In [1]:

```
%config IPCompleter.greedy=True
```

In [2]:

```
import numpy as np
import os
import pandas as pd
import matplotlib.pyplot as plt
```

In [4]:

```
os.getcwd()
```

Out[4]:

```
'C:\\Users\\Ravi Teja\\Documents'
```

In [5]:

```
os.chdir(r"D:\Simplilearn\DS with Python\MovieLens Project")
```

In [6]:

```
os.getcwd()
```

Out[6]:

```
'D:\\Simplilearn\\DS with Python\\MovieLens Project'
```

Reading Movies.dat file

In [7]:

```
movie_dataset=pd.read_table(r"./Datasets/movies.dat",sep="::",header=None,names=['Movie ID','Title','Genres'])
movie_dataset.shape
```

```
C:\Users\Ravi Teja\anaconda3\lib\site-packages\ipykernel_launcher.py:1: Pa
rserWarning: Falling back to the 'python' engine because the 'c' engine do
es not support regex separators (separators > 1 char and different from
'\s+' are interpreted as regex); you can avoid this warning by specifying
engine='python'.
  """Entry point for launching an IPython kernel.
```

Out[7]:

```
(3883, 3)
```

Reading Ratings.dat file

In [8]:

```
ratings_dataset=pd.read_table(r"./Datasets/ratings.dat",sep="::",header=None,names=['User ID','Movie ID','Rating','Timestamp'])
```

```
C:\Users\Ravi Teja\anaconda3\lib\site-packages\ipykernel_launcher.py:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
  """Entry point for launching an IPython kernel.
```

In [9]:

```
ratings_dataset.shape
```

Out[9]:

```
(1000209, 4)
```

Reading Users.dat file

In [10]:

```
users_dataset=pd.read_table(r"./Datasets/users.dat",sep="::",header=None,names=['User ID','Gender','Age','Occupation','Zip Code'])
```

```
C:\Users\Ravi Teja\anaconda3\lib\site-packages\ipykernel_launcher.py:1: ParserWarning: Falling back to the 'python' engine because the 'c' engine does not support regex separators (separators > 1 char and different from '\s+' are interpreted as regex); you can avoid this warning by specifying engine='python'.
  """Entry point for launching an IPython kernel.
```

In [11]:

```
users_dataset.head()
```

Out[11]:

	User ID	Gender	Age	Occupation	Zip Code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

In [12]:

```
movie_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   Movie ID  3883 non-null   int64 
 1   Title     3883 non-null   object
 2   Genres    3883 non-null   object
dtypes: int64(1), object(2)
memory usage: 91.1+ KB
```

In [13]:

```
users_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6040 entries, 0 to 6039
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   User ID     6040 non-null   int64 
 1   Gender      6040 non-null   object
 2   Age         6040 non-null   int64 
 3   Occupation  6040 non-null   int64 
 4   Zip Code    6040 non-null   object
dtypes: int64(3), object(2)
memory usage: 236.1+ KB
```

In [14]:

```
ratings_dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000209 entries, 0 to 1000208
Data columns (total 4 columns):
 #   Column     Non-Null Count    Dtype 
---  ------     --------------    ----- 
 0   User ID    1000209 non-null  int64 
 1   Movie ID   1000209 non-null  int64 
 2   Rating     1000209 non-null  int64 
 3   Timestamp  1000209 non-null  int64 
dtypes: int64(4)
memory usage: 30.5 MB
```

Creating master data with required columns

In [15]:

```
Master_data=(movie_dataset.merge(ratings_dataset,how="inner")).merge(users_dataset,how=
"inner")[['Movie ID','Title','User ID','Age','Gender','Occupation','Rating']]
```

In [16]:

```
Master_data.shape
```

Out[16]:

```
(1000209, 7)
```

In [17]:

```
Master_data.head()
```

Out[17]:

	Movie ID	Title	User ID	Age	Gender	Occupation	Rating
0	1	Toy Story (1995)	1	1	F	10	5
1	48	Pocahontas (1995)	1	1	F	10	5
2	150	Apollo 13 (1995)	1	1	F	10	5
3	260	Star Wars: Episode IV - A New Hope (1977)	1	1	F	10	4
4	527	Schindler's List (1993)	1	1	F	10	5

In [18]:

```
len(Master_data['Age'])
```

Out[18]:

```
1000209
```

In [19]:

```
import matplotlib.pyplot as plt
```

In [20]:

```
Master_data['Age'].value_counts()
```

Out[20]:

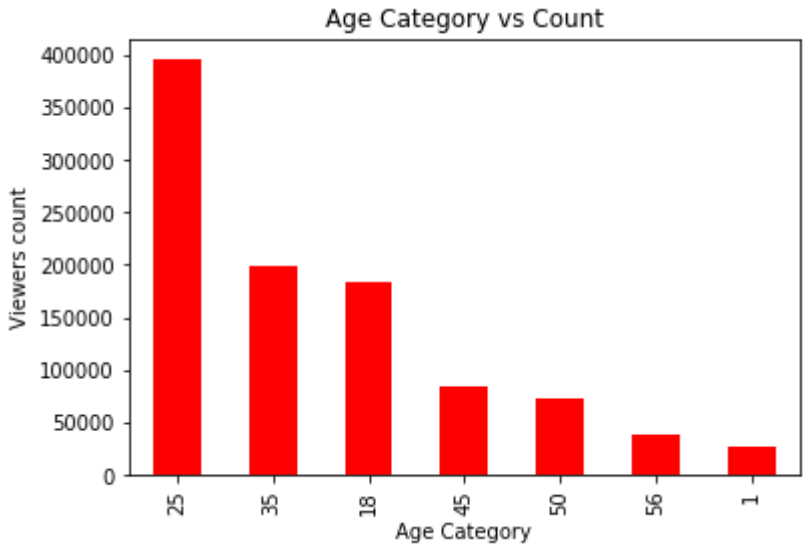
```
25    395556
35    199003
18    183536
45     83633
50     72490
56     38780
1      27211
Name: Age, dtype: int64
```

In [21]:

```
Master_data['Age'].value_counts().plot(kind="bar",color="red")
plt.xlabel("Age Category")
plt.ylabel("Viewers count")
plt.title("Age Category vs Count")
```

Out[21]:

```
Text(0.5, 1.0, 'Age Category vs Count')
```



Conclusion : The age category of 25 people have been watching movies very frequently

In [22]:

```
print(Master_data['Age'].mean())
print(Master_data['Age'].median())
print(Master_data['Age'].mode())
```

```
29.73831369243828
25.0
0    25
dtype: int64
```

In [23]:

```
import seaborn as sns
```

In [24]:

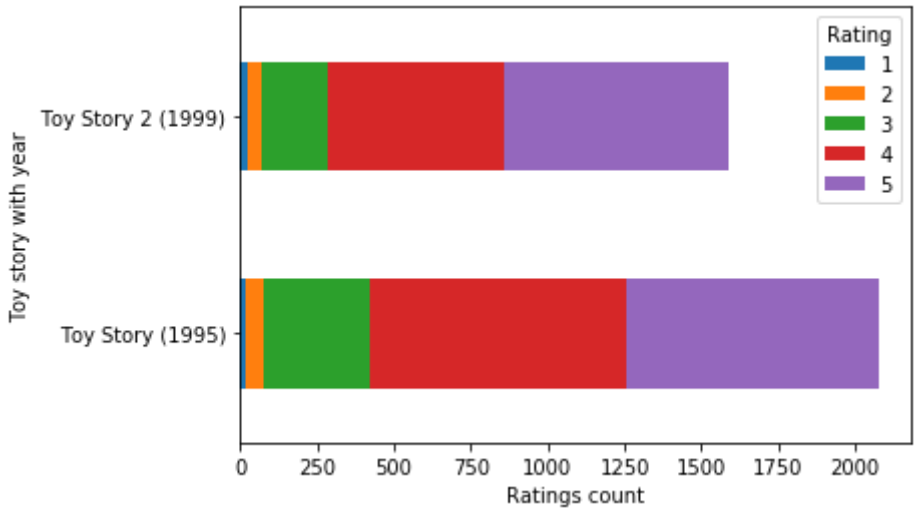
```
toy_rating=Master_data[Master_data['Title'].str.contains("Toy Story")][['Title','Rating']]
toy_rating.head()
```

Out[24]:

	Title	Rating
0	Toy Story (1995)	5
50	Toy Story 2 (1999)	4
53	Toy Story (1995)	4
124	Toy Story (1995)	4
263	Toy Story (1995)	5

In [25]:

```
toy_rating.groupby(["Title","Rating"]).size().unstack().plot(kind='barh',stacked=True,legend=True)
plt.xlabel("Ratings count")
plt.ylabel("Toy story with year")
plt.show()
```



Toy story avg rating

In [26]:

```
x=pd.DataFrame(toy_rating.groupby(["Title","Rating"]).size())
```

In [27]:

```
x
```

Out[27]:

		0
Title	**Rating**	
Toy Story (1995)	**1**	16
	2	61
	3	345
	4	835
	5	820
Toy Story 2 (1999)	**1**	25
	2	44
	3	214
	4	578
	5	724

Top 25 movies by viewership rating

In [28]:

```
df1=Master_data.groupby('Title').size().sort_values(ascending=False)
```

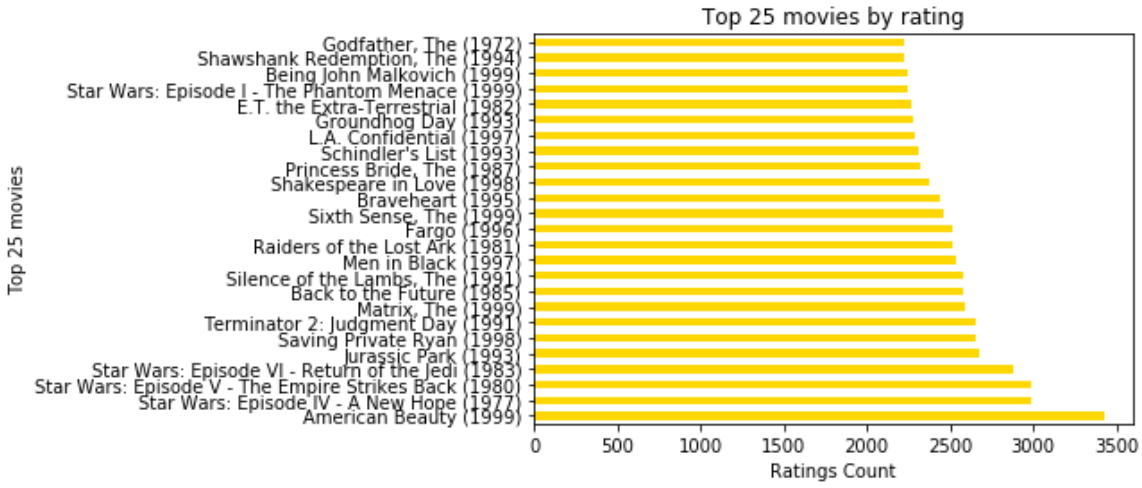
In [29]:

```
print(len(df1))
X=df1[:25]
```

```
3706
```

In [30]:

```
X.plot(kind='barh',color="gold")
plt.xlabel("Ratings Count")
plt.ylabel("Top 25 movies")
plt.title("Top 25 movies by rating")
plt.show()
```



In [31]:

```
movie_dataset.head()
```

Out[31]:

	Movie ID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

Ratings of all the movies for particular user with userid of 2696

In [32]:

```
Master_data[Master_data['User ID']==2696][['User ID','Title','Rating']]
```

Out[32]:

	User ID	Title	Rating
991035	2696	Client, The (1994)	3
991036	2696	Lone Star (1996)	5
991037	2696	Basic Instinct (1992)	4
991038	2696	E.T. the Extra-Terrestrial (1982)	3
991039	2696	Shining, The (1980)	4
991040	2696	Back to the Future (1985)	2
991041	2696	Cop Land (1997)	3
991042	2696	L.A. Confidential (1997)	4
991043	2696	Game, The (1997)	4
991044	2696	I Know What You Did Last Summer (1997)	2
991045	2696	Devil's Advocate, The (1997)	4
991046	2696	Midnight in the Garden of Good and Evil (1997)	4
991047	2696	Palmetto (1998)	4
991048	2696	Wild Things (1998)	4
991049	2696	Perfect Murder, A (1998)	4
991050	2696	I Still Know What You Did Last Summer (1998)	2
991051	2696	Psycho (1998)	4
991052	2696	Lake Placid (1999)	1
991053	2696	Talented Mr. Ripley, The (1999)	4
991054	2696	JFK (1991)	1

In [33]:

```
uniqGenre_list=[]
for i in movie_dataset['Genres']:
    j=i.split("|")
    for k in j :
        if  k not in uniqGenre_list:
            uniqGenre_list.append(k)
```

In [34]:

```
len(uniqGenre_list)
```

Out[34]:

```
18
```

In [ ]:

```
```

One hot encoding the Genre column

In [35]:

```
movie_dataset.head()
```

Out[35]:

	Movie ID	Title	Genres
0	1	Toy Story (1995)	Animation|Children's|Comedy
1	2	Jumanji (1995)	Adventure|Children's|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama
4	5	Father of the Bride Part II (1995)	Comedy

In [36]:

```
encode_genre= movie_dataset['Genres'].str.get_dummies("|")
```

In [37]:

```
encode_genre
```

Out[37]:

	Action	Adventure	Animation	Children's	Comedy	Crime	Documentary	Drama	Fant
0	0	0	1	1	1	0	0	0	
1	0	1	0	1	0	0	0	0	
2	0	0	0	0	1	0	0	0	
3	0	0	0	0	1	0	0	1	
4	0	0	0	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	
3878	0	0	0	0	1	0	0	0	
3879	0	0	0	0	0	0	0	1	
3880	0	0	0	0	0	0	0	1	
3881	0	0	0	0	0	0	0	1	
3882	0	0	0	0	0	0	0	1	

3883 rows × 18 columns

In [38]:

```
encode_genre.shape
```

Out[38]:

```
(3883, 18)
```

In [39]:

```
pd.concat([movie_dataset,encode_genre],axis=1)
```

Out[39]:

	Movie ID	Title	Genres	Action	Adventure	Animation	Children's
0	1	Toy Story (1995)	Animation|Children's|Comedy	0	0	1	1
1	2	Jumanji (1995)	Adventure|Children's|Fantasy	0	1	0	1
2	3	Grumpier Old Men (1995)	Comedy|Romance	0	0	0	0
3	4	Waiting to Exhale (1995)	Comedy|Drama	0	0	0	0
4	5	Father of the Bride Part II (1995)	Comedy	0	0	0	0
...	...	...	...	...	...	...	...
3878	3948	Meet the Parents (2000)	Comedy	0	0	0	0
3879	3949	Requiem for a Dream (2000)	Drama	0	0	0	0
3880	3950	Tigerland (2000)	Drama	0	0	0	0
3881	3951	Two Family House (2000)	Drama	0	0	0	0
3882	3952	Contender, The (2000)	Drama|Thriller	0	0	0	0

3883 rows × 21 columns

In [40]:

```
Master_data
```

Out[40]:

	Movie ID	Title	User ID	Age	Gender	Occupation	Rating
0	1	Toy Story (1995)	1	1	F	10	5
1	48	Pocahontas (1995)	1	1	F	10	5
2	150	Apollo 13 (1995)	1	1	F	10	5
3	260	Star Wars: Episode IV - A New Hope (1977)	1	1	F	10	4
4	527	Schindler's List (1993)	1	1	F	10	5
...	...	...	...	...	...	...	...
1000204	3513	Rules of Engagement (2000)	5727	25	M	4	4
1000205	3535	American Psycho (2000)	5727	25	M	4	2
1000206	3536	Keeping the Faith (2000)	5727	25	M	4	5
1000207	3555	U-571 (2000)	5727	25	M	4	3
1000208	3578	Gladiator (2000)	5727	25	M	4	5

1000209 rows × 7 columns

MULTIPLE LINEAR REGRESSION

In [41]:

```
X=Master_data.iloc[:,3:6].values
Y=Master_data.iloc[:,-1].values
```

In [42]:

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
X
```

Out[42]:

```
array([[1, 'F', 10],
       [1, 'F', 10],
       [1, 'F', 10],
       ...,
       [25, 'M', 4],
       [25, 'M', 4],
       [25, 'M', 4]], dtype=object)
```

In [43]:

```
Y
```

Out[43]:

```
array([5, 5, 5, ..., 5, 3, 5], dtype=int64)
```

In [44]:

```
ct=ColumnTransformer(transformers=[('encoder',OneHotEncoder(),[1])],remainder="passthro
ugh")
```

In [45]:

```
X=ct.fit_transform(X)
```

In [46]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,Y,test_size=.3,random_state=0)
```

In [47]:

```
from sklearn.linear_model import LinearRegression
```

In [48]:

```
model=LinearRegression()
```

In [49]:

```
model.fit(X_train,y_train)
```

Out[49]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=F
alse)
```

In [74]:

```
pred=model.predict(X_test)
```

In [51]:

```
model.coef_
```

Out[51]:

```
array([ 0.02686047, -0.02686047,  0.00522271,  0.00084735])
```

In [52]:

```
model.intercept_
```

Out[52]:

```
3.4331345478465978
```

In [75]:

```
pred[10:15]
```

Out[75]:

```
array([3.59330572, 3.53684185, 3.60601597, 3.50367228, 3.50367228])
```

In [54]:

```
y_test[10:15]
```

Out[54]:

```
array([5, 3, 5, 4, 2], dtype=int64)
```

In [55]:

```
from sklearn.metrics import mean_squared_error
```

In [56]:

```
mse=mean_squared_error(y_test,pred)
```

In [57]:

```
import math
rmse=math.sqrt(mse)
```

In [58]:

```
print(mse)
print(rmse)
```

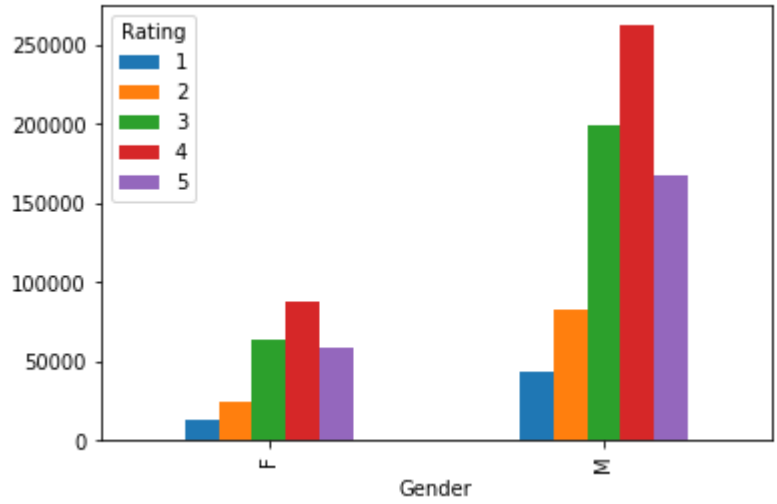
```
1.2440123449009362
1.1153530135795287
```

DATA VISUALIZATION

How rating is related to gender

In [59]:

```
Master_data.groupby(["Gender","Rating"]).size().unstack().plot(kind='bar',stacked=False
,legend=True)
plt.show()
```

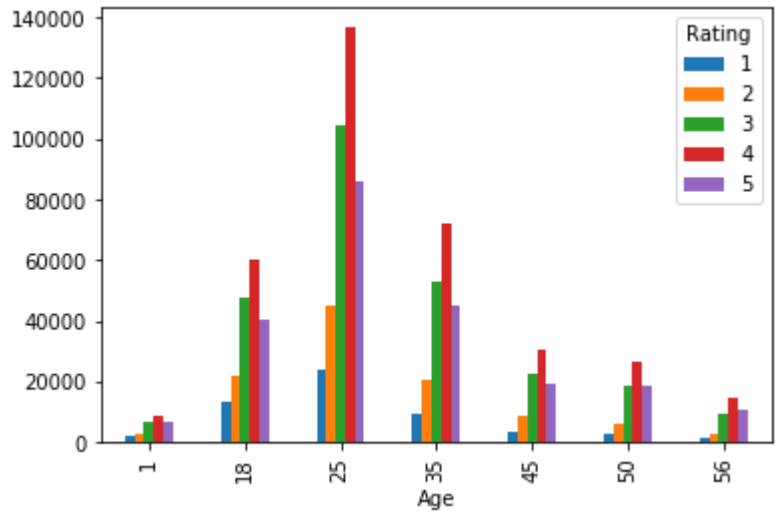


From the above graph, more males watch the movie than females

Ratings relation with Age category

In [60]:

```
Master_data.groupby(["Age","Rating"]).size().unstack().plot(kind='bar',stacked=False,le
gend=True)
plt.show()
```



In [61]:

```
Master_data.columns
```

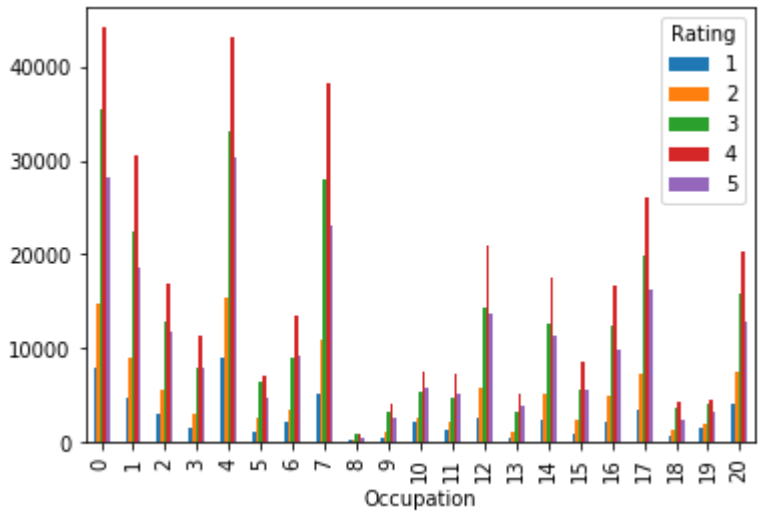
Out[61]:

```
Index(['Movie ID', 'Title', 'User ID', 'Age', 'Gender', 'Occupation',
       'Rating'],
      dtype='object')
```

Ratings relation with Occupation

In [90]:

```
Master_data.groupby(["Occupation","Rating"]).size().unstack().plot(kind='bar',stacked=False,legend=True)
plt.show()
```



USING DECISION TREE REGRESSION

In [68]:

```
from sklearn.tree import DecisionTreeRegressor
```

In [69]:

```
decision_tree=DecisionTreeRegressor(random_state=1)
```

In [70]:

```
decision_tree.fit(X_train,y_train)
```

Out[70]:

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=1, splitter='best')
```

In [73]:

```
pred2=decision_tree.predict(X_test)
```

In [76]:

```
pred2[1:10]
```

Out[76]:

```
array([3.5020694 , 3.47646909, 3.64970269, 3.53896053, 3.53665511,
       3.36253041, 3.54411331, 3.707854  , 3.45449355])
```

In [77]:

```
pred[1:10]
```

Out[77]:

```
array([3.55739322, 3.51214578, 3.67334109, 3.53853655, 3.50367228,
       3.60750981, 3.5956469 , 3.64722753, 3.53684185])
```

In [85]:

```
mse2=mean_squared_error(y_test,pred2)
```

In [88]:

```
rmse2=math.sqrt(mse2)
```

In [89]:

```
rmse2
```

Out[89]:

```
1.1089701089725315
```

In [ ]:

```

```