In [1]:
```
import pandas as pd
```

In [2]:
```
taxi=pd.read_excel("TaxiFare.csv.xlsx")
```

In [3]:
```
taxi
```

Out[3]:

	unique_id	amount	date_time_of_pickup	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	26:21.0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1
1	52:16.0	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1
2	35:00.0	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2
3	30:42.0	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1
4	51:00.0	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1
...	...	...	...	...	...	...	...	...
49995	25:15.0	15.0	2013-06-12 23:25:15 UTC	-73.999973	40.748531	-74.016899	40.705993	1
49996	19:18.0	7.5	2015-06-22 17:19:18 UTC	-73.984756	40.768211	-73.987366	40.760597	1
49997	53:00.0	6.9	2011-01-30 04:53:00 UTC	-74.002698	40.739428	-73.998108	40.759483	1
49998	09:00.0	4.5	2012-11-06 07:09:00 UTC	-73.946062	40.777567	-73.953450	40.779687	2
49999	13:14.0	10.9	2010-01-13 08:13:14 UTC	-73.932603	40.763805	-73.932603	40.763805	1

50000 rows × 8 columns

In [4]:
```
taxi.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]:
```
taxidata=taxi.drop("date_time_of_pickup",axis=1)
taxidata
```

Out[5]:

	unique_id	amount	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
0	26:21.0	4.5	-73.844311	40.721319	-73.841610	40.712278	1
1	52:16.0	16.9	-74.016048	40.711303	-73.979268	40.782004	1
2	35:00.0	5.7	-73.982738	40.761270	-73.991242	40.750562	2
3	30:42.0	7.7	-73.987130	40.733143	-73.991567	40.758092	1
4	51:00.0	5.3	-73.968095	40.768008	-73.956655	40.783762	1
...	...	...	...	...	...	...	...
49995	25:15.0	15.0	-73.999973	40.748531	-74.016899	40.705993	1
49996	19:18.0	7.5	-73.984756	40.768211	-73.987366	40.760597	1
49997	53:00.0	6.9	-74.002698	40.739428	-73.998108	40.759483	1
49998	09:00.0	4.5	-73.946062	40.777567	-73.953450	40.779687	2
49999	13:14.0	10.9	-73.932603	40.763805	-73.932603	40.763805	1

50000 rows × 7 columns

In [6]:
```
import sklearn
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
```

In [7]:
```
taxidata["unique_id"]=LE.fit_transform(taxidata["unique_id"])
```

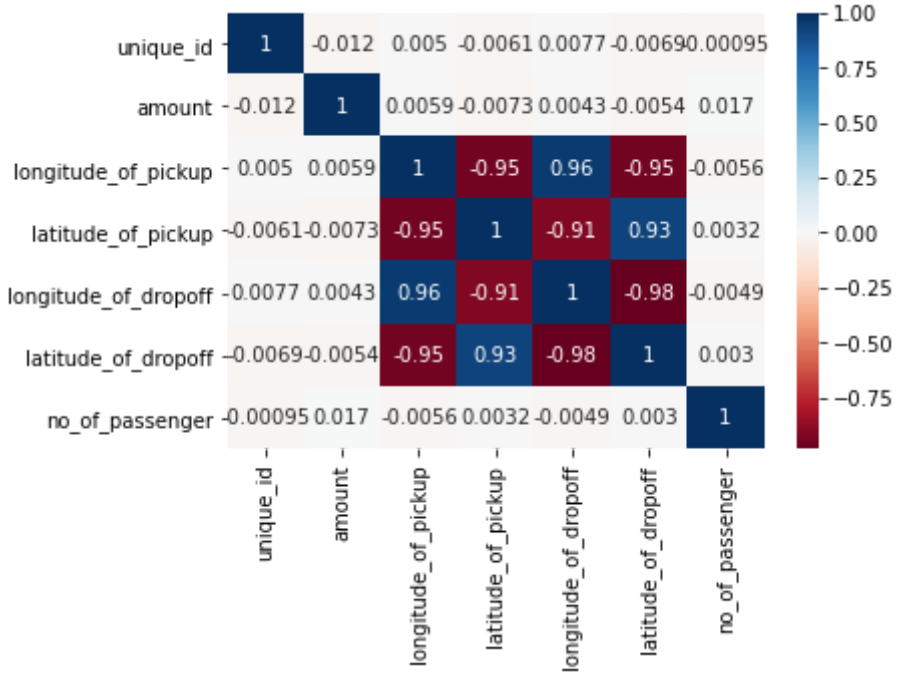
In [8]:
```
taxidata_corr=taxidata.corr()
taxidata_corr
```

Out[8]:

	unique_id	amount	longitude_of_pickup	latitude_of_pickup	longitude_of_dropoff	latitude_of_dropoff	no_of_passenger
unique_id	1.000000	-0.012349	0.005004	-0.006088	0.007732	-0.006911	-0.000947
amount	-0.012349	1.000000	0.005944	-0.007338	0.004286	-0.005442	0.016583
longitude_of_pickup	0.005004	0.005944	1.000000	-0.950588	0.956131	-0.946968	-0.005604
latitude_of_pickup	-0.006088	-0.007338	-0.950588	1.000000	-0.911123	0.928189	0.003237
longitude_of_dropoff	0.007732	0.004286	0.956131	-0.911123	1.000000	-0.982117	-0.004936
latitude_of_dropoff	-0.006911	-0.005442	-0.946968	0.928189	-0.982117	1.000000	0.002958
no_of_passenger	-0.000947	0.016583	-0.005604	0.003237	-0.004936	0.002958	1.000000

In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [10]:
```
sns.heatmap(taxidata_corr,annot=True,cmap="RdBu");
```



In [11]:
```
x_ind=taxidata.drop("amount",axis=1)
y_dep=taxidata.amount
```

In [12]:
```
from sklearn import model_selection
from sklearn.model_selection import train_test_split
```

In [13]:
```
x_train, x_test, y_train, y_test = train_test_split(x_ind, y_dep, test_size=0.2, random_state=2)
```

In [14]:
```
from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor(random_state=2)
```

In [15]:
```
model.fit(x_train,y_train)
```

Out[15]:
```
RandomForestRegressor(random_state=2)
```

In [16]:
```
y_pred=model.predict(x_test)
y_pred
```

Out[16]:
```
array([ 7.043 , 16.04  ,  7.368 , ..., 12.321 , 15.3377,  6.977 ])
```

In [17]:
```
model.score(x_test,y_test)
```

Out[17]:
```
0.7561585754382035
```

In [18]:
```
#hypertuning
```

In [19]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [20]:
```
parameters={"n_estimators":(200,300,400,500,600,7000),
            "max_features":("auto","sqrt","log2"),"min_samples_split":(2,4,6),"random_state":(0,1,2,3)}
```

In [21]:
```
RF=RandomizedSearchCV(RandomForestRegressor(),param_distributions=parameters,cv=2)
```

In [22]:
```
import warnings
warnings.filterwarnings("ignore")
```

In [23]:
```
RF.fit(x_train,y_train)
```

Out[23]:
```
RandomizedSearchCV(cv=2, estimator=RandomForestRegressor(),
                   param_distributions={'max_features': ('auto', 'sqrt',
                                                         'log2'),
                                       'min_samples_split': (2, 4, 6),
                                       'n_estimators': (200, 300, 400, 500,
                                                        600, 7000),
                                       'random_state': (0, 1, 2, 3)})
```

In [24]:
```
RF.best_estimator_
```

Out[24]:
```
RandomForestRegressor(max_features='log2', min_samples_split=6,
                      n_estimators=400, random_state=1)
```

In [26]:
```
model_after_Ht=RandomForestRegressor(min_samples_split=6, max_features='log2',n_estimators=400,random_state=1)
```

In [27]:
```
model_after_Ht.fit(x_train,y_train)
```

Out[27]:
```
RandomForestRegressor(max_features='log2', min_samples_split=6,
                      n_estimators=400, random_state=1)
```

In [28]:
```
y_pred_ht=model_after_Ht.predict(x_test)
```

In [29]:
```
model_after_Ht.score(x_test,y_test)
```

Out[29]:
```
0.7708754837776604
```

the final accuracy after using hypertuning is 77% which is a good fit

In [ ]: