## Bike Sharing Dataset

#### Import pandas, numpy, seaborn, matplotlib.pyplot packages

In [60]:

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**%**matplotlib inline

**import** seaborn **as** sns

**from** warnings **import** filterwarnings

filterwarnings('ignore')

#### Importing Dataset

In [61]:

df **=** pd.read\_csv('hour.csv')

df.head()

Out[61]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **instant** | **dteday** | **season** | **yr** | **mnth** | **hr** | **holiday** | **weekday** | **workingday** | **weathersit** | **temp** | **atemp** | **hum** | **windspeed** | **casual** | **registered** | **cnt** |
| **0** | 1 | 2011-01-01 | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 | 0.0 | 3 | 13 | 16 |
| **1** | 2 | 2011-01-01 | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 8 | 32 | 40 |
| **2** | 3 | 2011-01-01 | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 5 | 27 | 32 |
| **3** | 4 | 2011-01-01 | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 3 | 10 | 13 |
| **4** | 5 | 2011-01-01 | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 0 | 1 | 1 |

In [62]:

df.tail()

Out[62]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **instant** | **dteday** | **season** | **yr** | **mnth** | **hr** | **holiday** | **weekday** | **workingday** | **weathersit** | **temp** | **atemp** | **hum** | **windspeed** | **casual** | **registered** | **cnt** |
| **17374** | 17375 | 2012-12-31 | 1 | 1 | 12 | 19 | 0 | 1 | 1 | 2 | 0.26 | 0.2576 | 0.60 | 0.1642 | 11 | 108 | 119 |
| **17375** | 17376 | 2012-12-31 | 1 | 1 | 12 | 20 | 0 | 1 | 1 | 2 | 0.26 | 0.2576 | 0.60 | 0.1642 | 8 | 81 | 89 |
| **17376** | 17377 | 2012-12-31 | 1 | 1 | 12 | 21 | 0 | 1 | 1 | 1 | 0.26 | 0.2576 | 0.60 | 0.1642 | 7 | 83 | 90 |
| **17377** | 17378 | 2012-12-31 | 1 | 1 | 12 | 22 | 0 | 1 | 1 | 1 | 0.26 | 0.2727 | 0.56 | 0.1343 | 13 | 48 | 61 |
| **17378** | 17379 | 2012-12-31 | 1 | 1 | 12 | 23 | 0 | 1 | 1 | 1 | 0.26 | 0.2727 | 0.65 | 0.1343 | 12 | 37 | 49 |

**It is a Regression Problem - the Dependent variable is cnt(ie, count of total rental bikes)**

* By viewing Data we can say on Working day the count of total rental bikes(including both casual and registered) is high
* **Shape of Dataset**

In [63]:

df.shape

Out[63]:

(17379, 17)

* **The dataset has total 17379 rows & 17 attributes(columns)**
* **Checking Information of Dataset**

In [64]:

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 17379 entries, 0 to 17378  
Data columns (total 17 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 instant 17379 non-null int64   
 1 dteday 17379 non-null object   
 2 season 17379 non-null int64   
 3 yr 17379 non-null int64   
 4 mnth 17379 non-null int64   
 5 hr 17379 non-null int64   
 6 holiday 17379 non-null int64   
 7 weekday 17379 non-null int64   
 8 workingday 17379 non-null int64   
 9 weathersit 17379 non-null int64   
 10 temp 17379 non-null float64  
 11 atemp 17379 non-null float64  
 12 hum 17379 non-null float64  
 13 windspeed 17379 non-null float64  
 14 casual 17379 non-null int64   
 15 registered 17379 non-null int64   
 16 cnt 17379 non-null int64   
dtypes: float64(4), int64(12), object(1)  
memory usage: 2.3+ MB

* Dataset has 4 Float columns, 12 integer columns and 1 object (string) Columns
* checking null values

In [65]:

df.isnull().sum()

Out[65]:

instant 0  
dteday 0  
season 0  
yr 0  
mnth 0  
hr 0  
holiday 0  
weekday 0  
workingday 0  
weathersit 0  
temp 0  
atemp 0  
hum 0  
windspeed 0  
casual 0  
registered 0  
cnt 0  
dtype: int64

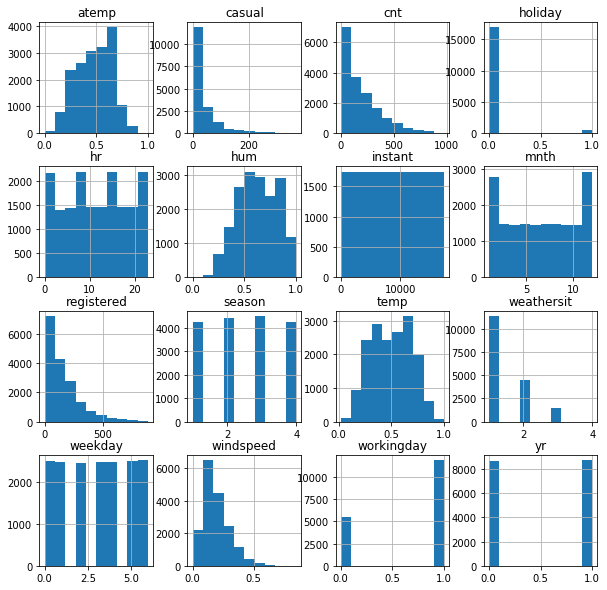
* There is no null values

**Plotting Histogram**

In [66]:

df.hist(figsize**=**(10,10))

plt.show()



### Inference from Histogram:

* Holiday, Season, weathersit, weekday, workingday, yr are Categorical Variables which are in Encoded format
* instant is irrelevent column

In [67]:

plt.figure(figsize**=**(15,10))

sns.heatmap(df.corr(), annot **=** **True**)

Out[67]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x23dec9f4c08>



Important Varibale by correlation matrix is:

* registered, casual, hum, atemp, temp, hr is Important variable for predicting cnt

## Data preprocessing

* Dropping irrelevent columns

In [68]:

df.columns

Out[68]:

Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'hr', 'holiday', 'weekday',  
 'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',  
 'casual', 'registered', 'cnt'],  
 dtype='object')

In [69]:

df **=** df.drop(['instant','dteday'], axis**=**1)

df.head()

Out[69]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **season** | **yr** | **mnth** | **hr** | **holiday** | **weekday** | **workingday** | **weathersit** | **temp** | **atemp** | **hum** | **windspeed** | **casual** | **registered** | **cnt** |
| **0** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 | 0.0 | 3 | 13 | 16 |
| **1** | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 8 | 32 | 40 |
| **2** | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 5 | 27 | 32 |
| **3** | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 3 | 10 | 13 |
| **4** | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 0 | 1 | 1 |

### Spliting data

In [70]:

x **=** df.drop(['cnt'], axis**=**1)

x.head()

*# x = df[['registered', 'casual', 'hum', 'atemp', 'temp', 'hr']]*

*# if predicting with important varibales - accuracy remains same (1.0)*

Out[70]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **season** | **yr** | **mnth** | **hr** | **holiday** | **weekday** | **workingday** | **weathersit** | **temp** | **atemp** | **hum** | **windspeed** | **casual** | **registered** |
| **0** | 1 | 0 | 1 | 0 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.81 | 0.0 | 3 | 13 |
| **1** | 1 | 0 | 1 | 1 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 8 | 32 |
| **2** | 1 | 0 | 1 | 2 | 0 | 6 | 0 | 1 | 0.22 | 0.2727 | 0.80 | 0.0 | 5 | 27 |
| **3** | 1 | 0 | 1 | 3 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 3 | 10 |
| **4** | 1 | 0 | 1 | 4 | 0 | 6 | 0 | 1 | 0.24 | 0.2879 | 0.75 | 0.0 | 0 | 1 |

In [71]:

y **=** df.iloc[:,**-**1:]

y.head()

Out[71]:

|  |  |
| --- | --- |
|  | **cnt** |
| **0** | 16 |
| **1** | 40 |
| **2** | 32 |
| **3** | 13 |
| **4** | 1 |

### Split into test and train dataset (70-30 ratio)

In [72]:

**from** sklearn.model\_selection **import** train\_test\_split

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x,y,test\_size**=**0.3,random\_state**=**0)

In [73]:

print("Dataset shape:", df.shape)

print("Input Features shape: ", x\_train.shape, y\_train.shape)

print("Output Features shape: ", x\_test.shape, y\_test.shape)

Dataset shape: (17379, 15)  
Input Features shape: (12165, 14) (12165, 1)  
Output Features shape: (5214, 14) (5214, 1)

### Applying Linear Regression

In [74]:

**from** sklearn.linear\_model **import** LinearRegression

lin **=** LinearRegression()

#### Fitting model

In [75]:

lin.fit(x\_train,y\_train)

Out[75]:

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

#### Predicting values

In [76]:

pred **=** lin.predict(x\_test)

In [77]:

pred

Out[77]:

array([[ 7.],  
 [ 5.],  
 [743.],  
 ...,  
 [499.],  
 [302.],  
 [229.]])

#### Finding score

In [78]:

**from** sklearn.metrics **import** r2\_score

In [79]:

r2\_score(y\_test, pred)

Out[79]:

1.0

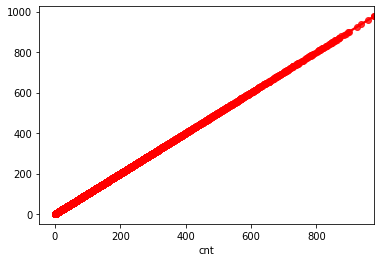
#### Ploting graph for test values

In [80]:

sns.regplot(y\_test,pred,color**=**'r')

Out[80]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x23ded6592c8>



## Diabetes Dataset

### Objective

The objective of the dataset is to diagnostically predict whether or not a patient has diabetes

#### Import pandas, numpy, seaborn, matplotlib.pyplot packages

In [1]:

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**%**matplotlib inline

**import** seaborn **as** sns

**from** warnings **import** filterwarnings

filterwarnings('ignore')

In [2]:

df **=** pd.read\_csv('diabetes.csv')

df.head()

Out[2]:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** | **Outcome** |
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 | 1 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 | 0 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 | 1 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 | 0 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 | 1 |

**It is a Classification Problem - the Dependent variable is Outcome**

* **Shape of Dataset**

In [3]:

df.shape

Out[3]:

(768, 9)

* **The dataset has total 768 rows & 9 Attributes**
* **Checking Information of Dataset**

In [4]:

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 768 entries, 0 to 767  
Data columns (total 9 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Pregnancies 768 non-null int64   
 1 Glucose 768 non-null int64   
 2 BloodPressure 768 non-null int64   
 3 SkinThickness 768 non-null int64   
 4 Insulin 768 non-null int64   
 5 BMI 768 non-null float64  
 6 DiabetesPedigreeFunction 768 non-null float64  
 7 Age 768 non-null int64   
 8 Outcome 768 non-null int64   
dtypes: float64(2), int64(7)  
memory usage: 54.1 KB

* Dataset has 2 Float columns, 7 integer columns

#### checking null values

In [5]:

df.isnull().sum()

Out[5]:

Pregnancies 0  
Glucose 0  
BloodPressure 0  
SkinThickness 0  
Insulin 0  
BMI 0  
DiabetesPedigreeFunction 0  
Age 0  
Outcome 0  
dtype: int64

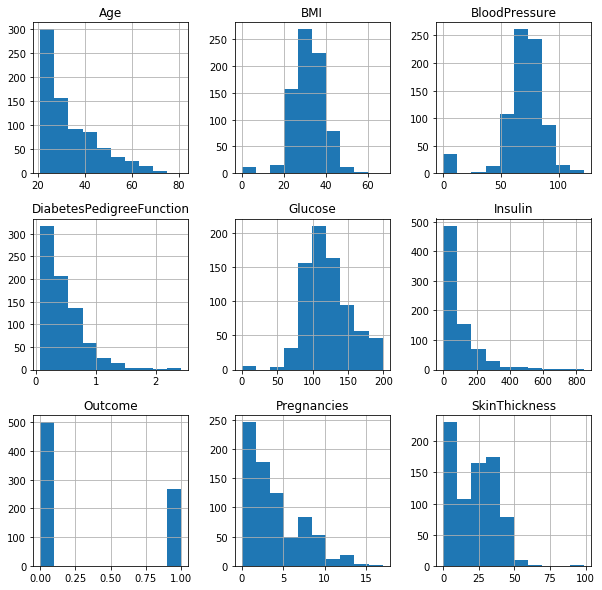
* There is no null values

**Plotting Histogram**

In [6]:

df.hist(figsize**=**(10,10))

plt.show()



### Inference from Histogram:

* Outcome Categorical Variables which is in Encoded format

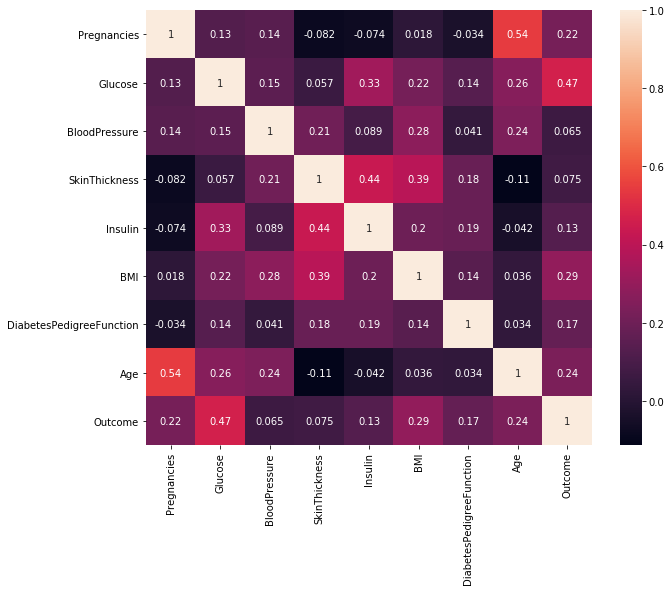
In [9]:

plt.figure(figsize**=**(10,8))

sns.heatmap(df.corr(), annot **=** **True**)

Out[9]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x1bc04253148>



### Spliting data

In [10]:

x **=** df.drop(['Outcome'], axis**=**1)

x.head()

Out[10]:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Pregnancies** | **Glucose** | **BloodPressure** | **SkinThickness** | **Insulin** | **BMI** | **DiabetesPedigreeFunction** | **Age** |
| **0** | 6 | 148 | 72 | 35 | 0 | 33.6 | 0.627 | 50 |
| **1** | 1 | 85 | 66 | 29 | 0 | 26.6 | 0.351 | 31 |
| **2** | 8 | 183 | 64 | 0 | 0 | 23.3 | 0.672 | 32 |
| **3** | 1 | 89 | 66 | 23 | 94 | 28.1 | 0.167 | 21 |
| **4** | 0 | 137 | 40 | 35 | 168 | 43.1 | 2.288 | 33 |

In [11]:

y **=** df.iloc[:,**-**1:]

y.head()

Out[11]:

|  |  |
| --- | --- |
|  | **Outcome** |
| **0** | 1 |
| **1** | 0 |
| **2** | 1 |
| **3** | 0 |
| **4** | 1 |

### Split into test and train dataset (70-30 ratio)

In [12]:

**from** sklearn.model\_selection **import** train\_test\_split

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x,y,test\_size**=**0.3,random\_state**=**0)

In [13]:

print("Dataset shape:", df.shape)

print("Input Features shape: ", x\_train.shape, y\_train.shape)

print("Output Features shape: ", x\_test.shape, y\_test.shape)

Dataset shape: (768, 9)  
Input Features shape: (537, 8) (537, 1)  
Output Features shape: (231, 8) (231, 1)

### Applying Logistic Regression

In [14]:

**from** sklearn.linear\_model **import** LogisticRegression

log **=** LogisticRegression()

#### Fitting model

In [15]:

log.fit(x\_train,y\_train)

Out[15]:

LogisticRegression(C=1.0, class\_weight=None, dual=False, fit\_intercept=True,  
 intercept\_scaling=1, l1\_ratio=None, max\_iter=100,  
 multi\_class='auto', n\_jobs=None, penalty='l2',  
 random\_state=None, solver='lbfgs', tol=0.0001, verbose=0,  
 warm\_start=False)

#### Predicting values

In [16]:

pred **=** log.predict(x\_test)

In [17]:

pred

Out[17]:

array([1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0,  
 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,  
 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1,  
 1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1,  
 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  
 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,  
 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,  
 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0,  
 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,  
 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0], dtype=int64)

#### Finding accuracy score and confusion matrix

In [18]:

**from** sklearn.metrics **import** accuracy\_score, confusion\_matrix

In [19]:

accuracy\_score(y\_test,pred)

Out[19]:

0.7792207792207793

In [20]:

confusion\_matrix(y\_test,pred)

Out[20]:

array([[141, 16],  
 [ 35, 39]], dtype=int64)

## Boston Dataset

#### Import pandas, numpy, seaborn, matplotlib.pyplot packages

In [1]:

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**%**matplotlib inline

**import** seaborn **as** sns

**from** warnings **import** filterwarnings

filterwarnings('ignore')

In [2]:

df **=** pd.read\_csv('boston\_data.csv')

df.head()

Out[2]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **crim** | **zn** | **indus** | **chas** | **nox** | **rm** | **age** | **dis** | **rad** | **tax** | **ptratio** | **black** | **lstat** | **medv** |
| **0** | 0.15876 | 0.0 | 10.81 | 0.0 | 0.413 | 5.961 | 17.5 | 5.2873 | 4.0 | 305.0 | 19.2 | 376.94 | 9.88 | 21.7 |
| **1** | 0.10328 | 25.0 | 5.13 | 0.0 | 0.453 | 5.927 | 47.2 | 6.9320 | 8.0 | 284.0 | 19.7 | 396.90 | 9.22 | 19.6 |
| **2** | 0.34940 | 0.0 | 9.90 | 0.0 | 0.544 | 5.972 | 76.7 | 3.1025 | 4.0 | 304.0 | 18.4 | 396.24 | 9.97 | 20.3 |
| **3** | 2.73397 | 0.0 | 19.58 | 0.0 | 0.871 | 5.597 | 94.9 | 1.5257 | 5.0 | 403.0 | 14.7 | 351.85 | 21.45 | 15.4 |
| **4** | 0.04337 | 21.0 | 5.64 | 0.0 | 0.439 | 6.115 | 63.0 | 6.8147 | 4.0 | 243.0 | 16.8 | 393.97 | 9.43 | 20.5 |

**It is a Regression Problem - the Dependent variable is medv**

* **Shape of Dataset**

In [3]:

df.shape

Out[3]:

(404, 14)

* **The dataset has total 404 rows & 14 attributes**
* **Checking Information of Dataset**

In [4]:

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 404 entries, 0 to 403  
Data columns (total 14 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 crim 404 non-null float64  
 1 zn 404 non-null float64  
 2 indus 404 non-null float64  
 3 chas 404 non-null float64  
 4 nox 404 non-null float64  
 5 rm 404 non-null float64  
 6 age 404 non-null float64  
 7 dis 404 non-null float64  
 8 rad 404 non-null float64  
 9 tax 404 non-null float64  
 10 ptratio 404 non-null float64  
 11 black 404 non-null float64  
 12 lstat 404 non-null float64  
 13 medv 404 non-null float64  
dtypes: float64(14)  
memory usage: 44.3 KB

* Dataset has all float columns

#### checking null values

In [5]:

df.isnull().sum()

Out[5]:

crim 0  
zn 0  
indus 0  
chas 0  
nox 0  
rm 0  
age 0  
dis 0  
rad 0  
tax 0  
ptratio 0  
black 0  
lstat 0  
medv 0  
dtype: int64

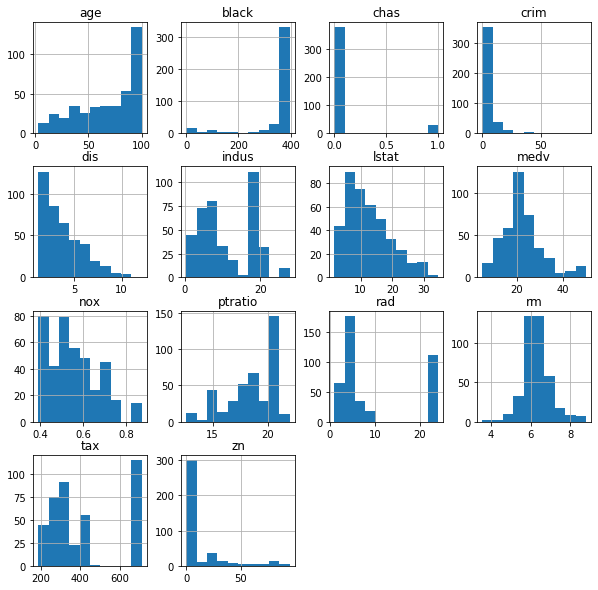
* There is no null values

**Plotting Histogram**

In [6]:

df.hist(figsize**=**(10,10))

plt.show()



### Inference from Histogram:

* chas is Categorical Variables which are in Encoded format

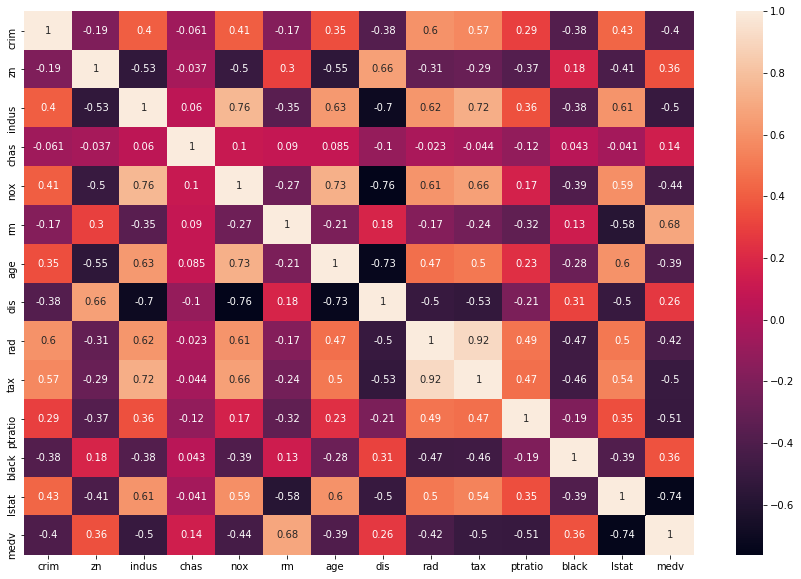
In [8]:

plt.figure(figsize**=**(15,10))

sns.heatmap(df.corr(), annot **=** **True**)

Out[8]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x24bb9296308>



### Spliting data

In [9]:

x **=** df.drop(['medv'], axis**=**1)

x.head()

Out[9]:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **crim** | **zn** | **indus** | **chas** | **nox** | **rm** | **age** | **dis** | **rad** | **tax** | **ptratio** | **black** | **lstat** |
| **0** | 0.15876 | 0.0 | 10.81 | 0.0 | 0.413 | 5.961 | 17.5 | 5.2873 | 4.0 | 305.0 | 19.2 | 376.94 | 9.88 |
| **1** | 0.10328 | 25.0 | 5.13 | 0.0 | 0.453 | 5.927 | 47.2 | 6.9320 | 8.0 | 284.0 | 19.7 | 396.90 | 9.22 |
| **2** | 0.34940 | 0.0 | 9.90 | 0.0 | 0.544 | 5.972 | 76.7 | 3.1025 | 4.0 | 304.0 | 18.4 | 396.24 | 9.97 |
| **3** | 2.73397 | 0.0 | 19.58 | 0.0 | 0.871 | 5.597 | 94.9 | 1.5257 | 5.0 | 403.0 | 14.7 | 351.85 | 21.45 |
| **4** | 0.04337 | 21.0 | 5.64 | 0.0 | 0.439 | 6.115 | 63.0 | 6.8147 | 4.0 | 243.0 | 16.8 | 393.97 | 9.43 |

In [10]:

y **=** df.iloc[:, **-**1:]

y.head()

Out[10]:

|  |  |
| --- | --- |
|  | **medv** |
| **0** | 21.7 |
| **1** | 19.6 |
| **2** | 20.3 |
| **3** | 15.4 |
| **4** | 20.5 |

### Split into test and train dataset (70-30 ratio)

In [11]:

**from** sklearn.model\_selection **import** train\_test\_split

x\_train, x\_test, y\_train, y\_test **=** train\_test\_split(x,y,test\_size**=**0.3,random\_state**=**0)

In [12]:

print("Dataset shape:", df.shape)

print("Input Features shape: ", x\_train.shape, y\_train.shape)

print("Output Features shape: ", x\_test.shape, y\_test.shape)

Dataset shape: (404, 14)  
Input Features shape: (282, 13) (282, 1)  
Output Features shape: (122, 13) (122, 1)

### Applying Linear Regression

In [13]:

**from** sklearn.linear\_model **import** LinearRegression

lin **=** LinearRegression()

#### Fitting model

In [14]:

lin.fit(x\_train,y\_train)

Out[14]:

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

#### Predicting values

In [17]:

pred **=** lin.predict(x\_test)

#### Finding score

In [18]:

**from** sklearn.metrics **import** r2\_score

In [19]:

r2\_score(y\_test, pred)

Out[19]:

0.750266989310347

#### Ploting graph for test values

In [20]:

sns.regplot(y\_test, pred, color**=**'g')

Out[20]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x24bc3d20548>

