Importing libraries

In [2]:
```
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
```

In [3]:
```
bowlers_stats_auction_data =pd.DataFrame()
batsmen_stats_auction_data =pd.DataFrame()
```

In [4]:
```
batsmen_filepath = "./data/IPL Player Stats/Batting Stats/all_batsmen_data_2016
bowlers_filepath = "./data/IPL Player Stats/Bowling Stats/all_bowlers_data_2016
```

In [5]:
```
#reading the data
all_batsmen = pd.read_csv(batsmen_filepath)
all_bowlers = pd.read_csv(bowlers_filepath)
print(all_bowlers.shape)
print(all_batsmen.shape)
(all_batsmen.head())
```

```
(615, 14)
(338, 10)
```

Out[5]:

	Unnamed: 0	Player	Inns	Runs	Avg	SR	50	100	4s	6s
0	0	AB de Villiers	77	2592	42.345000	156.153333	25	1	195	148
1	1	Aaron Finch	51	1180	24.098000	136.744000	9	0	114	49
2	2	Abdul Samad	20	226	12.176667	118.493333	0	0	12	14
3	3	Abhijeet Tomar	1	4	4.000000	50.000000	0	0	1	0
4	4	Abhinav Manohar	7	108	18.000000	144.000000	0	0	14	3

Checking for duplicates

In [6]:
```
all_bowlers[all_bowlers.duplicated()]
```

Out[6]:

Unnamed: 0	POS	Player	Mat	Inns	Ov	Runs	Wkts	BBI	Avg	Econ	SR	4w	5w

In [7]:
```
(all_batsmen[all_batsmen.duplicated()])
```

Out[7]:

Unnamed: 0	Player	Inns	Runs	Avg	SR	50	100	4s	6s

In [8]:
```
all_batsmen = all_batsmen.drop(columns="Unnamed: 0")
```

In [9]:
```
all_batsmen["Player"] = all_batsmen["Player"].str.lower()
all_batsmen["Player"] = all_batsmen["Player"].apply(lambda x : x.strip())
print(all_batsmen.shape)
```

```
(338, 9)
```

Reading auction data

In [10]:
```
auction_data = pd.read_csv("./data/all_auction_data*.csv")
auction_data.shape
```

Out[10]: (586, 5)

Joining auction data with batsmen stats data

In [11]:
```
batsmen_stats_auction_data = pd.merge(all_batsmen, auction_data,  how='inner',
                                      right_on = ['player_name'])
batsmen_stats_auction_data.head()
```

Out[11]:

	Player	Inns	Runs	Avg	SR	50	100	4s	6s	Unnamed: 0	player_name	
0	aaron finch	51	1180	24.098000	136.744000	9	0	114	49	283	aaron finch	Ba
1	abdul samad	20	226	12.176667	118.493333	0	0	12	14	291	abdul samad	Ro
2	abhijeet tomar	1	4	4.000000	50.000000	0	0	1	0	85	abhijeet tomar	Ba
3	abhishek sharma	34	667	25.692000	136.292000	2	0	64	25	204	abhishek sharma	Ro
4	adam milne	6	23	5.750000	76.220000	0	0	0	1	18	adam milne	

In [12]:
```
all_bowlers["Player"] = all_bowlers["Player"].str.lower()
all_bowlers["Player"] = all_bowlers["Player"].apply(lambda x : x.strip())
```

In [13]:
```
print(auction_data.shape)
auction_data[auction_data["type"]=="Bowler"].head()
```

(586, 5)

Out[13]:

	Unnamed: 0	player_name	type	sold_price	year
4	4	deepak chahar	Bowler	4.966667e+07	2022
8	8	k.m. asif	Bowler	2.000000e+06	2022
9	9	tushar deshpande	Bowler	2.000000e+06	2022
12	12	maheesh theekshana	Bowler	7.000000e+06	2022
14	14	simarjeet singh	Bowler	2.000000e+06	2022

Joining auction data with bowlers stats data

In [14]:
```
bowlers_stats_auction_data = pd.merge(all_bowlers, auction_data[auction_data["t
                                      left_on=['Player'], right_on = ['player_r
```

In [15]:
```
batsmen_stats_auction_data.head()
```

Out[15]:

	Player	Inns	Runs	Avg	SR	50	100	4s	6s	Unnamed: 0	player_name	
0	aaron finch	51	1180	24.098000	136.744000	9	0	114	49	283	aaron finch	Ba
1	abdul samad	20	226	12.176667	118.493333	0	0	12	14	291	abdul samad	Ro
2	abhijeet tomar	1	4	4.000000	50.000000	0	0	1	0	85	abhijeet tomar	Ba
3	abhishek sharma	34	667	25.692000	136.292000	2	0	64	25	204	abhishek sharma	Ro
4	adam milne	6	23	5.750000	76.220000	0	0	0	1	18	adam milne	

In [16]:

```
bowlers_stats_auction_data.head()
```

Out[16]:

	Unnamed: 0_x	POS	Player	Mat	Inns	Ov	Runs	Wkts	BBI	Avg	Econ	SR	4w
0	0	1	bhuvneshwar kumar	17	17	66.0	490	23	5/19	21.30	7.42	17.21	1
1	86	1	bhuvneshwar kumar	14	14	52.0	369	26	5/19	14.19	7.05	12.07	0
2	209	34	bhuvneshwar kumar	12	12	46.0	354	9	5/19	39.33	7.66	30.77	0
3	275	18	bhuvneshwar kumar	15	15	59.0	461	13	5/19	35.46	7.81	27.23	0
4	398	54	bhuvneshwar kumar	4	4	14.0	99	3	5/19	33.00	6.98	28.33	0

In [17]:

```
bowlers_stats_auction_data.columns
```

Out[17]:

```
Index(['Unnamed: 0_x', 'POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts',
       'BBI', 'Avg', 'Econ', 'SR', '4w', '5w', 'Unnamed: 0_y', 'player_name',
       'type', 'sold_price', 'year'],
      dtype='object')
```

In [18]:

```
batsmen_stats_auction_data["sold_price"] = batsmen_stats_auction_data.sold_pric
batsmen_stats_auction_data.head()
```

Out[18]:

	Player	Inns	Runs	Avg	SR	50	100	4s	6s	Unnamed: 0	player_name	
0	aaron finch	51	1180	24.098000	136.744000	9	0	114	49	283	aaron finch	Ba
1	abdul samad	20	226	12.176667	118.493333	0	0	12	14	291	abdul samad	Ro
2	abhijeet tomar	1	4	4.000000	50.000000	0	0	1	0	85	abhijeet tomar	Ba
3	abhishek sharma	34	667	25.692000	136.292000	2	0	64	25	204	abhishek sharma	Ro
4	adam milne	6	23	5.750000	76.220000	0	0	0	1	18	adam milne	

In [19]:

```
batsmen_stats_auction_data = batsmen_stats_auction_data.drop(columns = ["SR", '
batsmen_stats_auction_data.head()
```

Out[19]:

	Inns	Runs	sold_price
0	51	1180	37.600000
1	20	226	2.000000
2	1	4	4.000000
3	34	667	35.250000
4	6	23	19.333333

In [20]:

```
bowlers_stats_auction_data.head()
```

Out[20]:

	Unnamed: 0_x	POS	Player	Mat	Inns	Ov	Runs	Wkts	BBI	Avg	Econ	SR	4w
0	0	1	bhuvneshwar kumar	17	17	66.0	490	23	5/19	21.30	7.42	17.21	1
1	86	1	bhuvneshwar kumar	14	14	52.0	369	26	5/19	14.19	7.05	12.07	0
2	209	34	bhuvneshwar kumar	12	12	46.0	354	9	5/19	39.33	7.66	30.77	0
3	275	18	bhuvneshwar kumar	15	15	59.0	461	13	5/19	35.46	7.81	27.23	0
4	398	54	bhuvneshwar kumar	4	4	14.0	99	3	5/19	33.00	6.98	28.33	0

train and test data split

In [21]:

```
from sklearn.model_selection import train_test_split
```

In [22]:

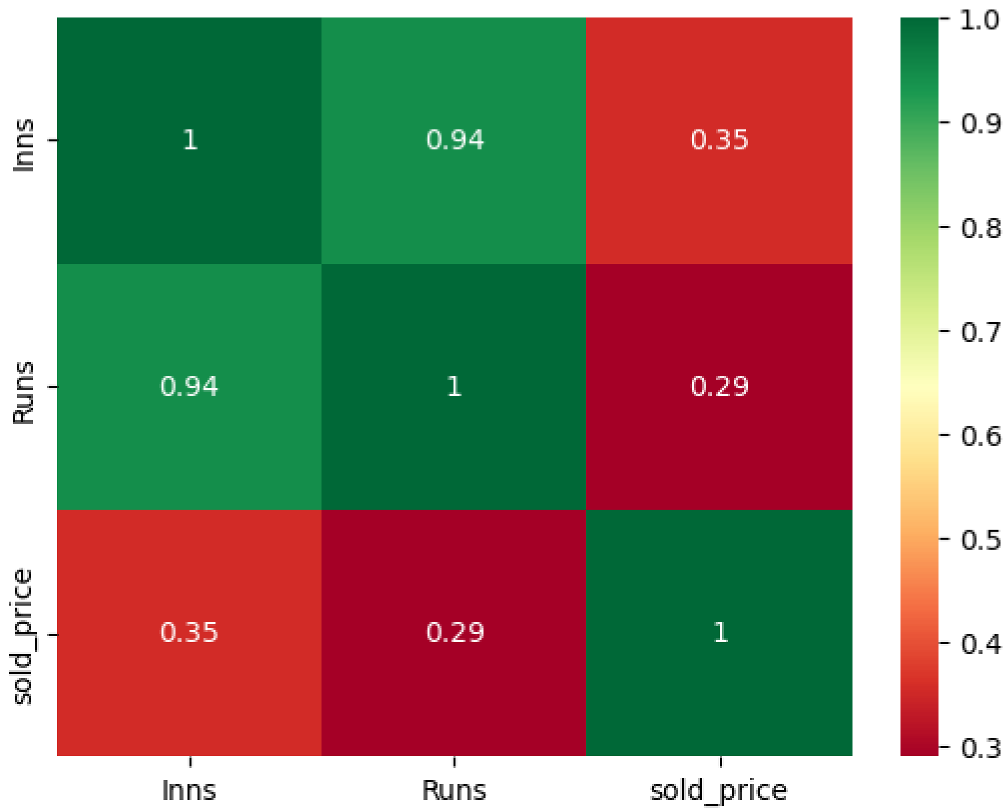
```
X_train, X_test, y_train, y_test = train_test_split(batsmen_stats_auction_data.
```

In [23]:
```
(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```
Out[23]:
```
((272, 2), (31, 2), (272,), (31,))
```

In [24]:
```
corr = batsmen_stats_auction_data.corr()
corr_features = corr.index
```

In [25]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

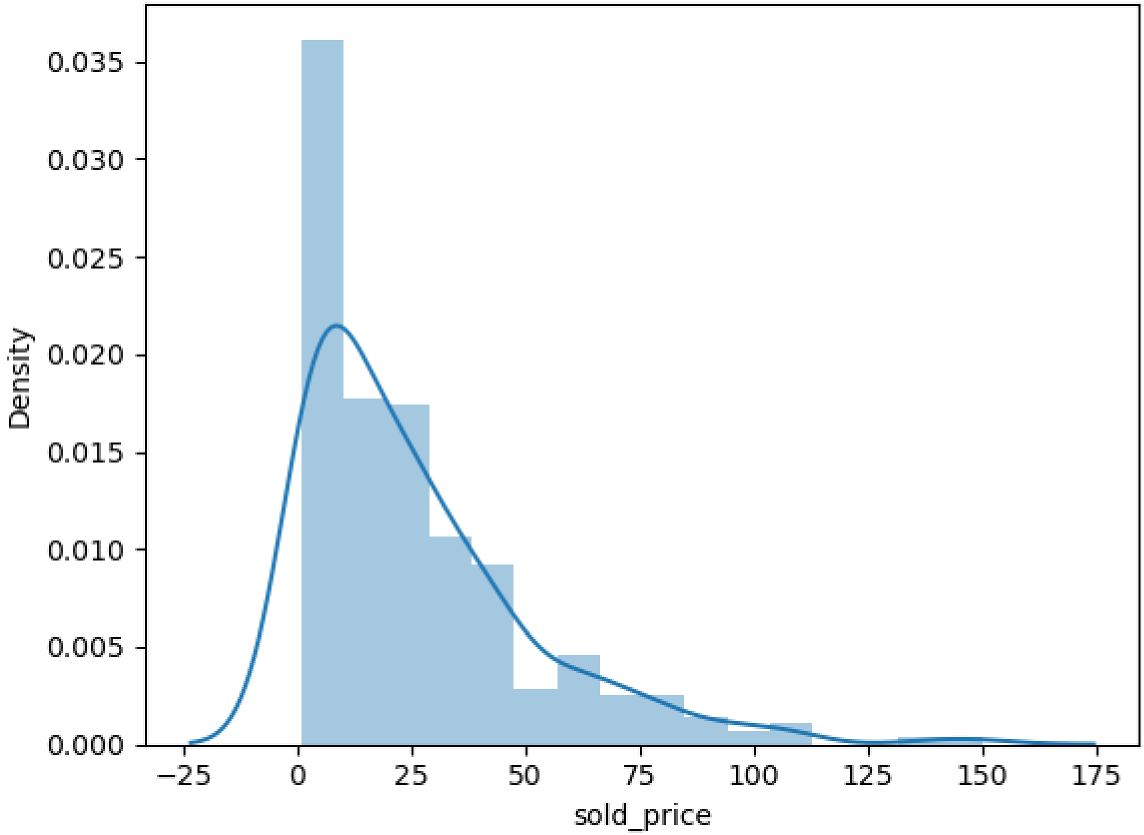
In [26]:
```
g = sns.heatmap(data=batsmen_stats_auction_data[corr_features].corr(),
                annot=True, cmap='RdYlGn')
```



In [27]:
```
batsmen_stats_auction_data.head()
```
Out[27]:

	Inns	Runs	sold_price
0	51	1180	37.600000
1	20	226	2.000000
2	1	4	4.000000
3	34	667	35.250000
4	6	23	19.333333

In [28]:
```
sns.distplot(batsmen_stats_auction_data["sold_price"])
```
Out[28]:
```
<Axes: xlabel='sold_price', ylabel='Density'>
```



In [29]:
```
batsmen_stats_auction_data.dtypes
```

Out[29]:
```
Inns              int64
Runs              int64
sold_price      float64
dtype: object
```

Modelling batsmen data set

Linear Regression

In [30]:
```
from sklearn.linear_model import LinearRegression
```

In [31]:
```
regressor = LinearRegression()

regressor.fit(X_train, y_train)
```

Out[31]:
```
▾ LinearRegression
LinearRegression()
```

In [32]:
```
#regressor.score(X_test, y_test)
y_pred_lr = regressor.predict(X_test)
```

In [33]:
```
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as m
import numpy as np
print("---- Linear Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_lr)))
```

```
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_lr)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_lr
```

```
---- Linear Regression - Model Evaluation ----
Mean Absolute Error (MAE): 21.904289786918266
Mean Squared Error (MSE): 906.2024861486932
Root Mean Squared Error (RMSE): 30.10319727452041
```

Decision Tree Regressor Model on batsmen data

In [34]:
```
from sklearn.tree import DecisionTreeRegressor
decision_regressor = DecisionTreeRegressor()
decision_regressor.fit(X_train,y_train)
```

Out[34]:
```
▾ DecisionTreeRegressor
DecisionTreeRegressor()
```

In [35]:
```
y_pred_dr = decision_regressor.predict(X_test)
```

In [36]:
```
print("---- Decision Tree Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_dr)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_dr)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_dr
```

```
---- Decision Tree Regression - Model Evaluation ----
Mean Absolute Error (MAE): 23.63763440860215
Mean Squared Error (MSE): 1241.0720967741936
Root Mean Squared Error (RMSE): 35.228853185623194
```

RandomForest Regression - batsmen data

In [37]:
```
from sklearn.ensemble import RandomForestRegressor
random_regressor = RandomForestRegressor()
random_regressor.fit(X_train,y_train)
```

Out[37]:
```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [38]:
```
y_pred_rfr = random_regressor.predict(X_test)
```

In [39]:
```
print("---- Random Forest Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_rfr)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_rfr)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_rf
```

```
---- Random Forest Regression - Model Evaluation ----
Mean Absolute Error (MAE): 20.483322202439144
Mean Squared Error (MSE): 818.7600973567529
Root Mean Squared Error (RMSE): 28.613984297136128
```

Ada Boost Regressor model - batsmen data

In [40]:
```
from sklearn.ensemble import AdaBoostRegressor
adb_regressor = AdaBoostRegressor(base_estimator=regressor, n_estimators=100)
adb_regressor.fit(X_train, y_train)
```

Out[40]:
```
▸ AdaBoostRegressor
▸ base_estimator: LinearRegression
    ▸ LinearRegression
```

In [41]:
```
y_pred_adb = adb_regressor.predict(X_test)
```

In [42]:
```
print("---- Ada Boost Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test, y_pred_adb)))
print("Mean Squared Error (MSE): {}".format(mse(y_test, y_pred_adb)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test, y_pred_ac
```

```
---- Ada Boost Regression - Model Evaluation ----
Mean Absolute Error (MAE): 21.821809632035976
Mean Squared Error (MSE): 799.3936886600872
Root Mean Squared Error (RMSE): 28.27355104439637
```

Linear Regression has performed well then any other regressions

Below is the test conducted on sample inputs we can able to measure maximum worth of the batsmen with the help of player stats

In [43]:
```
def predict_batsman_price( Inns, Runs):
    t = [ Inns, Runs]
    test = np.array([t])
    #print(test)
    return regressor.predict(test)[0]
```

In [44]:
```
print("sample input 1 (KL Rahul's Data): predicted value = {} million Rupees".\
            format(predict_batsman_price( 228, 4163,)))
print("sample input 2 (Rishabh Pant's Data): predicted value = {} million Rupee
            format(predict_batsman_price( 134, 2838)))
print("sample input 3 (Dummy Data): predicted value = {} million Rupees".\
            format(predict_batsman_price( 1, 10)))
```

```
sample input 1 (KL Rahul's Data): predicted value = 131.32569949630064 million
Rupees
sample input 2 (Rishabh Pant's Data): predicted value = 77.36403654679917 mill
ion Rupees
sample input 3 (Dummy Data): predicted value = 15.616234283757871 million Rupe
es
```

Bowler Value prediction

In [45]:
```
bowlers_stats_auction_data.columns
```

Out[45]:
```
Index(['Unnamed: 0_x', 'POS', 'Player', 'Mat', 'Inns', 'Ov', 'Runs', 'Wkts',
       'BBI', 'Avg', 'Econ', 'SR', '4w', '5w', 'Unnamed: 0_y', 'player_name',
       'type', 'sold_price', 'year'],
      dtype='object')
```

Data Cleaning

In [46]:
```
bowlers_stats_auction_data.drop(columns=["POS","BBI","Mat","Avg", "Econ", "SR",
```

In [47]:
```
bowlers_stats_auction_data["sold_price"] = bowlers_stats_auction_data.sold_pric
bowlers_stats_auction_data.head()
```

Out[47]:

	Inns	Ov	Runs	Wkts	sold_price
0	17	66.0	490	23	42.25
1	14	52.0	369	26	42.25
2	12	46.0	354	9	42.25
3	15	59.0	461	13	42.25
4	4	14.0	99	3	42.25

In [48]:
```
bowlers_stats_auction_data.head()
```

Out[48]:

	Inns	Ov	Runs	Wkts	sold_price
0	17	66.0	490	23	42.25
1	14	52.0	369	26	42.25
2	12	46.0	354	9	42.25
3	15	59.0	461	13	42.25
4	4	14.0	99	3	42.25

Identifying correlations between attributes

In [49]:
```
corr = bowlers_stats_auction_data.corr()
corr_features = corr.index

corr_features
```

Out[49]:
```
Index(['Inns', 'Ov', 'Runs', 'Wkts', 'sold_price'], dtype='object')
```

In [50]:
```
import seaborn as sns
g = sns.heatmap(data=bowlers_stats_auction_data[corr_features].corr(), annot=Tr
```

In [51]:
```
bowlers_stats_auction_data.head()
```

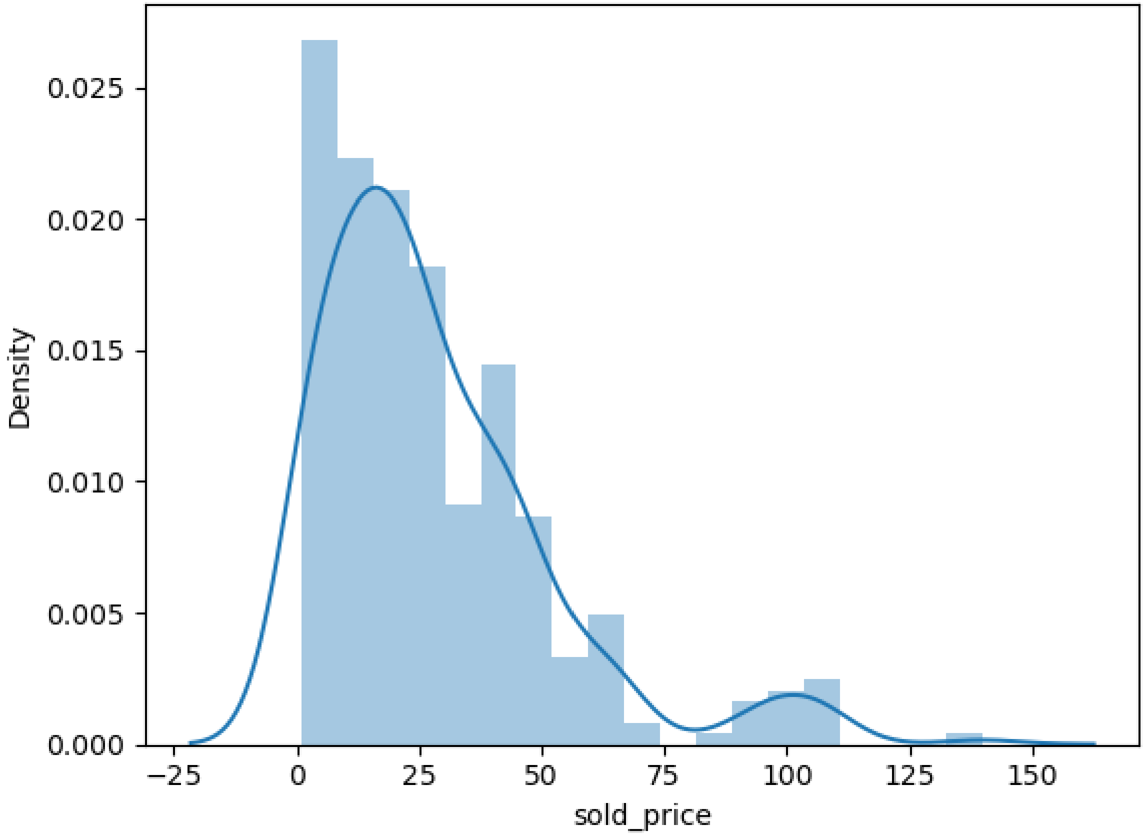
Out[51]:

	Inns	Ov	Runs	Wkts	sold_price
0	17	66.0	490	23	42.25
1	14	52.0	369	26	42.25
2	12	46.0	354	9	42.25
3	15	59.0	461	13	42.25
4	4	14.0	99	3	42.25

distribution of sold_price

In [52]:
```
import seaborn as sns
sns.distplot(bowlers_stats_auction_data["sold_price"])
```

Out[52]: <Axes: xlabel='sold_price', ylabel='Density'>



In [53]:
```
X_train_bowlers, X_test_bowlers, y_train_bowlers, y_test_bowlers = train_test_s

X_train_bowlers.shape, y_train_bowlers.shape, X_test_bowlers.shape, y_test_bowl
```

Out[53]:
```
((297, 4), (297, 1), (34, 4), (34, 1))
```

In [54]:
```
from sklearn.linear_model import LinearRegression
```

In [55]:
```
linear_regressor = LinearRegression()
linear_regressor.fit(X_train_bowlers, y_train_bowlers)
```

Out[55]:
```
▾ LinearRegression
LinearRegression()
```

In [56]:
```
y_pred_lrb = linear_regressor.predict(X_test_bowlers)
```

In [57]:
```
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as m
import numpy as np
print("---- linear Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test_bowlers, y_pred_lrb)))
print("Mean Squared Error (MSE): {}".format(mse(y_test_bowlers, y_pred_lrb)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test_bowlers, y
```

```
---- linear Regression - Model Evaluation ----
Mean Absolute Error (MAE): 18.806985460509313
Mean Squared Error (MSE): 561.4687336458443
Root Mean Squared Error (RMSE): 23.695331473643584
```

Decision Tree Regression on bowlers data

In [58]:
```
from sklearn.tree import DecisionTreeRegressor

b_dt_regressor = DecisionTreeRegressor()
b_dt_regressor.fit(X_train_bowlers, y_train_bowlers)

y_pred_dtr = b_dt_regressor.predict(X_test_bowlers)
```

In [59]:
```
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as m
import numpy as np
print("---- Decision Tree Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test_bowlers, y_pred_dtr)))
print("Mean Squared Error (MSE): {}".format(mse(y_test_bowlers, y_pred_dtr)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test_bowlers, y
```

```
---- Decision Tree Regression - Model Evaluation ----
Mean Absolute Error (MAE): 25.050595238095237
Mean Squared Error (MSE): 1199.1091369464452
Root Mean Squared Error (RMSE): 34.62815526340445
```

Random Forest Regression - bowler data

In [60]:
```
from sklearn.ensemble import RandomForestRegressor
b_random_regressor = RandomForestRegressor()
b_random_regressor.fit(X_train_bowlers,y_train_bowlers)

y_pred_rfrb = b_random_regressor.predict(X_test_bowlers)
```

In [61]:
```
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as m
import numpy as np
print("---- Random Forest Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test_bowlers, y_pred_rfrb)))
print("Mean Squared Error (MSE): {}".format(mse(y_test_bowlers, y_pred_rfrb)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test_bowlers, y
```

```
---- Random Forest Regression - Model Evaluation ----
Mean Absolute Error (MAE): 22.69773914565826
Mean Squared Error (MSE): 843.8259864157225
Root Mean Squared Error (RMSE): 29.048683040986944
```

Ada Boost Regressor - bowler data

In [62]:
```
from sklearn.ensemble import AdaBoostRegressor
adb_regressor_b = AdaBoostRegressor(base_estimator=linear_regressor, n_estimato
adb_regressor_b.fit(X_train_bowlers, y_train_bowlers)

y_pred_adarb = adb_regressor_b.predict(X_test_bowlers)
```

In [63]:
```
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as m
import numpy as np
print("---- ADA Regression - Model Evaluation ----")
print("Mean Absolute Error (MAE): {}".format(mae(y_test_bowlers, y_pred_adarb))
print("Mean Squared Error (MSE): {}".format(mse(y_test_bowlers, y_pred_adarb)))
print("Root Mean Squared Error (RMSE): {}".format(np.sqrt(mse(y_test_bowlers, y
```

```
---- ADA Regression - Model Evaluation ----
Mean Absolute Error (MAE): 19.215608868350053
Mean Squared Error (MSE): 521.583405724245
Root Mean Squared Error (RMSE): 22.838200579823383
```

In [64]:
```
X_test_bowlers.columns
```

Out[64]:
```
Index(['Inns', 'Ov', 'Runs', 'Wkts'], dtype='object')
```

Random Forest Regression model performed well on bowlers data

Predictions using sample input

In [65]:
```
def predict_batsman_price( Inns, Overs, wkts, Runs ):
    t = [ Inns, Overs, wkts, Runs ]
    test = np.array([t])
    #print(test)
    return b_random_regressor.predict(test)[0]
```

In [66]:
```
print("sample input 1: predicted value = {} million Rupees".format(predict_bats

print("sample input 2: predicted value = {} million Rupees".format(predict_bats

print("sample input 3: predicted value = {} million Rupees".format(predict_bats
```

```
sample input 1: predicted value = 29.03988095238096 million Rupees
sample input 2: predicted value = 27.241607142857156 million Rupees
sample input 3: predicted value = 17.99195238095238 million Rupees
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: