In [2]:

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

In [3]:

```python
mydata=pd.read_csv("TaxiFare.csv")
```

In [4]:

```python
mydata.head(5)
```

Out[4]:

	unique_id	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passe
0	26:21.0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	
1	52:16.0	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	
2	35:00.0	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	
3	30:42.0	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	
4	51:00.0	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	

In [5]:
```
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 [6]:
```
mydata.describe()
```

Out[6]:

	amount	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
count	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000	50000.000000
mean	11.364171	-72.509756	39.933759	-72.504616	39.926251	1.667840
std	9.685557	10.393860	6.224857	10.407570	6.014737	1.289195
min	-5.000000	-75.423848	-74.006893	-84.654241	-74.006377	0.000000
25%	6.000000	-73.992062	40.734880	-73.991152	40.734372	1.000000
50%	8.500000	-73.981840	40.752678	-73.980082	40.753372	1.000000
75%	12.500000	-73.967148	40.767360	-73.963584	40.768167	2.000000
max	200.000000	40.783472	401.083332	40.851027	43.415190	6.000000

In [7]:
```
mydata.shape
```

Out[7]:
```
(50000, 8)
```

In [8]:
```
mydata.isnull().sum()
```

Out[8]:
```
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 [9]:
```
mydata.rename(columns={'amount' : 'target'}, inplace=True)
```

In [10]:
```
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   target                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 [11]:
```
from sklearn.preprocessing import LabelEncoder
```

In [12]:
```
LE=LabelEncoder()
```

In [13]:

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

In [14]:

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

In [15]:

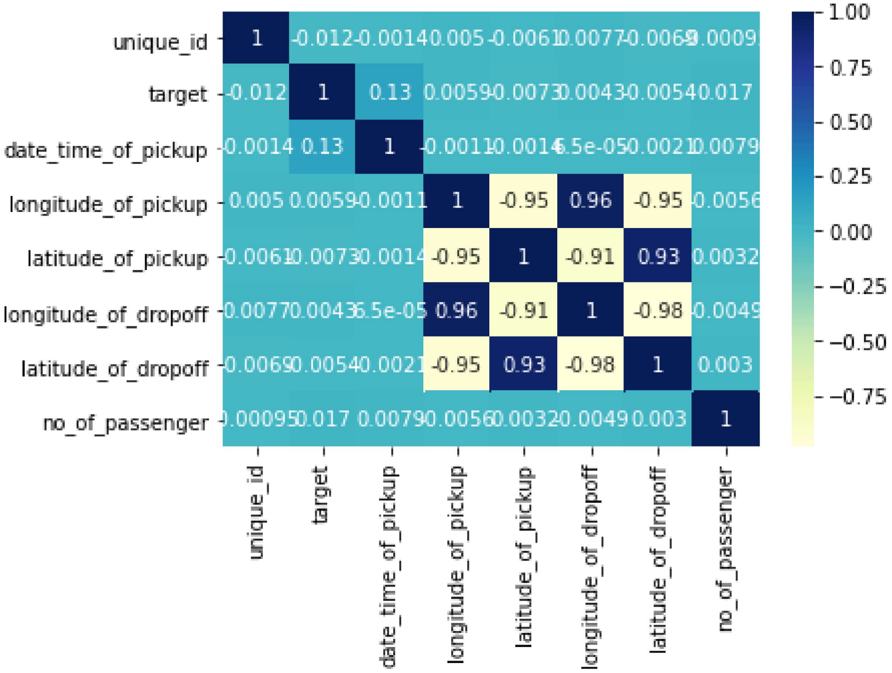
```
mydata_corr
```

Out[15]:

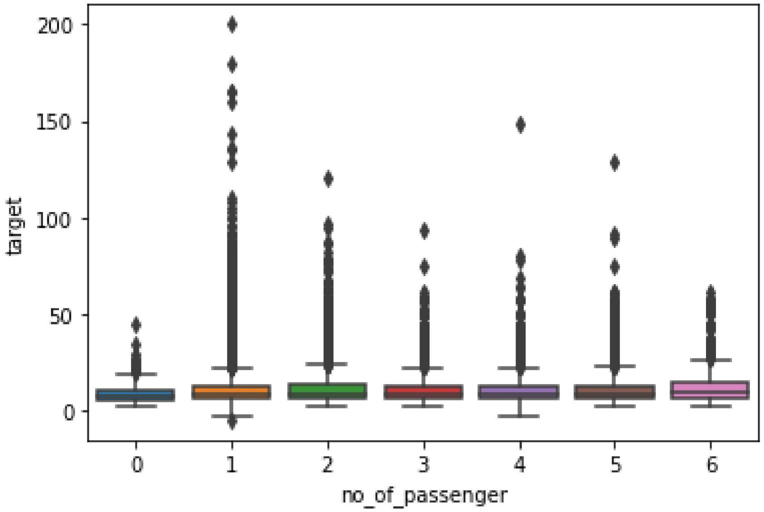
	unique_id	target	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_
unique_id	1.000000	-0.012349	-0.001434	0.005004	-0.006088	0.007732	-0.
target	-0.012349	1.000000	0.125868	0.005944	-0.007338	0.004286	-0.
date_time_of_pickup	-0.001434	0.125868	1.000000	-0.001135	-0.001375	0.000065	-0.
longitude_of_pickup	0.005004	0.005944	-0.001135	1.000000	-0.950588	0.956131	-0.
latitude_of_pickup	-0.006088	-0.007338	-0.001375	-0.950588	1.000000	-0.911123	0.
longitude_of_dropoff	0.007732	0.004286	0.000065	0.956131	-0.911123	1.000000	-0.
latitude_of_dropoff	-0.006911	-0.005442	-0.002147	-0.946968	0.928189	-0.982117	1.
no_of_passenger	-0.000947	0.016583	0.007934	-0.005604	0.003237	-0.004936	0.

In [16]:

```
sns.heatmap(mydata_corr,annot=True,cmap='YlGnBu');
```



In [17]:
```
sns.boxplot(x='no_of_passenger',y='target',data=mydata);
```



In [18]:
```
y_dep=mydata.target
```

In [19]:
```
y_dep
```

Out[19]:
```
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: target, Length: 50000, dtype: float64
```

In [20]:
```
x_ind=mydata.drop("target",axis=1)
```

In [21]:
```
x_ind
```

Out[21]:

	unique_id	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	1579	3408	-73.844311	40.721319	-73.841610	40.712278	1
1	3133	7748	-74.016048	40.711303	-73.979268	40.782004	1
2	2097	20152	-73.982738	40.761270	-73.991242	40.750562	2
3	1839	25488	-73.987130	40.733143	-73.991567	40.758092	1
4	3057	8973	-73.968095	40.768008	-73.956655	40.783762	1
...	...	...	...	...	...	...	...
49995	1513	34451	-73.999973	40.748531	-74.016899	40.705993	1
49996	1157	49424	-73.984756	40.768211	-73.987366	40.760597	1
49997	3177	15821	-74.002698	40.739428	-73.998108	40.759483	1
49998	540	29672	-73.946062	40.777567	-73.953450	40.779687	2
49999	794	7927	-73.932603	40.763805	-73.932603	40.763805	1

50000 rows × 7 columns

random forest

In [22]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
```

In [23]:
```
x_train,x_test,y_train,y_test=train_test_split(x_ind,y_dep,train_size=0.8,random_state=2)
```

In [24]:
```
model_rf=RandomForestRegressor(random_state=2)
```

In [25]:
```
model_rf=model_rf.fit(x_train,y_train)
```

In [26]:
```
y_pred=model_rf.predict(x_test)
```

In [27]:
```
y_pred
```

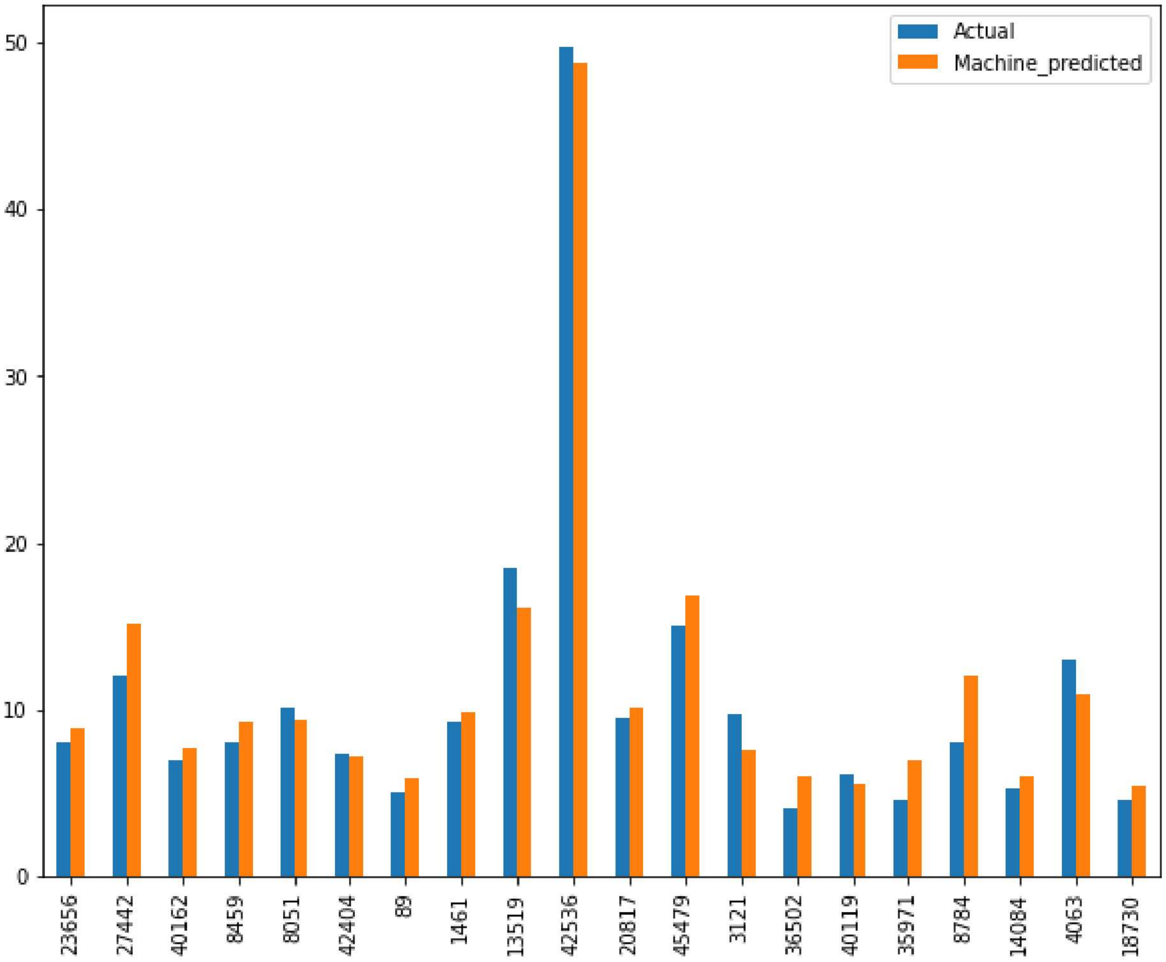
Out[27]:
```
array([ 8.93 , 15.09 ,  7.655, ..., 13.682, 15.258,  7.325])
```

In [28]:
```
final_comp=pd.DataFrame({"Actual" : y_test, "Machine_predicted" : y_pred})
final_comp.head()
```

Out[28]:

	Actual	Machine_predicted
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

In [29]:
```
com_g=final_comp.head(20)
com_g.plot(kind='bar',figsize=(10,8))
plt.show()
```



hyper parameter tuning

In [30]:

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

In [31]:

```
parameters={'n_estimators':(200,300,400,500),'max_features':('auto','sqrt','log2'),
           'min_samples_split':(2,4,6),'random_state':(0,1,2,3,4,5)}
parameters
```

Out[31]:

```
{'n_estimators': (200, 300, 400, 500),
 'max_features': ('auto', 'sqrt', 'log2'),
 'min_samples_split': (2, 4, 6),
 'random_state': (0, 1, 2, 3, 4, 5)}
```

In [32]:

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

In [33]:

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

Out[33]:

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(),
                   param_distributions={'max_features': ('auto', 'sqrt',
                                                         'log2'),
                                        'min_samples_split': (2, 4, 6),
                                        'n_estimators': (200, 300, 400, 500),
                                        'random_state': (0, 1, 2, 3, 4, 5)})
```

In [34]:
```
RF.best_estimator_
```

Out[34]:
```
RandomForestRegressor(max_features='log2', min_samples_split=4,
                      n_estimators=500, random_state=5)
```

In [35]:
```
model_hp=RandomForestRegressor(max_features='log2', min_samples_split=4,
                      n_estimators=500, random_state=5)
```

In [36]:
```
model_hp=model_hp.fit(x_train,y_train)
```

In [37]:
```
y_pred_hp=model_hp.predict(x_test)
```

In [38]:
```
y_pred_hp
```

Out[38]:
```
array([ 7.70883056, 14.74769357,  8.94780294, ..., 14.64888637,
       15.66868202,  7.78873532])
```

In [39]:
```
f_comp1=pd.DataFrame({'Actual' : y_test, 'Machine_predicted' : y_pred_hp})
```

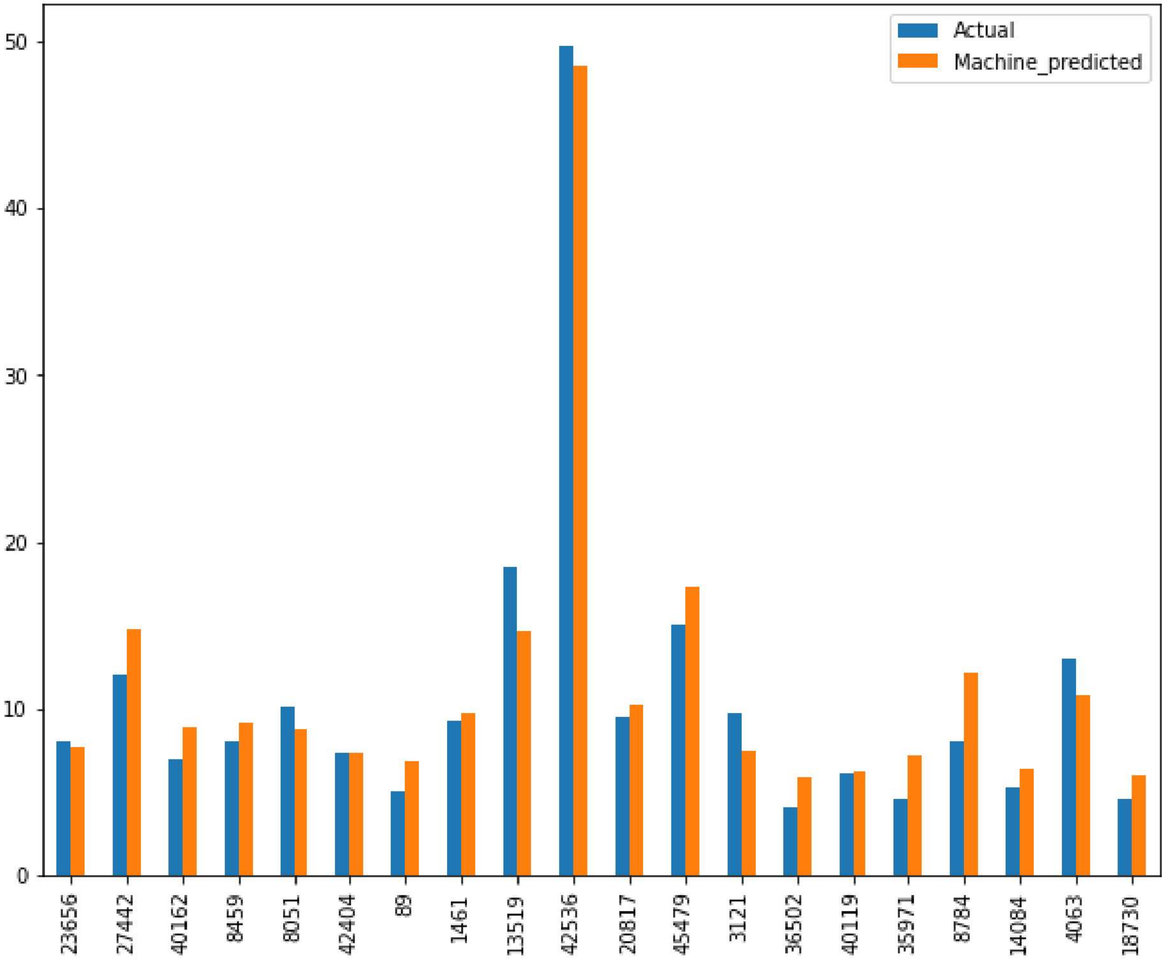
In [40]:
```
f_comp1.head()
```

Out[40]:

	Actual	Machine_predicted
23656	8.0	7.708831
27442	12.0	14.747694
40162	7.0	8.947803
8459	8.0	9.182084
8051	10.1	8.752762

In [43]:

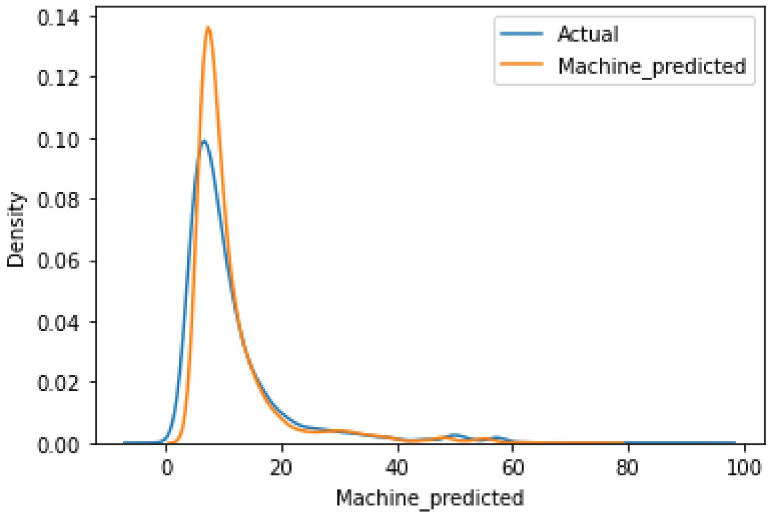
```
com_g= f_comp1.head(20)
com_g.plot(kind='bar', figsize=(10,8))
plt.show()
```



In [44]:

```
sns.distplot(f_comp1['Actual'], hist=False)
sns.distplot(f_comp1['Machine_predicted'], hist=False)
plt.legend(['Actual', 'Machine_predicted'])
plt.show()
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depre
cated function and will be removed in a future version. Please adapt your code to use either `displot` (a figu
re-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a depre
cated function and will be removed in a future version. Please adapt your code to use either `displot` (a figu
re-level function with similar flexibility) or `kdeplot` (an axes-level function for kernel density plots).
  warnings.warn(msg, FutureWarning)
```



In [45]:

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

Out[45]:

```
78.66965704902545
```

In [ ]: