In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
mydata=pd.read_csv('C:Downloads/TaxiFare.csv')
```

In [3]:

```
mydata.head()
```

Out[3]:

	unique_id	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_o
0	26:21.0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-7
1	52:16.0	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-7
2	35:00.0	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-7
3	30:42.0	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-7
4	51:00.0	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-7

In [4]:

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   unique_id            50000 non-null  object 
 1   amount               50000 non-null  float64
 2   date_time_of_pickup  50000 non-null  object 
 3   longitude_of_pickup  50000 non-null  float64
 4   latitude_of_pickup   50000 non-null  float64
 5   longitude_of_dropoff 50000 non-null  float64
 6   latitude_of_dropoff  50000 non-null  float64
 7   no_of_passenger      50000 non-null  int64  
dtypes: float64(5), int64(1), object(2)
memory usage: 3.1+ MB
```

In [5]:

```
mydata.shape
```

Out[5]:

```
(50000, 8)
```

In [6]:

```
mydata.isnull().sum()
```

Out[6]:

```
unique_id               0
amount                  0
date_time_of_pickup     0
longitude_of_pickup     0
latitude_of_pickup      0
longitude_of_dropoff    0
latitude_of_dropoff     0
no_of_passenger         0
dtype: int64
```

In [7]:

```
mydata.describe()
```

Out[7]:

	amount	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_d
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.0
mean	11.364171	-72.509756	39.933759	-72.504616	39.9
std	9.685557	10.393860	6.224857	10.407570	6.0
min	-5.000000	-75.423848	-74.006893	-84.654241	-74.0
25%	6.000000	-73.992062	40.734880	-73.991152	40.7
50%	8.500000	-73.981840	40.752678	-73.980082	40.7
75%	12.500000	-73.967148	40.767360	-73.963584	40.7
max	200.000000	40.783472	401.083332	40.851027	43.4

In [8]:

```
from sklearn.preprocessing import LabelEncoder
```

In [9]:

```
LE=LabelEncoder()
```

In [10]:

```
mydata.unique_id=LE.fit_transform(mydata.unique_id)
mydata.date_time_of_pickup=LE.fit_transform(mydata.date_time_of_pickup)
```

In [11]:

```
mydata
```

Out[11]:

	unique_id	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitu
0	1579	4.5	3408	-73.844311	40.721319	
1	3133	16.9	7748	-74.016048	40.711303	
2	2097	5.7	20152	-73.982738	40.761270	
3	1839	7.7	25488	-73.987130	40.733143	
4	3057	5.3	8973	-73.968095	40.768008	
...	...	...	...	...	...	
49995	1513	15.0	34451	-73.999973	40.748531	
49996	1157	7.5	49424	-73.984756	40.768211	
49997	3177	6.9	15821	-74.002698	40.739428	
49998	540	4.5	29672	-73.946062	40.777567	
49999	794	10.9	7927	-73.932603	40.763805	

50000 rows × 8 columns

In [12]:

```
mydata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 8 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   unique_id            50000 non-null  int32  
 1   amount               50000 non-null  float64
 2   date_time_of_pickup  50000 non-null  int32  
 3   longitude_of_pickup  50000 non-null  float64
 4   latitude_of_pickup   50000 non-null  float64
 5   longitude_of_dropoff 50000 non-null  float64
 6   latitude_of_dropoff  50000 non-null  float64
 7   no_of_passenger      50000 non-null  int64  
dtypes: float64(5), int32(2), int64(1)
memory usage: 2.7 MB
```

In [13]:

```
mydata.describe()
```

Out[13]:

	unique_id	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup
count	50000.00000	50000.000000	50000.000000	50000.000000	50000.000000
mean	1793.93710	11.364171	24770.967840	-72.509756	39.933759
std	1037.39357	9.685557	14295.321372	10.393860	6.224857
min	0.00000	-5.000000	0.000000	-75.423848	-74.006893
25%	900.00000	6.000000	12388.750000	-73.992062	40.734880
50%	1798.00000	8.500000	24774.500000	-73.981840	40.752678
75%	2697.00000	12.500000	37144.250000	-73.967148	40.767360
max	3596.00000	200.000000	49554.000000	40.783472	401.083332

In [14]:

```
mydata_corr=mydata.corr()
```

In [15]:

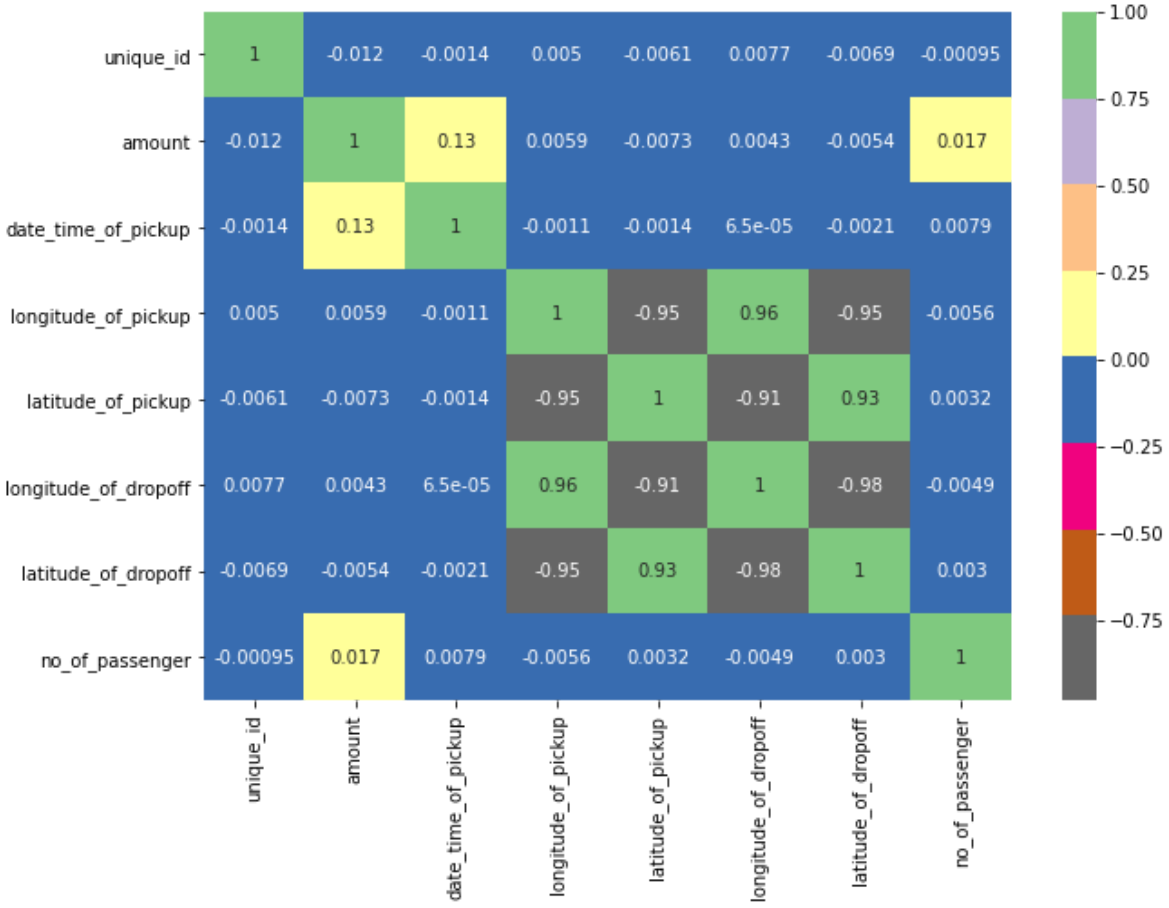
```
mydata_corr
```

Out[15]:

	unique_id	amount	date_time_of_pickup	longitude_of_pickup	latitude_of
unique_id	1.000000	-0.012349	-0.001434	0.005004	-(
amount	-0.012349	1.000000	0.125868	0.005944	-(
date_time_of_pickup	-0.001434	0.125868	1.000000	-0.001135	-(
longitude_of_pickup	0.005004	0.005944	-0.001135	1.000000	-(
latitude_of_pickup	-0.006088	-0.007338	-0.001375	-0.950588	1
longitude_of_dropoff	0.007732	0.004286	0.000065	0.956131	-(
latitude_of_dropoff	-0.006911	-0.005442	-0.002147	-0.946968	(
no_of_passenger	-0.000947	0.016583	0.007934	-0.005604	(

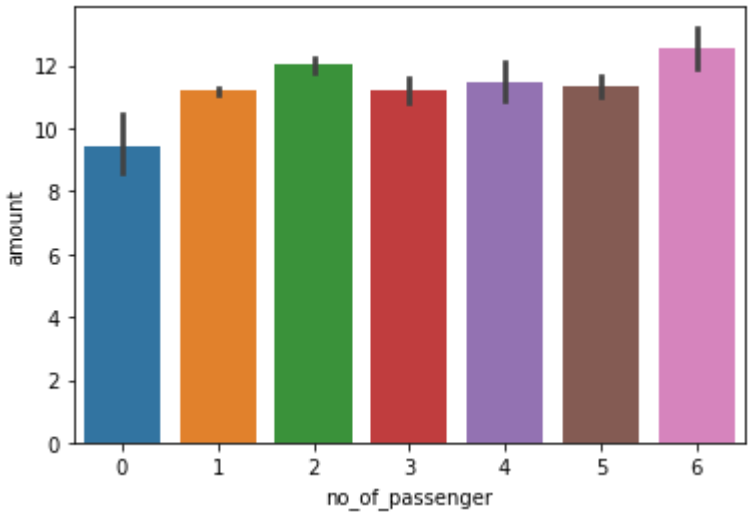
In [16]:

```
plt.figure(figsize=(10,7))
sns.heatmap(mydata_corr,annot=True,cmap='Accent_r');
plt.show()
```



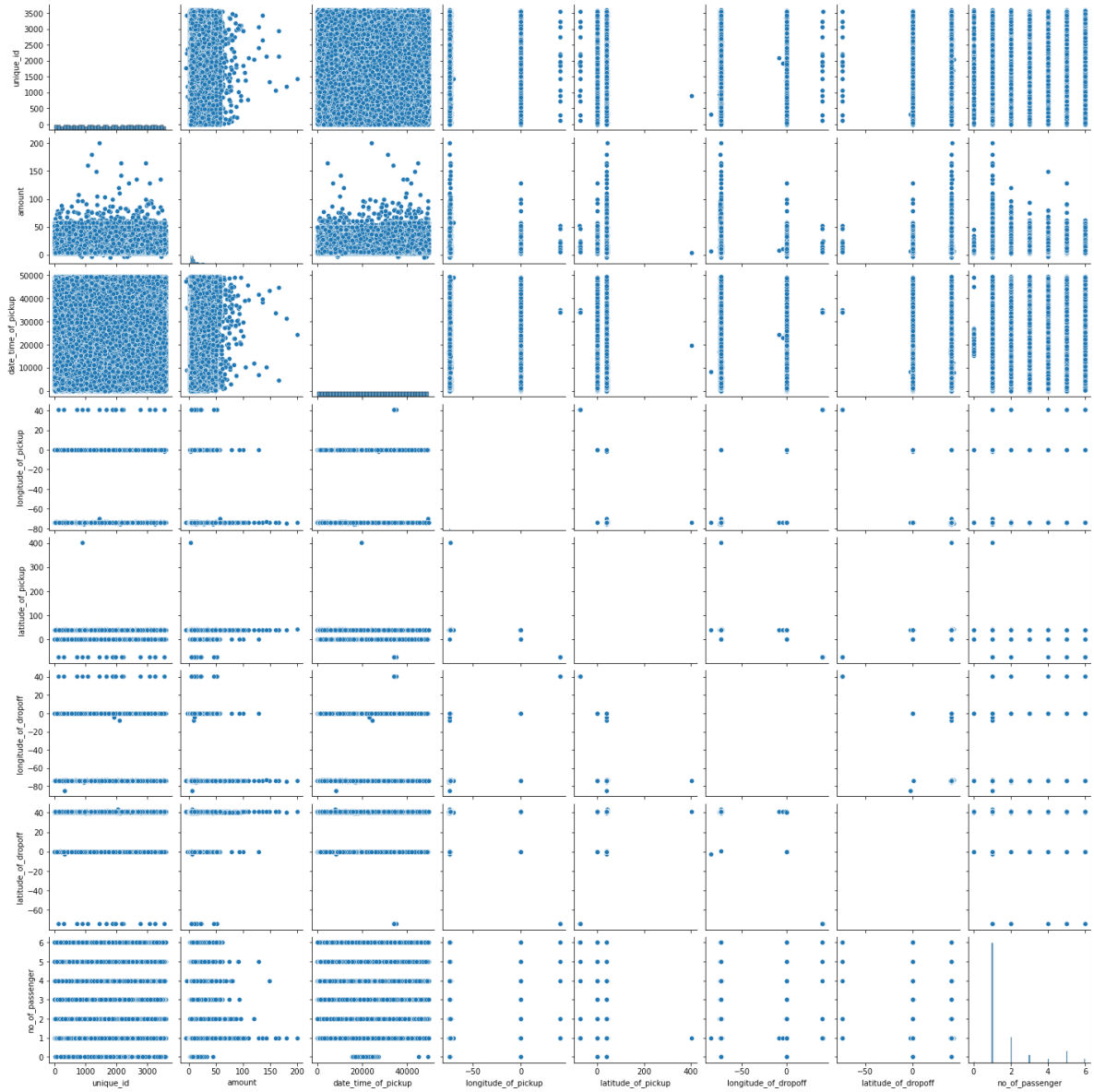
In [17]:

```
sns.barplot(x='no_of_passenger',y='amount',data=mydata);
```



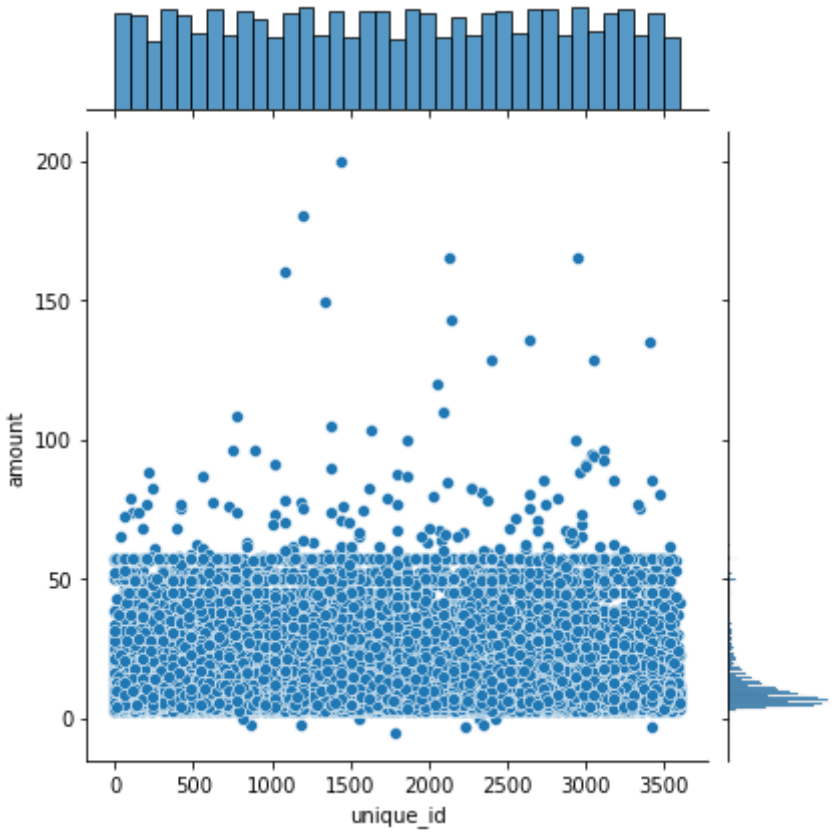
In [18]:

```
sns.pairplot(mydata);
```



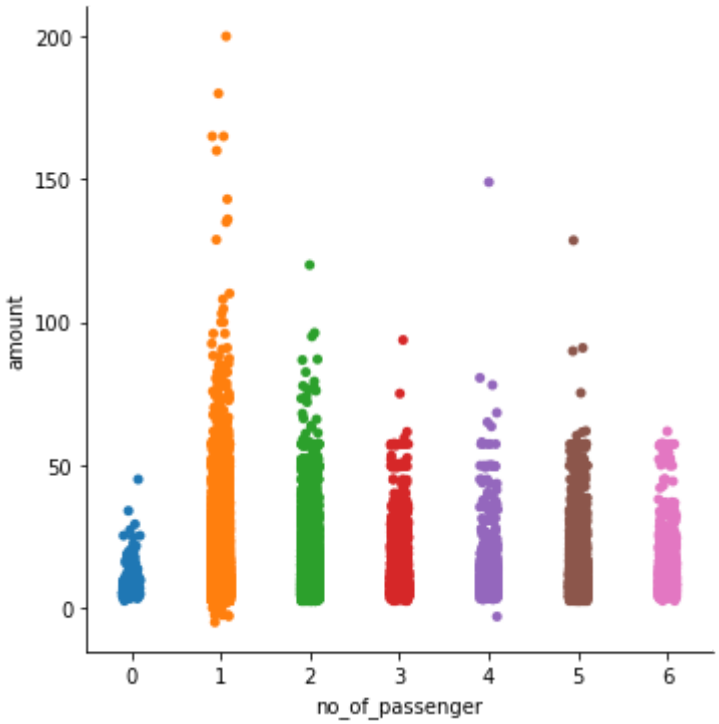
In [19]:

```
sns.jointplot(x='unique_id',y='amount',data=mydata);
```



In [20]:

```
sns.catplot(x='no_of_passenger',y='amount',data=mydata);
```



Separating dep and ind variables.

In [27]:

```
y_dep= mydata.amount
```

In [28]:

```
y_dep
```

Out[28]:

```
0         4.5
1        16.9
2         5.7
3         7.7
4         5.3
        ...
49995    15.0
49996     7.5
49997     6.9
49998     4.5
49999    10.9
Name: amount, Length: 50000, dtype: float64
```

In [23]:

```
x_ind=mydata.drop("amount",axis=1)
```

In [24]:

```
x_ind
```

Out[24]:

	unique_id	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_drc
0	1579	3408	-73.844311	40.721319	-73.84
1	3133	7748	-74.016048	40.711303	-73.97
2	2097	20152	-73.982738	40.761270	-73.99
3	1839	25488	-73.987130	40.733143	-73.99
4	3057	8973	-73.968095	40.768008	-73.95
...	...	...	...	...	
49995	1513	34451	-73.999973	40.748531	-74.01
49996	1157	49424	-73.984756	40.768211	-73.98
49997	3177	15821	-74.002698	40.739428	-73.99
49998	540	29672	-73.946062	40.777567	-73.95
49999	794	7927	-73.932603	40.763805	-73.93

50000 rows × 7 columns

Random Forest

In [25]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

In [29]:

```
x_train, x_test, y_train, y_test=train_test_split(x_ind,y_dep,train_size=0.8,random_state=2
```

In [30]:

```
model_rf=RandomForestRegressor(random_state=2)
```

In [31]:

```
model_rf=model_rf.fit(x_train,y_train)
```

In [32]:

```
y_pred=model_rf.predict(x_test)
```

In [33]:

```
y_pred
```

Out[33]:

```
array([ 8.93 , 15.09 ,  7.655, ..., 13.682, 15.258,  7.325])
```

In [38]:

```
final_comp=pd.DataFrame({"Actual":y_test, "Machine_pred":y_pred})
```

In [39]:

```
final_comp
```

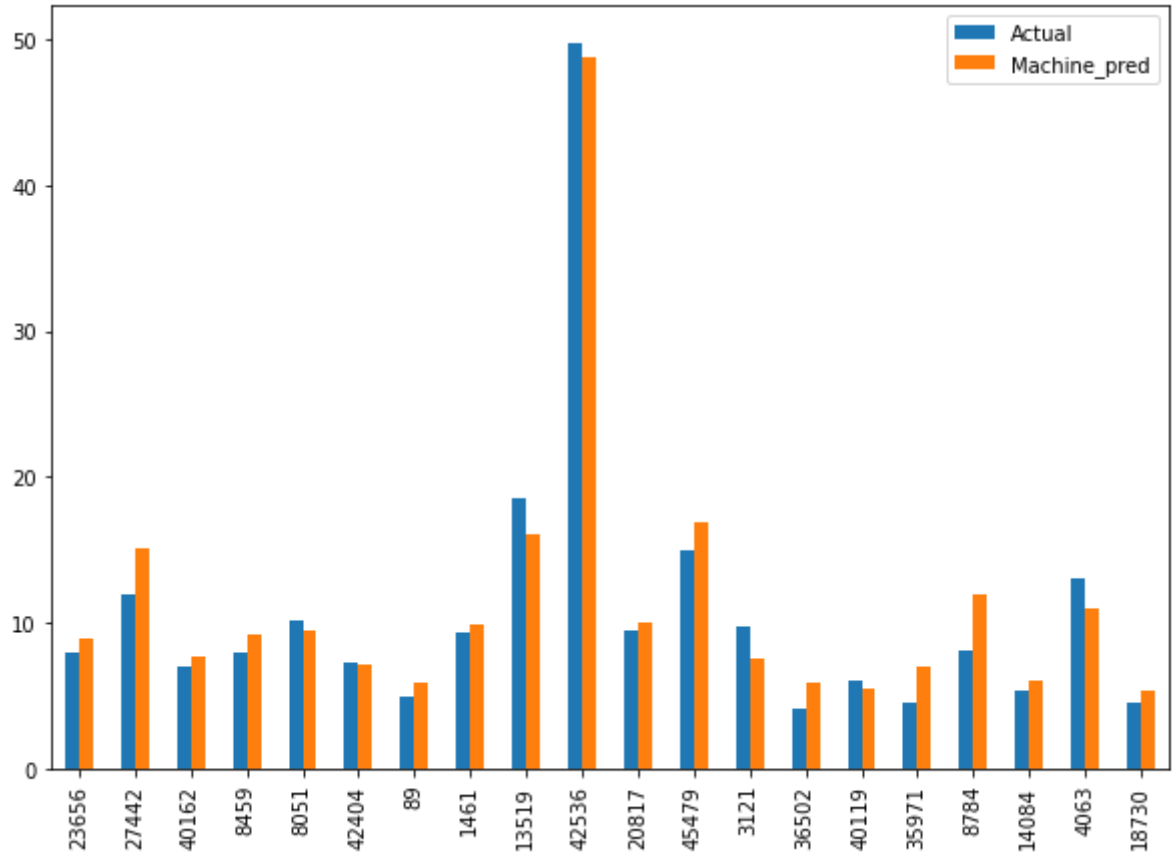
Out[39]:

	Actual	Machine_pred
23656	8.0	8.930
27442	12.0	15.090
40162	7.0	7.655
8459	8.0	9.255
8051	10.1	9.413
...	...	...
44231	15.0	14.643
18034	9.5	9.410
33856	12.5	13.682
15906	12.5	15.258
40899	7.5	7.325

10000 rows × 2 columns

In [40]:

```
comp_g=final_comp.head(20)
comp_g.plot(kind='bar',figsize=(10,7))
plt.show()
```



HyperParameter Tuning

In [42]:

```
from sklearn.model_selection import RandomizedSearchCV
```

In [43]:

```
parameters={"n_estimators":(200,300,400,500), "max_features":("auto","sqrt","log2"),
            "min_samples_split":(2,4,6),"random_state":(0,1,2,3)}
```

In [44]:

```
RF=RandomizedSearchCV(RandomForestRegressor(),param_distributions=parameters,cv=5)
```

In [45]:

```
RF.fit(x_train,y_train)
```

Out[45]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(),
                   param_distributions={'max_features': ('auto', 'sqrt',
                                                         'log2'),
                                        'min_samples_split': (2, 4, 6),
                                        'n_estimators': (200, 300, 400, 50
0),
                                        'random_state': (0, 1, 2, 3)})
```

In [46]:

```
RF.best_estimator_
```

Out[46]:

```
RandomForestRegressor(max_features='sqrt', n_estimators=500, random_state=3)
```

In [47]:

```
model_hp=RandomForestRegressor(max_features='sqrt', n_estimators=500, random_state=3)
```

In [48]:

```
model_hp=model_hp.fit(x_train,y_train)
```

In [49]:

```
y_pred_hp=model_hp.predict(x_test)
```

In [50]:

```
y_pred_hp
```

Out[50]:

```
array([ 7.8466 , 14.48732,  8.91132, ..., 14.57538, 16.50536,  7.796  ])
```

In [51]:

```
f_comp=pd.DataFrame({"Actual":y_test, "Machine_pred":y_pred_hp})
```

In [52]:

```
f_comp
```

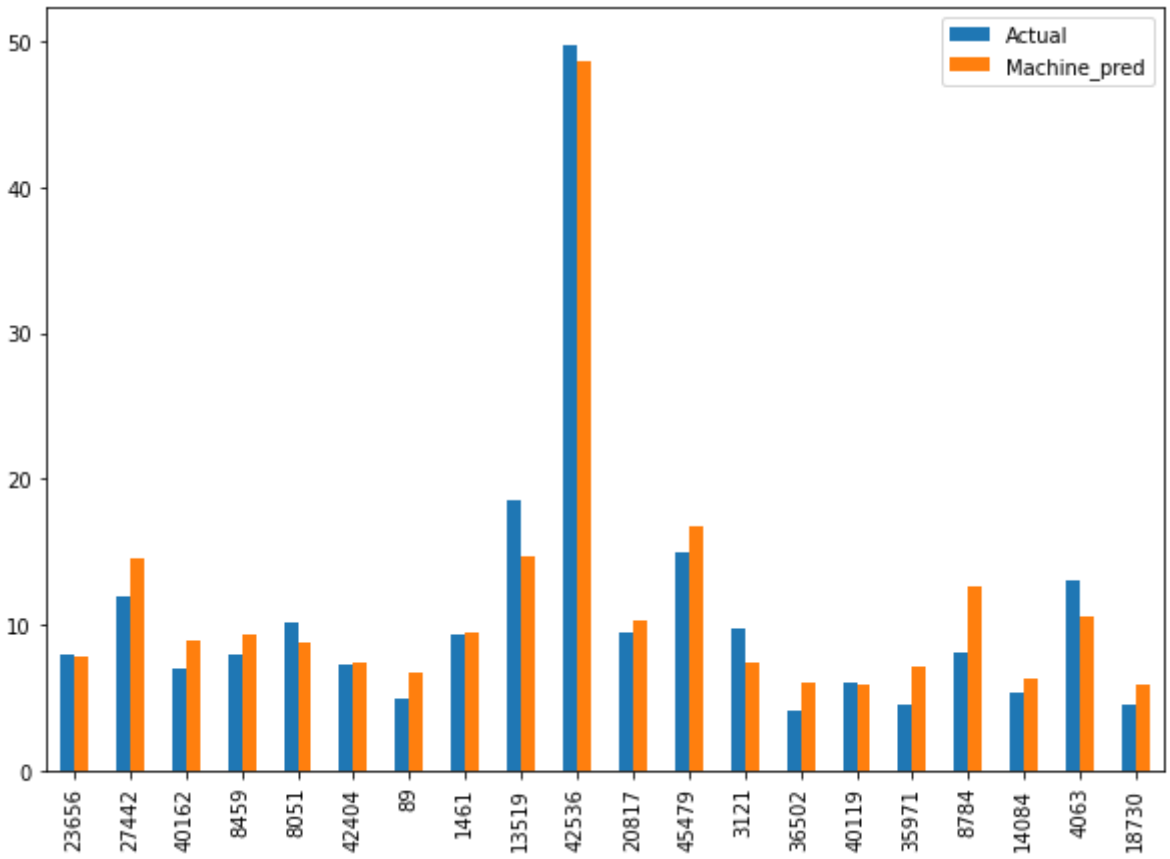
Out[52]:

	Actual	Machine_pred
23656	8.0	7.84660
27442	12.0	14.48732
40162	7.0	8.91132
8459	8.0	9.35400
8051	10.1	8.77400
...	...	...
44231	15.0	15.55424
18034	9.5	9.07786
33856	12.5	14.57538
15906	12.5	16.50536
40899	7.5	7.79600

10000 rows × 2 columns

In [53]:

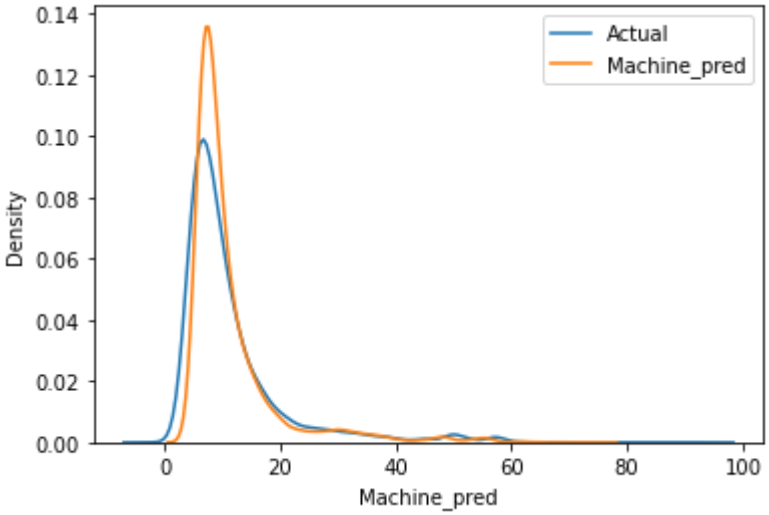
```
comp_g=f_comp.head(20)
comp_g.plot(kind='bar',figsize=(10,7))
plt.show()
```



In [54]:

```
sns.distplot(f_comp['Actual'], hist = False)
sns.distplot(f_comp['Machine_pred'], hist = False)
plt.legend(['Actual','Machine_pred'])
plt.show()
```

```
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\Users\aneef\anaconda3\lib\site-packages\seaborn\distributions.py:2557: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level
function with similar flexibility) or `kdeplot` (an axes-level function for
kernel density plots).
  warnings.warn(msg, FutureWarning)
```



In [55]:

```
model_hp.score(x_test,y_test)*100
```

Out[55]:

```
78.47108344089692
```

In [ ]:

```

```