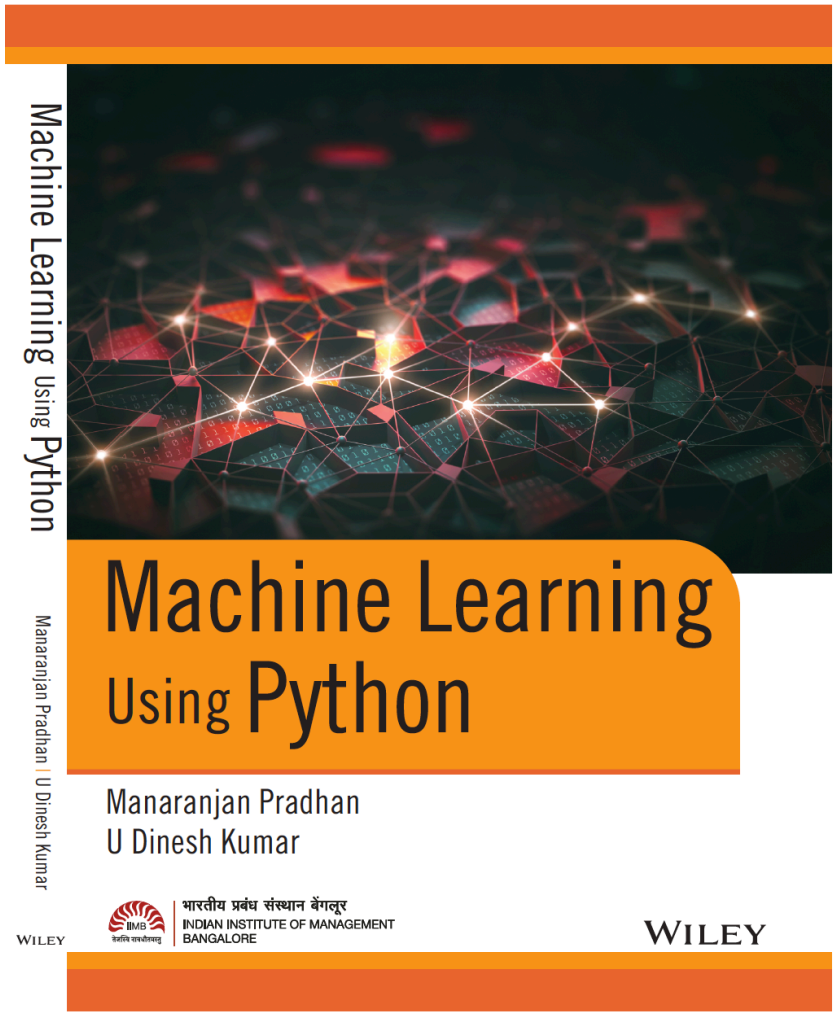


Chapter 02: Descriptive Analytics

Prepared By: Purvi Tiwari

Learning Objectives

- Working with DataFrames and perform basic exploratory Analysis
- Data Preparation Activities: Filtering, grouping, ordering, joining etc.
- Dealing with Missing Values
- Prepare plots such as bar plot, histogram, distribution plot, box plot, scatter plot, pair plot and heat maps to find insights.

Working with DataFrame

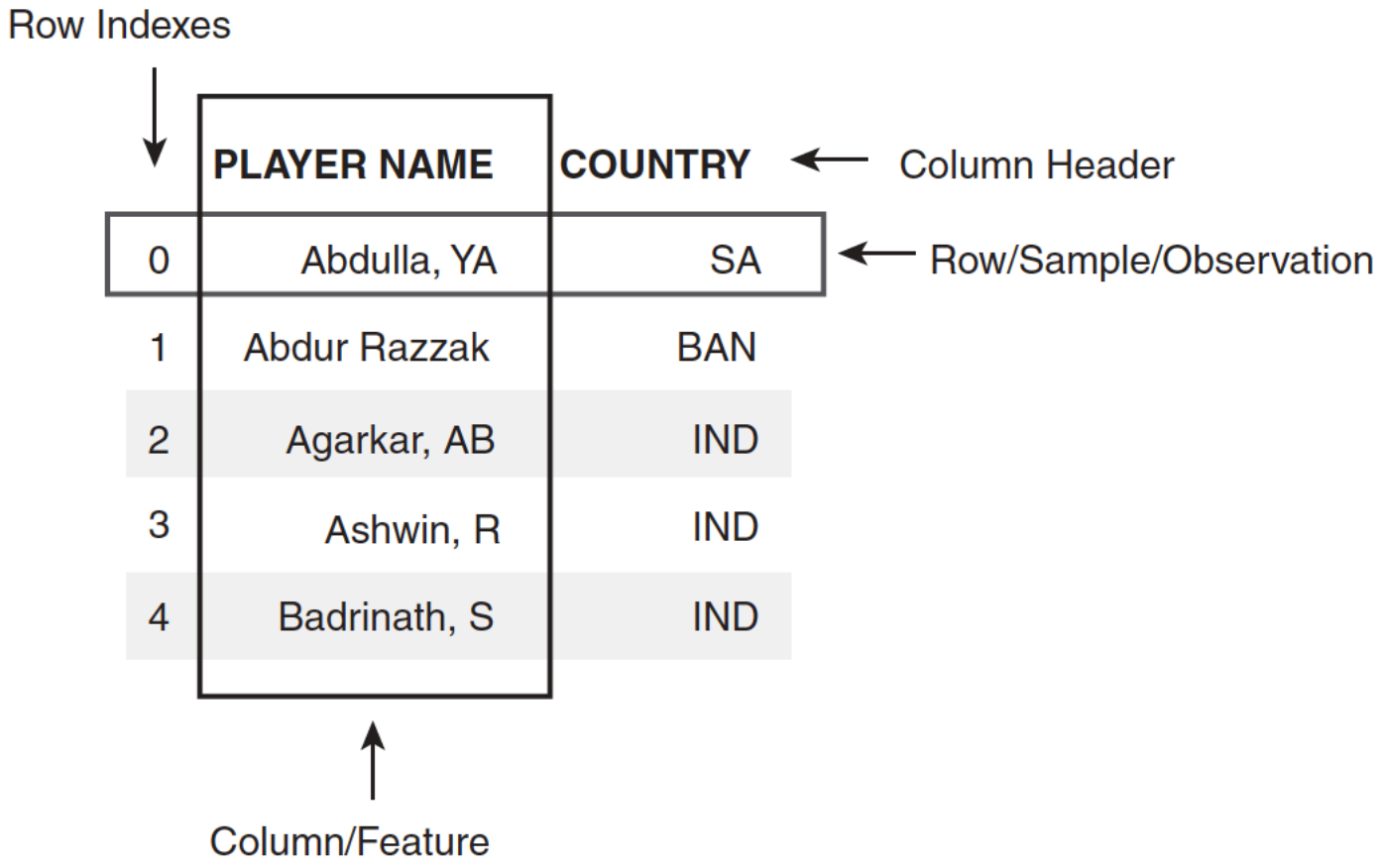


FIGURE 2.1 Structure of a DataFrame.

IPL Dataset

TABLE 2.1 IPL auction price data description

Data Code	Data Type	Description
AGE	Categorical	Age of the player at the time of auction classified into 3 categories. Category 1 (L25) means the player is less than 25 years old, 2 means that age is between 25 and 35 years (B25–35) and category 3 means that the age is more than 35 (A35).
RUNS-S	Continuous	Number of runs scored by a player
RUNS-C	Continuous	Number of runs conceded by a player
HS	Continuous	Highest score by the batsman in IPL
AVE-B	Continuous	Average runs scored by the batsman in IPL
AVE-BL	Continuous	Bowling average (Number of runs conceded / number of wickets taken) in IPL
SR-B	Continuous	Batting strike rate (ratio of the number of runs scored to the number of balls faced) in IPL
SR-BL	Continuous	Bowling strike rate (ratio of the number of balls bowled to the number of wickets taken) in IPL
SIXERS	Continuous	Number of six runs scored by a player in IPL
WKTS	Continuous	Number of wickets taken by a player in IPL
ECON	Continuous	Economy rate of a bowler (number of runs conceded by the bowler per over) in IPL
CAPTAINCY EXP	Categorical	Captained either a T20 team or a national team
ODI-SR-B	Continuous	Batting strike rate in One-Day Internationals
ODI-SR-BL	Continuous	Bowling strike rate in One-Day Internationals
ODI-RUNS-S	Continuous	Runs scored in One-Day Internationals
ODI-WKTS	Continuous	Wickets taken in One-Day Internationals
T-RUNS-S	Continuous	Runs scored in Test maches
T-WKTS	Continuous	Wickets taken in Test maches
PLAYER-SKILL	Categorical	Player's primary skill (batsman, bowler, or all-rounder)
COUNTRY	Categorical	Country of origin of the player (AUS: Australia; IND: India; PAK: Pakistan; SA: South Africa; SL: Sri Lanka; NZ: New Zealand; WI: West Indies; OTH: Other countries)
YEAR-A	Categorical	Year of Auction in IPL
IPL TEAM	Categorical	Team(s) for which the player had played in the IPL (CSK: Chennai Super Kings, DC: Deccan Chargers, DD: Delhi Daredevils, KXI: Kings XI Punjab, KKR: Kolkata Knight Riders; MI: Mumbai Indians; PWI: Pune Warriors India; RR: Rajasthan Royals; RCB: Royal Challengers Bangalore). A + sign was used to indicate that the player had played for more than one team. For example, CSK+ would mean that the player had played tor CSK as well as for one or more other teams.

Loading Dataset into DataFrame

```
import pandas as pd
```

```
ipl_auction_df = pd.read_csv('IPL IMB381IPL2013.csv')
```

```
ipl_auction_df.head(5)
```

	Sl. NO.	Player Name	Age	...	Auction Year	Base Price	Sold Price
0	1	Abdulla, YA	2	...	2009	50000	50000
1	2	Abdur Razzak	2	...	2008	50000	50000
2	3	Agarkar, AB	2	...	2008	200000	350000
3	4	Ashwin, R	1	...	2011	100000	850000
4	5	Badrinath, S	2	...	2011	100000	800000

Finding Summary of the DataFrame

```
ipl_auction_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 130 entries, 0 to 129
Data columns (total 26 columns):
Sl.NO.           130   non-null   int64
PLAYER NAME      130   non-null   object
AGE              130   non-null   int64
COUNTRY          130   non-null   Object
TEAM             130   non-null   Object
PLAYING ROLE     130   non-null   Object
T-RUNS           130   non-null   int64
T-WKTS           130   non-null   int64
ODI-RUNS-S       130   non-null   int64
ODI-SR-B         130   non-null   float64
ODI-WKTS         130   non-null   int64
ODI-SR-BL        130   non-null   float64
CAPTAINCY EXP    130   non-null   int64
RUNS-S           130   non-null   int64
HS               130   non-null   int64
AVE              130   non-null   float64
SR-B             130   non-null   float64
SIXERS           130   non-null   int64
RUNS-C           130   non-null   int64
WKTS             130   non-null   int64
AVE-BL           130   non-null   float64
ECON             130   non-null   float64
SR-BL            130   non-null   float64
AUCTION YEAR     130   non-null   int64
BASE PRICE       130   non-null   int64
SOLD PRICE       130   non-null   int64
dtypes: float64(7), int64(15), object(4)
memory usage: 26.5+ KB
```

Slicing and Indexing of DataFrame

```
ipl_auction_df[0:5]
```

	Sl. NO.	Player Name	Age	...	Auction Year	Base Price	Sold Price
0	1	Abdulla, YA	2	...	2009	50000	50000
1	2	Abdur Razzak	2	...	2008	50000	50000
2	3	Agarkar, AB	2	...	2008	200000	350000
3	4	Ashwin, R	1	...	2011	100000	850000
4	5	Badrinath, S	2	...	2011	100000	800000

```
ipl_auction_df[-5:]
```

	Sl. NO.	Player Name	Age	...	Auction Year	Base Price	Sold Price
125	126	Yadav, AS	2	...	2010	50000	750000
126	127	Younis Khan	2	...	2008	225000	225000
127	128	Yuvraj Singh	2	...	2011	400000	1800000
128	129	Zaheer Khan	2	...	2008	200000	450000
129	130	Zoysa, DNT	2	...	2008	100000	110000

Value Counts and Cross Tabulations

```
ipl_auction_df.COUNTRY.value_counts()
```

```
IND      53
AUS      22
SA       16
SL       12
PAK       9
NZ        7
WI        6
ENG       3
ZIM       1
BAN       1
Name: COUNTRY, dtype: int64
```

```
pd.crosstab( ipl_auction_df['AGE'], ipl_auction_df['PLAYING ROLE'] )
```

Age \ Playing Role	Allrounder	Batsman	Bowler	W. Keeper
1	4	5	7	0
2	25	21	29	11
3	6	13	8	1

Cross Tabulation feature helps in find occurrences for the combination of values for two columns.

