_ball, matches_result)
    ball_by_ball, matches_result = remove_unnecessary_rows(ball_by_ball, matches_re
    ball_by_ball = select_innings_and_overs(ball_by_ball)
    data = get_final_training_dataframe(ball_by_ball, matches_result)
    return data
```

In [11]:
```
class MyModel:
    def __init__(self):
        pass
```

In [12]:
```
def get_trained_model(X_train, y_train):
    from sklearn.linear_model import LinearRegression
    return LinearRegression().fit(X_train, y_train)
```

In [13]:
```
def MyModel_fit(self, training_data):
    data = preprocess(training_data)

    X = data.iloc[:, :-1]
    y = data["total_runs"]

    self.ct = ColumnTransformer(transformers = [
        ('ohe', OneHotEncoder(categories = "auto", drop='first', sparse_output=Fals
    ], remainder = 'passthrough')

    self.scaler = StandardScaler()

    X_ohe = pd.DataFrame(self.ct.fit_transform(X))
    X_std = self.scaler.fit_transform(X_ohe)

    X_train, X_test, y_train, y_test = train_test_split(X_std, y, test_size = 0.2)
    self.model = get_trained_model(X_train, y_train)

    self.debug = {
        "data": data,
        "X_ohe": X_ohe,
        "X_std": X_std,
        "X_train": X_train,
        "X_test": X_test,
        "y_train": y_train,
        "y_test": y_test
    }
```

In [14]:
```
def MyModel_predict(self, test_data, dev=False):
    if (dev == False):
        test_data = test_data.iloc[:, 1:]
        test_data['count_batsmen'] = [len(x) for x in test_data['batsmen']]
        test_data['count_bowlers'] = [len(x) for x in test_data['bowlers']]
        test_data = do_venue_mapping(test_data)

    test_data_ohe = self.ct.transform(test_data)
    test_data_std = self.scaler.transform(test_data_ohe)
    return self.model.predict(test_data_std)
```

In [15]:
```
MyModel.fit = MyModel_fit
MyModel.predict = MyModel_predict
```

Main.py

In [16]:
```
ball_by_ball = pd.read_csv('./Data/IPL_Ball_by_Ball_2008_2022.csv')
matches_result = pd.read_csv('./Data/IPL_Matches_Result_2008_2022.csv')
```

In [17]:
```
a_model = MyModel()
```

In [18]:
```
a_model.fit([ball_by_ball, matches_result])
```

```
C:\Users\k26ra\AppData\Local\Temp\ipykernel_18360\671351269.py:3: SettingWithCopyWar
ning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/u
ser_guide/indexing.html#returning-a-view-versus-a-copy
  ball_by_ball.innings = ball_by_ball.innings.replace({1: 0, 2: 1})
```

In [19]:
```
from sklearn.metrics import mean_absolute_error
sample = a_model.debug['data'].sample(frac=.5)
X = sample.iloc[:, 0:-1]
y = sample.iloc[:, -1]
y_pred = a_model.predict(X, dev=True)
mean_absolute_error(y, y_pred)
```

Out[19]:
```
8.280540344552167
```

FilesUsed

In [20]:
```
import os
```

In [21]:
```
files = os.listdir('./FilesUsed')
```

In [31]:
```
for file in files:
    if 'test_file_matchid' in file:
        X_file_name = './FilesUsed/' + file
        y_file_name = './FilesUsed/' + 'test_file_labels_matchid_' + file[-6:]

        print(f'X_file_name: {X_file_name}')

        X = pd.read_csv(X_file_name)
        y = pd.read_csv(y_file_name)['actual_runs']

        print(f"'{X['venue'][0]}'")
        print(f"{X['venue'][0] in venue_mapping}")

        y_pred = a_model.predict(X)
        print(*y)
        print(*y_pred)
        print(mean_absolute_error(y, y_pred), '\n')
```

```
X_file_name: ./FilesUsed/test_file_matchid_12.csv
'Wankhede Stadium , Mumbai'
True
61.0 68.0
155.7364196858824 202.8119920394721
114.77420586267725

X_file_name: ./FilesUsed/test_file_matchid_13.csv
'Narendra Modi Stadium, Ahmedabad'
True
54.0 43.0
165.50119569745095 36.92874466147033
58.78622551799031

X_file_name: ./FilesUsed/test_file_matchid_14.csv
'Rajiv Gandhi International Stadium, Hyderabad'
True
41.0 34.0
-28.67676118408346 180.74676569517428
108.21176343962887

X_file_name: ./FilesUsed/test_file_matchid_15.csv
'M.Chinnaswamy Stadium, Bangalore'
True
56.0 37.0
170.30055647582594 24.884633908191685
63.20796128381713

X_file_name: ./FilesUsed/test_file_matchid_16.csv
'Arun Jaitley Stadium, Delhi'
True
51.0 68.0
257.72018484872314 272.5678169811059
205.64400091491453

X_file_name: ./FilesUsed/test_file_matchid_17.csv
'MA Chidambaram Stadium, Chennai'
True
57.0 45.0
107.88245080625184 180.36576003389644
93.12410542007413

X_file_name: ./FilesUsed/test_file_matchid_18.csv
'Punjab Cricket Association IS Bindra Stadium,Chandigarh'
True
52.0 56.0
188.86539124311835 150.41775272059283
115.64157198185559

X_file_name: ./FilesUsed/test_file_matchid_19.csv
'Eden Gardens, Kolkata'
True
65.0 62.0
127.19836383473871 5.766806407263374
59.21577871373767
```

```
X_file_name: ./FilesUsed/test_file_matchid_20.csv
'M Chinnaswamy Stadium, Bengaluru'
True
47.0 32.0
217.49619873892232 75.29579700745931
106.89599787319082

X_file_name: ./FilesUsed/test_file_matchid_21.csv
' Bharat Ratna Shri Atal Bihari Vajpayee Ekana Cricket Stadium, Lucknow'
False
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[31], line 14
     11 print(f"'{X['venue'][0]}'")
     12 print(f"{X['venue'][0] in venue_mapping}")
---> 14 y_pred = a_model.predict(X)
     15 print(*y)
     16 print(*y_pred)

Cell In[14], line 8, in MyModel_predict(self, test_data, dev)
      5     test_data['count_bowlers'] = [len(x) for x in test_data['bowlers']]
      6     test_data = do_venue_mapping(test_data)
----> 8 test_data_ohe = self.ct.transform(test_data)
      9 test_data_std = self.scaler.transform(test_data_ohe)
     10 return self.model.predict(test_data_std)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_set_
output.py:140, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    145             *data_to_wrap[1:],
    146         )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\compose\_co
lumn_transformer.py:800, in ColumnTransformer.transform(self, X)
    795 else:
    796     # ndarray was used for fitting or transforming, thus we only
    797     # check that n_features_in_ is consistent
    798     self._check_n_features(X, reset=False)
--> 800 Xs = self._fit_transform(
    801     X,
    802     None,
    803     _transform_one,
    804     fitted=True,
    805     column_as_strings=fit_dataframe_and_transform_dataframe,
    806 )
    807 self._validate_output(Xs)
    809 if not Xs:
    810     # All transformers are None

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\compose\_co
lumn_transformer.py:658, in ColumnTransformer._fit_transform(self, X, y, func, fitte
d, column_as_strings)
    652 transformers = list(
    653     self._iter(
    654         fitted=fitted, replace_strings=True, column_as_strings=column_as_str
ings
    655     )
    656 )
    657 try:
--> 658     return Parallel(n_jobs=self.n_jobs)(
```

```
    659         delayed(func)(
    660             transformer=clone(trans) if not fitted else trans,
    661             X=_safe_indexing(X, column, axis=1),
    662             y=y,
    663             weight=weight,
    664             message_clsname="ColumnTransformer",
    665             message=self._log_message(name, idx, len(transformers)),
    666         )
    667         for idx, (name, trans, column, weight) in enumerate(transformers, 1)
    668     )
    669 except ValueError as e:
    670     if "Expected 2D array, got 1D array instead" in str(e):

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\parallel.py:63, in Parallel.__call__(self, iterable)
     58 config = get_config()
     59 iterable_with_config = (
     60     (_with_config(delayed_func, config), args, kwargs)
     61     for delayed_func, args, kwargs in iterable
     62 )
---> 63 return super().__call__(iterable_with_config)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:1048, in Parallel.__call__(self, iterable)
   1039 try:
   1040     # Only set self._iterating to True if at least a batch
   1041     # was dispatched. In particular this covers the edge
   (...)
   1045     # was very quick and its callback already dispatched all the
   1046     # remaining jobs.
   1047     self._iterating = False
-> 1048     if self.dispatch_one_batch(iterator):
   1049         self._iterating = self._original_iterator is not None
   1051     while self.dispatch_one_batch(iterator):

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:864, in Parallel.dispatch_one_batch(self, iterator)
    862     return False
    863 else:
--> 864     self._dispatch(tasks)
    865     return True

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:782, in Parallel._dispatch(self, batch)
    780 with self._lock:
    781     job_idx = len(self._jobs)
--> 782     job = self._backend.apply_async(batch, callback=cb)
    783     # A job can complete so quickly than its callback is
    784     # called before we get here, causing self._jobs to
    785     # grow. To ensure correct results ordering, .insert is
    786     # used (rather than .append) in the following line
    787     self._jobs.insert(job_idx, job)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\_parallel_backends.py:208, in SequentialBackend.apply_async(self, func, callback)
    206 def apply_async(self, func, callback=None):
```

```
    207     """Schedule a func to be run"""
--> 208     result = ImmediateResult(func)
    209     if callback:
    210         callback(result)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\_parallel_backends.py:572, in ImmediateResult.__init__(self, batch)
    569 def __init__(self, batch):
    570     # Don't delay the application, to avoid keeping the input
    571     # arguments in memory
--> 572     self.results = batch()

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:263, in BatchedCalls.__call__(self)
    259 def __call__(self):
    260     # Set the default nested backend to self._backend but do not set the
    261     # change the default number of processes to -1
    262     with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 263         return [func(*args, **kwargs)
    264                 for func, args, kwargs in self.items]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\joblib\parallel.py:263, in <listcomp>(.0)
    259 def __call__(self):
    260     # Set the default nested backend to self._backend but do not set the
    261     # change the default number of processes to -1
    262     with parallel_backend(self._backend, n_jobs=self._n_jobs):
--> 263         return [func(*args, **kwargs)
    264                 for func, args, kwargs in self.items]

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\parallel.py:123, in _FuncWrapper.__call__(self, *args, **kwargs)
    121     config = {}
    122 with config_context(**config):
--> 123     return self.function(*args, **kwargs)

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\pipeline.py:876, in _transform_one(transformer, X, y, weight, **fit_params)
    875 def _transform_one(transformer, X, y, weight, **fit_params):
--> 876     res = transformer.transform(X)
    877     # if we have a weight for this transformer, multiply output
    878     if weight is None:

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\utils\_set_output.py:140, in _wrap_method_output.<locals>.wrapped(self, X, *args, **kwargs)
    138 @wraps(f)
    139 def wrapped(self, X, *args, **kwargs):
--> 140     data_to_wrap = f(self, X, *args, **kwargs)
    141     if isinstance(data_to_wrap, tuple):
    142         # only wrap the first output for cross decomposition
    143         return (
    144             _wrap_data_with_container(method, data_to_wrap[0], X, self),
    145             *data_to_wrap[1:],
    146         )

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessi
```

```
ng\_encoders.py:917, in OneHotEncoder.transform(self, X)
    912 # validation of X happens in _check_X called by _transform
    913 warn_on_unknown = self.drop is not None and self.handle_unknown in {
    914     "ignore",
    915     "infrequent_if_exist",
    916 }
--> 917 X_int, X_mask = self._transform(
    918     X,
    919     handle_unknown=self.handle_unknown,
    920     force_all_finite="allow-nan",
    921     warn_on_unknown=warn_on_unknown,
    922 )
    923 self._map_infrequent_categories(X_int, X_mask)
    925 n_samples, n_features = X_int.shape

File ~\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\preprocessi
ng\_encoders.py:174, in _BaseEncoder._transform(self, X, handle_unknown, force_all_f
inite, warn_on_unknown)
    169 if handle_unknown == "error":
    170     msg = (
    171         "Found unknown categories {0} in column {1}"
    172         " during transform".format(diff, i)
    173     )
--> 174     raise ValueError(msg)
    175 else:
    176     if warn_on_unknown:

ValueError: Found unknown categories [nan] in column 0 during transform
```