Sorting DataFrame by Column Values

```
ipl_auction_df[['PLAYER NAME', 'SOLD PRICE']].sort_values
('SOLD PRICE')[0:5]
```

	Player Name	Sold Price
73	Noffke, AA	20000
46	Kamran Khan	24000
0	Abdulla, YA	50000
1	Abdur Razzak	50000
118	Van der Merwe	50000

```
ipl_auction_df[['PLAYER NAME', 'SOLD PRICE']].sort_values('SOLD
PRICE', ascending = False)[0:5]
```

	Player Name	Sold Price
93	Sehwag, V	1800000
127	Yuvraj Singh	1800000
50	Kohli, V	1800000
111	Tendulkar, SR	1800000
113	Tiwary, SS	1600000

To sort the records in descending order, pass *False* to *ascending* parameter.

Creating New Columns

```
ipl_auction_df['premium'] = ipl_auction_df['SOLD PRICE'] -
                            ipl_auction_df['BASE PRICE']
```

```
ipl_auction_df[['PLAYER NAME', 'BASE PRICE', 'SOLD PRICE',
'premium']][0:5]
```

	Player Name	Base Price	Sold Price	Premium
0	Abdulla, YA	50000	50000	0
1	Abdur Razzak	50000	50000	0
2	Agarkar, AB	200000	350000	150000
3	Ashwin, R	100000	850000	750000
4	Badrinath, S	100000	800000	700000

Grouping and Aggregating

To find average *SOLD PRICE* for each age category, group all records by *AGE* and then apply *mean()* on *SOLD PRICE* column.

```
soldprice_by_age = ipl_auction_df.groupby('AGE')['SOLD PRICE'].
                   mean().reset_index()
print(soldprice_by_age)
```

	Age	Sold Price
0	1	720250.000000
1	2	484534.883721
2	3	520178.571429

Grouping and Aggregating (Cntd.)

```
soldprice_by_age_role = ipl_auction_df.groupby(['AGE', 'PLAYING
                        ROLE'])['SOLD PRICE'].mean().reset_index()
print(soldprice_by_age_role)
```

	Age	Playing Role	Sold Price
0	1	Allrounder	587500.000
1	1	Batsman	1110000.000
2	1	Bowler	517714.286
3	2	Allrounder	449400.000
4	2	Batsman	654761.905
5	2	Bowler	397931.034
6	2	W. Keeper	467727.273
7	3	Allrounder	766666.667
8	3	Batsman	457692.308
9	3	Bowler	414375.000
10	3	W. Keeper	700000.000

Joining DataFrames

```
soldprice_comparison = soldprice_by_age_role.merge (soldprice_by_age,
                                                    on = 'AGE',
                                                   how = 'outer')
```

```
soldprice_comparison
```

	Age	Playing Role	Sold Price_x	Sold Price_y
0	1	Allrounder	587500.000	720250.000
1	1	Batsman	1110000.000	720250.000
2	1	Bowler	517714.286	720250.000
3	2	Allrounder	449400.000	484534.884
4	2	Batsman	654761.905	484534.884
5	2	Bowler	397931.034	484534.884
6	2	W. Keeper	467727.273	484534.884
7	3	Allrounder	766666.667	520178.571
8	3	Batsman	457692.308	520178.571
9	3	Bowler	414375.000	520178.571
10	3	W. Keeper	700000.000	520178.571

Renaming Columns

```
soldprice_comparison.rename( columns = { 'SOLD PRICE_x': 'SOLD_PRICE_
AGE_ROLE','SOLD PRICE_y': 'SOLD_PRICE_AGE' }, inplace = True )
```

```
soldprice_comparison.head(5)
```

	Age	Playing Role	Sold_price_age_role	Sold_price_age
0	1	Allrounder	587500.000	720250.000
1	1	Batsman	1110000.000	720250.000
2	1	Bowler	517714.286	720250.000
3	2	Allrounder	449400.000	484534.884
4	2	Batsman	654761.905	484534.884

Applying Operations to Multiple Columns

```
soldprice_comparison['change'] = soldprice_comparison.apply(lambda
                                 rec:(rec.SOLD_PRICE_AGE_ROLE -
                                 rec.SOLD_PRICE_AGE) / rec.SOLD_
                                 PRICE_AGE, axis = 1)
```

```
soldprice_comparison
```

	Age	Playing Role	Sold_price_age_role	Sold_price_age	Change
0	1	Allrounder	587500.000	720250.000	–0.184
1	1	Batsman	1110000.000	720250.000	0.541
2	1	Bowler	517714.286	720250.000	–0.281
3	2	Allrounder	449400.000	484534.884	–0.073
4	2	Batsman	654761.905	484534.884	0.351
5	2	Bowler	397931.034	484534.884	–0.179
6	2	W. Keeper	467727.273	484534.884	–0.035
7	3	Allrounder	766666.667	520178.571	0.474
8	3	Batsman	457692.308	520178.571	–0.120
9	3	Bowler	414375.000	520178.571	–0.203
10	3	W. Keeper	700000.000	520178.571	0.346

Filtering Records Based On Conditions

```
ipl_auction_df[ipl_auction_df['SIXERS'] > 80 ][['PLAYER NAME',
'SIXERS']]
```

	Player Name	Sixers
26	Gayle, CH	129
28	Gilchrist, AC	86
82	Pathan, YK	81
88	Raina, SK	97
97	Sharma, RG	82

Removing a Column or a Raw from Dataset

```
ipl_auction_df.drop('Sl.NO.', inplace = True, axis = 1)
```

```
ipl_auction_df.columns
```

```
Index (['PLAYER NAME', 'AGE', 'COUNTRY', 'TEAM', 'PLAYING ROLE',
        'T-RUNS', 'T-WKTS', 'ODI-RUNS-S', 'ODI-SR-B', 'ODI-WKTS',
        'ODI-SR-BL', 'CAPTAINCY EXP', 'RUNS-S', 'HS', 'AVE',
        'SR-B', 'SIXERS', 'RUNS-C', 'WKTS', 'AVE-BL', 'ECON', 'SR-BL',
        'AUCTION YEAR', 'BASE PRICE', 'SOLD PRICE', 'premium'],
         dtype = 'object')
```

To avoid creating new DataFrames and make changes to the existing DataFrame, there is another parameter *inplace* available, which can be set to *True.*

Handling Missing Values

Autos-mpg dataset: It contains information about different cars and their characteristics

1. mpg – miles per gallon
2. cylinders – Number of cylinders (values between 4 and 8)
3. displacement – Engine displacement (cu. inches)
4. horsepower – Engine horsepower
5. weight – Vehicle weight (lbs.)
6. acceleration – Time to accelerate from 0 to 60 mph (sec.)
7. year – Model year (modulo 100)
8. origin – Origin of car (1. American, 2. European, 3. Japanese)
9. name – Vehicle name

Summary of Autos-mpg data

```
autos.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
mpg              398  non-null  float64
cylinders        398  non-null  int64
displacement     398  non-null  float64
horsepower       398  non-null  object
weight           398  non-null  float64
acceleration     398  non-null  float64
year             398  non-null  int64
origin           398  non-null  int64
name             398  non-null  object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

Here the column *horsepower* has been inferred as *object*, whereas it should have been inferred as *float64*. This may be because some of the rows contain non-numeric values in the *horsepower* column.

Assigning Names to the Columns

```
autos.columns = ['mpg','cylinders', 'displacement',
                 'horsepower', 'weight', 'acceleration',
                 'year', 'origin', 'name']
autos.head(5)
```

	mpg	cylinders	displacement	...	year	origin	name
0	18.0	8	307.0	...	70	1	chevrolet chevelle malibu
1	15.0	8	350.0	...	70	1	buick skylark 320
2	18.0	8	318.0	...	70	1	plymouth satellite
3	16.0	8	304.0	...	70	1	amc rebel sst
4	17.0	8	302.0	...	70	1	ford torino

5 rows × 9 columns

Handling Missing Values (Cntd.)

```
autos[“horsepower”] = pd.to_numeric(autos[“horsepower”],
errors = ‘corece’) autos.info()
```

```
<class ‘pandas.core.frame.DataFrame’>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
mpg                 398  non-null  float64
cylinders           398  non-null  int64
displacement        398  non-null  float64
horsepower          392  non-null  float64
weight              398  non-null  float64
acceleration        398  non-null  float64
year                398  non-null  int64
origin              398  non-null  int64
name                398  non-null  object
dtypes: float64(5), int64(3), object(1)
memory usage: 28.1+ KB
```

Handling Missing Values (Cntd.)

```
autos[autos.horsepower.isnull()]
```

	mpg	cylinders	displacement	...	year	origin	name
32	25.0	4	98.0	...	71	1	ford pinto
126	21.0	6	200.0	...	74	1	ford maverick
330	40.9	4	85.0	...	80	2	renault lecar deluxe
336	23.6	4	140.0	...	80	1	ford mustang cobra
354	34.5	4	100.0	...	81	2	renault 18i
374	23.0	4	151.0	...	82	1	ame concord di

6 rows × 9 columns

Handling Missing Values (Cntd.)

```
autos = autos.dropna(subset = ['horsepower']
```

```
autos[autos.horsepower.isnull()]
```

mpg	cylinders	displacement	...	year	origin	name

0 rows × 9 columns

Exploration of Data Using Visualization

Data Visualization is useful

- To gain insights in data
- To understand what happened in the past in a given context
- For feature engineering

Drawing Plots

```
import matplotlib.pyplot as plt
import seaborn as sn
%matplotlib inline
```

Bar Chart

- A frequency chart for qualitative variables (or categorical variables)
- Used to assess the most-occurring and least-occurring categories within a dataset

```
sn.barplot(x = 'AGE', y = 'SOLD PRICE', data = soldprice_by_age);
```

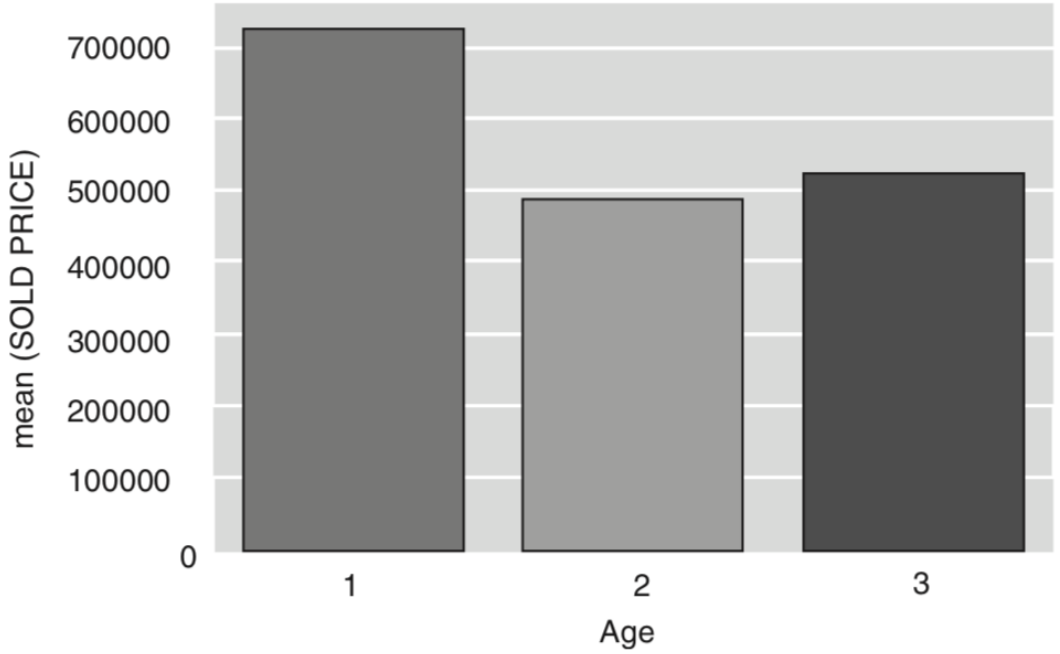
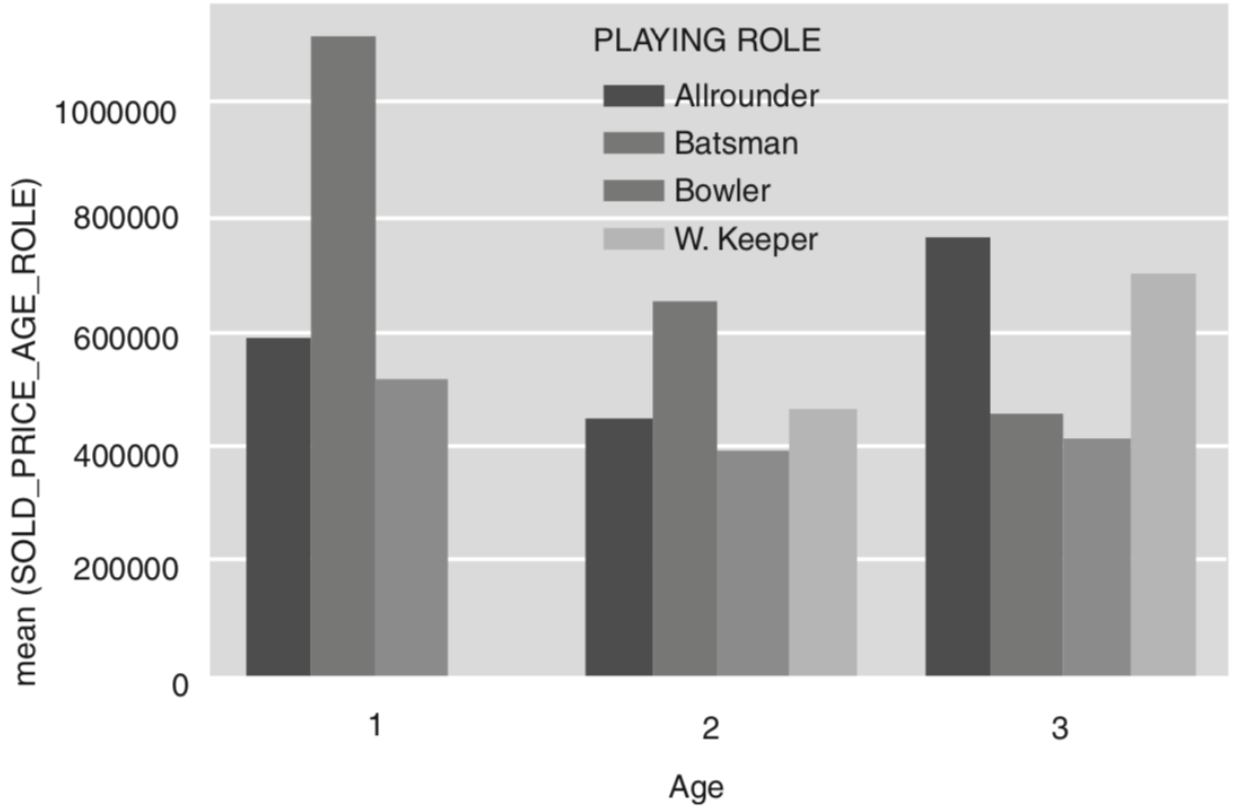


FIGURE 2.3 Bar plot for average sold price versus age.

Bar Chart (Cntd.)

```
sn.barplot(x = 'AGE', y = 'SOLD_PRICE_AGE_ROLE', hue = 'PLAYING
ROLE', data = soldprice_comparison);
```



Histogram

- A plot that shows the frequency distribution of a set of continuous variable.
- Gives an insight into the underlying distribution of the variable, outliers, skewness etc.

```
plt.hist( ipl_auction_df['SOLD PRICE'], bins = 20 );
```

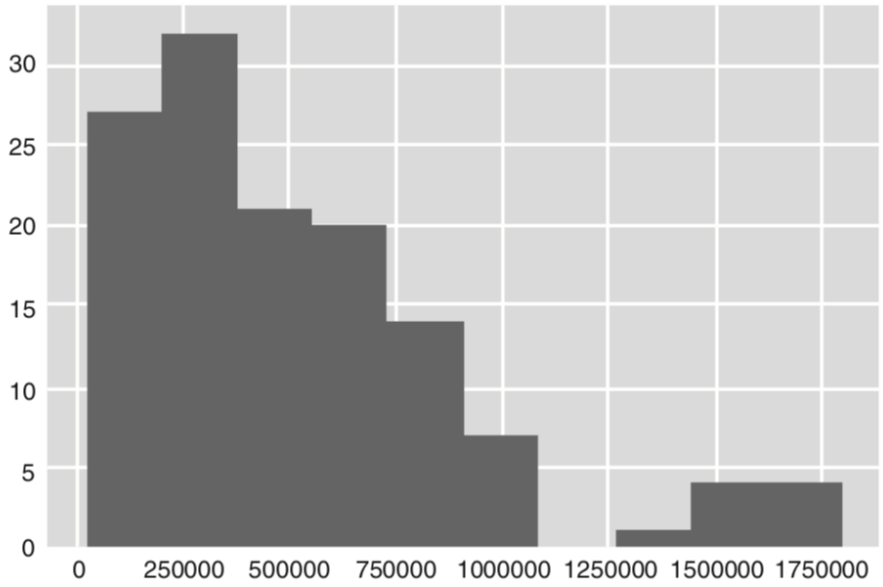


FIGURE 2.5 Histogram for SOLD PRICE.

Note: By default, *plt.hist()* function creates 10 bins in the histogram. To create more bins, the bins parameter can be set in the *hist()* method accordingly.

Distribution or Density Plot

- Depicts the distribution of data over a continuous interval
- Gives insights into what might be the distribution of the population

```
sn.distplot(ipl_auction_df['SOLD PRICE']);
```

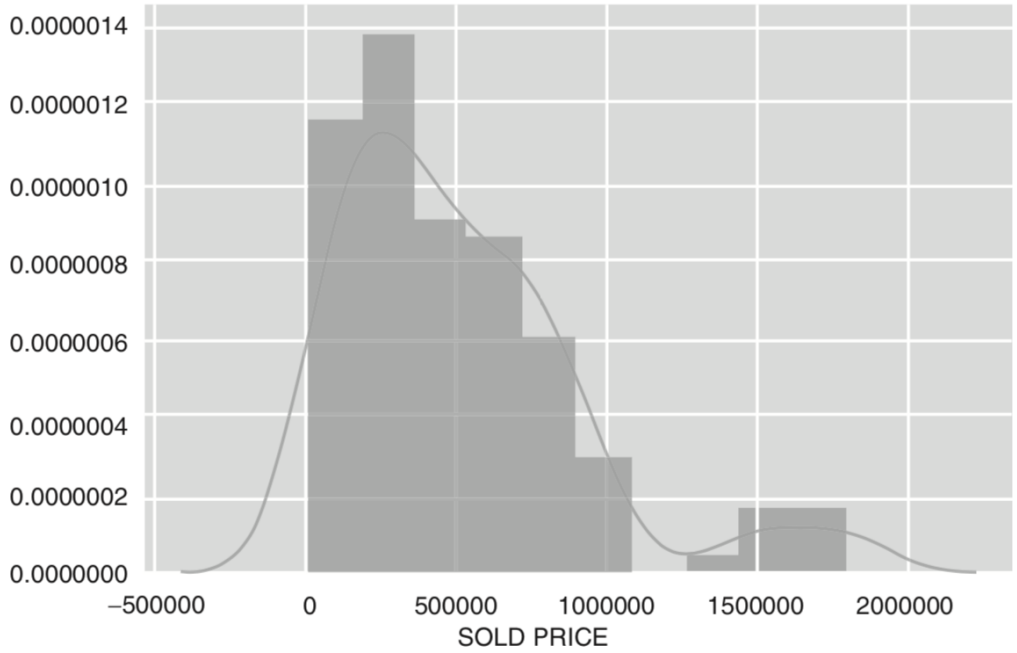


FIGURE 2.7 Distribution plot for SOLD PRICE.

Box Plot

Box plot is designed by identifying the following descriptive statistics:

1. Lower quartile (1st quartile), median and upper quartile (3rd quartile).

2. Lowest and highest values.

3. Inter-quartile range (IQR).

```
box = sn.boxplot(ipl_auction_df['SOLD PRICE']);
```

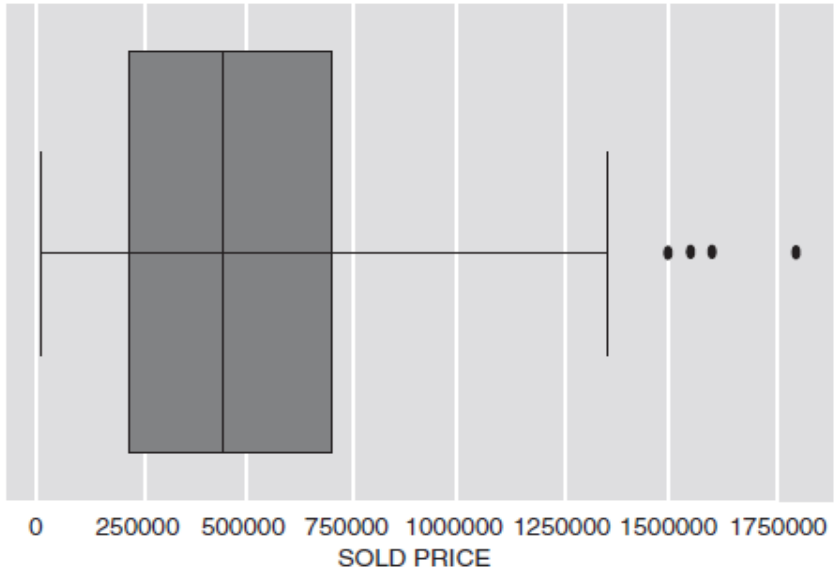


FIGURE 2.8 Box plot for SOLD PRICE.

Box Plot (Cntd.)
IQR

- IQR is the distance (difference) between the 3rd quartile and 1st quartile.
- The length of the box is equivalent to IQR.
- The whisker of the box plot extends till Q1 – 1.5IQR and Q3 + 1.5IQR
- Observations beyond these two limits are potential outliers.

- The *caps* key in box variable returns the min and max values of the distribution.

```
[item.get_ydata()[0] for item in box['caps']]
```

```
[20000.0, 1350000.0]
```

Box Plot (Cntd.)
IQR

- The *whiskers* key in box variable returns the values of the distribution at 25 and 75 quantiles.

```
[item.get_ydata()[0] for item in box['whiskers']]
```

```
[225000.0, 700000.0]
```

So, inter-quartile range (IQR) is 700,000 – 225,000 = 475,000.

- The *medians* key in box variable returns the median value of the distribution.

```
[item.get_ydata()[0] for item in box['medians']]
```

```
[437500.0]
```

Box Plot (Cntd.)

The box plot in Figure 2.8 shows that some of the players are auctioned at *SOLD PRICE*, which seem to be outliers. Let us find out the player names.

```
ipl_auction_df[ipl_auction_df['SOLD PRICE'] > 1350000.0][['PLAYER
NAME', 'PLAYING ROLE', 'SOLD PRICE']]
```

	Player Name	Playing Role	Sold Price
15	Dhoni, MS	W. Keeper	1500000
23	Flintoff, A	Allrounder	1550000
50	Kohli, V	Batsman	1800000
83	Pietersen, KP	Batsman	1550000
93	Sehwag, V	Batsman	1800000
111	Tendulkar, SR	Batsman	1800000
113	Tiwary, SS	Batsman	1600000
127	Yuvraj Singh	Batsman	1800000

Comparing Distributions

```
sn.distplot(ipl_auction_df[ipl_auction_df['CAPTAINCY EXP'] == 1]
           ['SOLD PRICE'],
           color = 'y',
           label = 'Captaincy Experience')
sn.distplot(ipl_auction_df[ipl_auction_df['CAPTAINCY EXP'] == 0]
           ['SOLD PRICE'],
           color = 'r',
           label = 'No Captaincy Experience');
plt.legend();
```

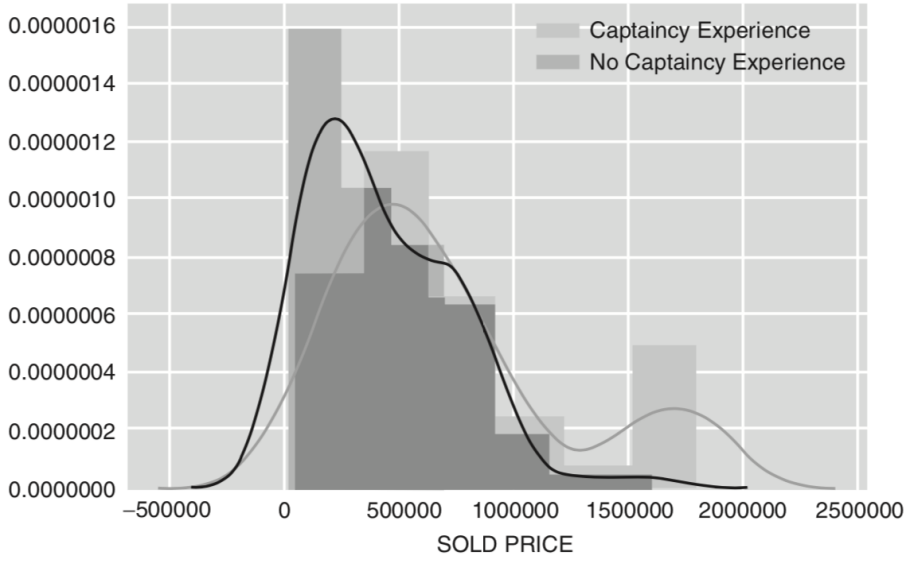


FIGURE 2.10 Distribution plots comparing SOLD PRICE with and without captaincy experience.

Comparing Distributions (Cntd.)

```
sn.boxplot(x = 'PLAYING ROLE', y = 'SOLD PRICE',
           data = ipl_auction_df);
```

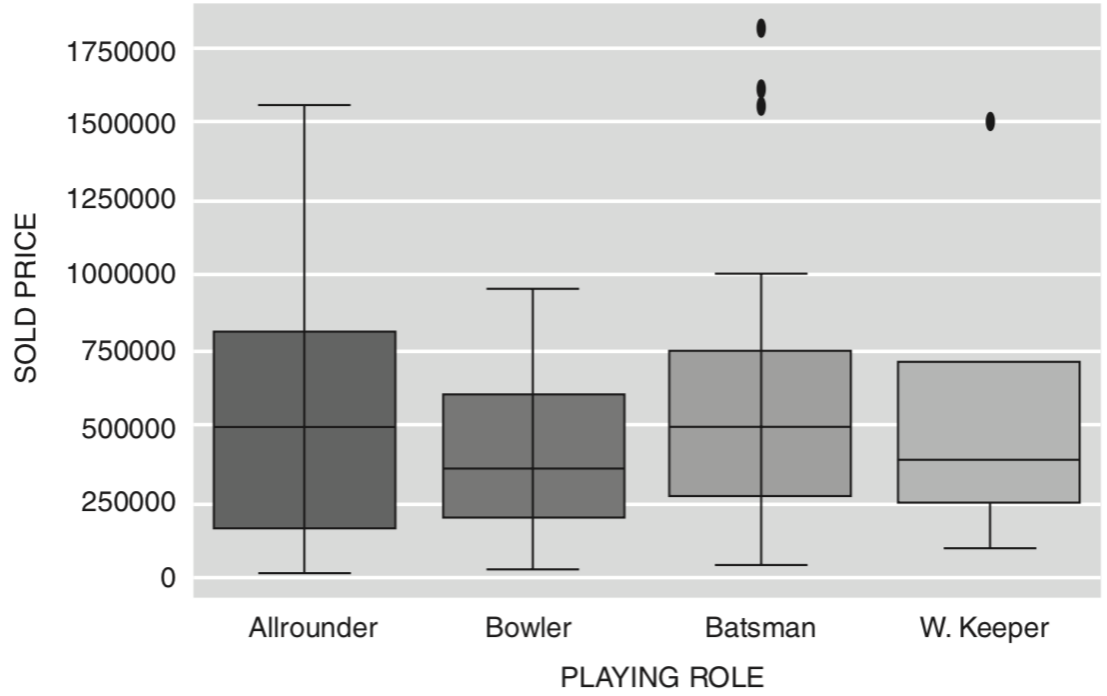


FIGURE 2.11 Box plot of SOLD PRICE for different playing roles.

Few Observations from the plot in Figure 2.11

1. The median SOLD PRICE for all rounders and batsmen are higher than bowlers and wicket keepers.
2. All rounders who are paid more than 1,35,0000 USD are not considered outliers. All rounders have relatively high variance.
3. There are outliers in batsman and wicket keeper category.
4. In the outlier analysis we have already found out that MS DHONI is an outlier in the wicket keeper category.

Scatter Plot

```
ipl_batsman_df = ipl_auction_df[ipl_auction_df['PLAYING ROLE'] ==
                 'Batsman']
```

```
plt.scatter(x = ipl_batsman_df.SIXERS,
            y = ipl_batsman_df['SOLD PRICE']);
```

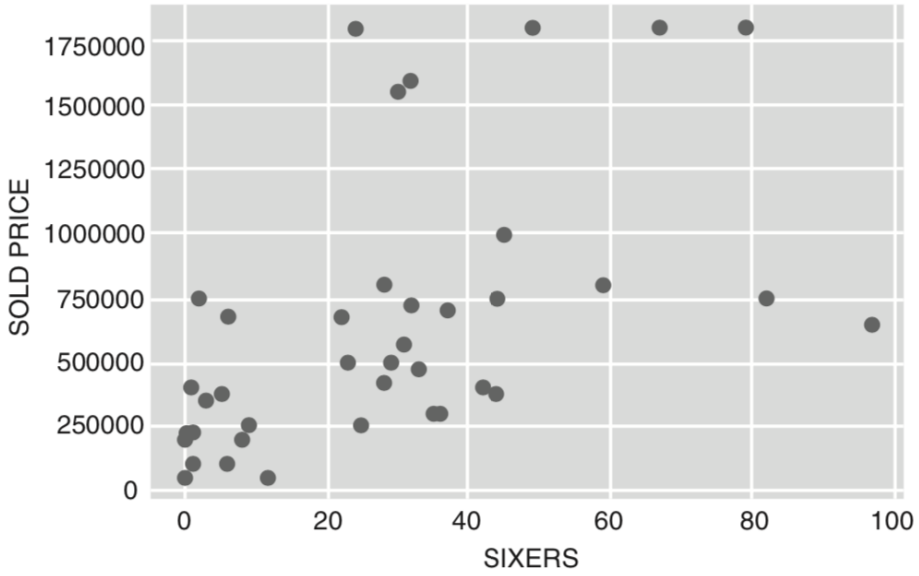


FIGURE 2.12 Scatter plot for SOLD PRICE versus SIXERS.

- Two variables are plotted along two axes
- Can reveal correlation present between two variables, if any
- Useful for assessing the strength of the relationship and to find the outliers in the data
- Mostly used during regression model building to decide on the initial model

Scatter Plot (Cntd.)

To draw the direction of relationship between the variables, *regplot() of seaborn can be used*

```
sn.regplot( x = 'SIXERS',
            y = 'SOLD PRICE',
            data = ipl_batsman_df );
```

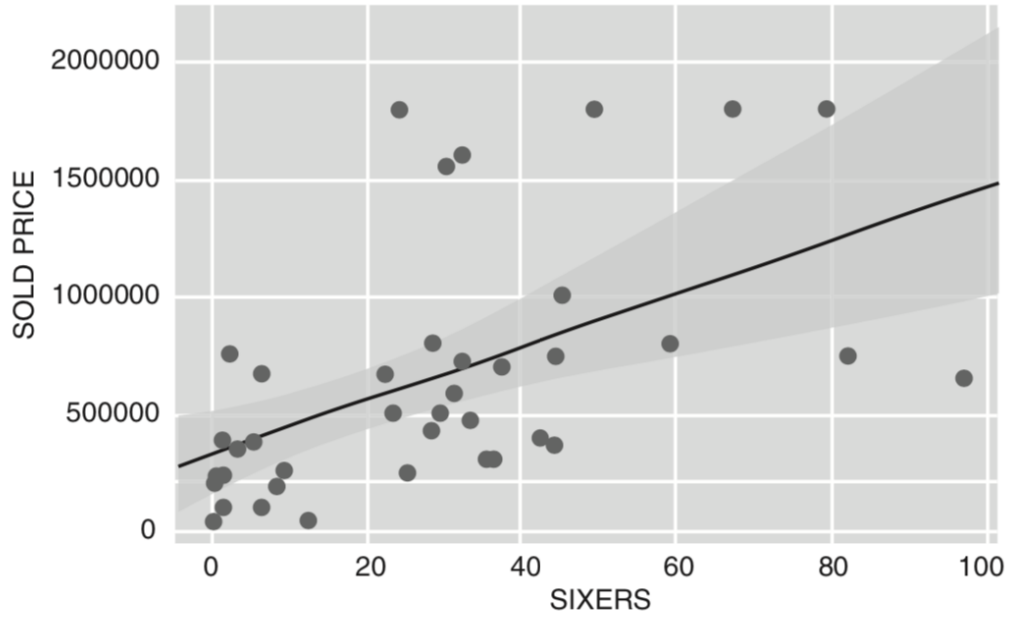


FIGURE 2.13 Scatter plot for SOLD PRICE versus SIXERS with a regression line.

Pair Plot

```
influential_features = ['SR-B', 'AVE', 'SIXERS', 'SOLD PRICE']
```

```
sn.pairplot(ipl_auction_df[influential_features], size=2)
```

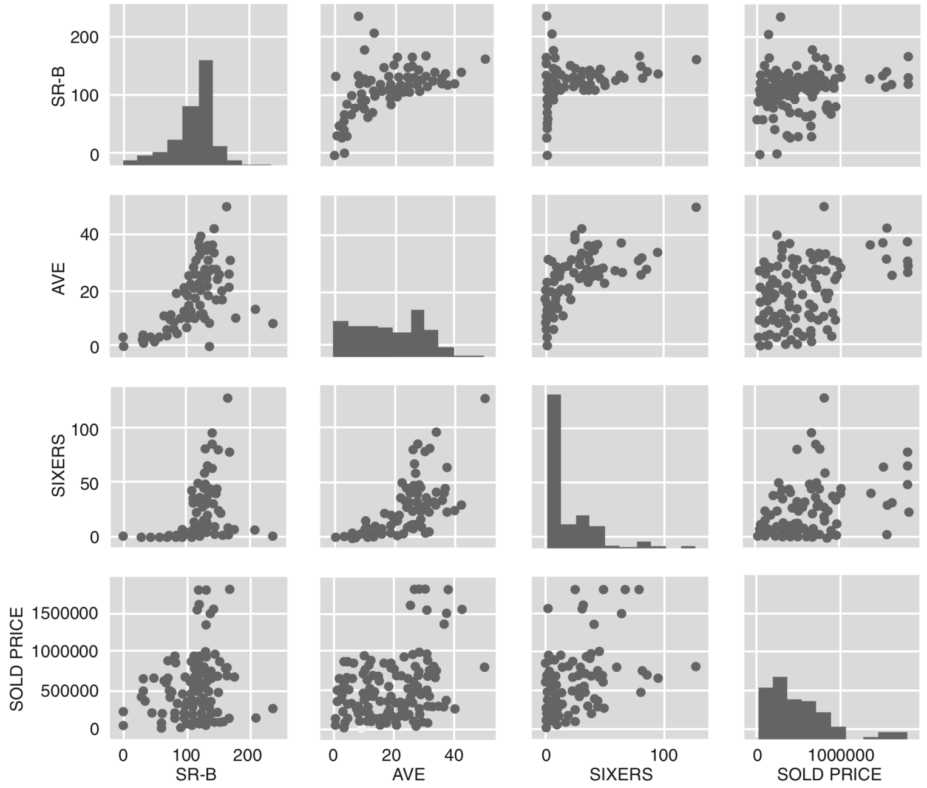


FIGURE 2.14 Pair plot between variables.

Correlation and Heatmap

- Correlation is used for measuring the strength and direction of the linear relationship between two continuous random variables X and Y
- It is a statistical measure that indicates the extent to which two variables change together
- Positive correlation – the variables increase/ decrease together
- Negative correlation – if one variable increases, the other decreases
- The correlation value lies between -1.0 and 1.0. The sign indicates whether it is positive or negative correlation.
- -1.0 indicates a perfect negative correlation, whereas +1.0 indicates perfect positive correlation.

Correlation and Heatmap (Cntd.)

```
ipl_auction_df[influential_features].corr()
```

	SR-B	AVE	Sixers	Sold Price
SR-B	1.000000	0.583579	0.425394	0.184278
AVE	0.583579	1.000000	0.705365	0.396519
Sixers	0.425394	0.705365	1.000000	0.450609
Sold Price	0.184278	0.396519	0.450609	1.000000

Correlation and Heatmap (Cntd.)

```
sn.heatmap(ipl_auction_df[influential_features].corr(), annot=True);
```

FIGURE 2.15 Heatmap of correlation values.

Conclusion

- The Objective of descriptive analytics is simple comprehension of data using data summarization, basic statistical measures and visualization.
- Pandas library provides support for DataFrame and its operations like filtering, grouping, aggregations, sorting, joining etc.
- *Matplotlib* and *seaborn* are the most widely used libraries for data visualization.
- Plots like histograms, distribution plots, bar charts, heatmap, scatter plot can be created to find insights during exploratory analysis.

Thank You!