In [3]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pylab
from sklearn.model_selection import train_test_split
from sklearn import metrics

from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics
from sklearn import preprocessing
```

In [9]:
```
df = pd.read_csv("C:\\Users\\vaishnavi\\OneDrive\\Desktop\\uber.csv")
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype  
---  ------             --------------   -----  
 0   Unnamed: 0         200000 non-null  int64  
 1   key                200000 non-null  object 
 2   fare_amount        200000 non-null  float64
 3   pickup_datetime    200000 non-null  object 
 4   pickup_longitude   200000 non-null  float64
 5   pickup_latitude    200000 non-null  float64
 6   dropoff_longitude  199999 non-null  float64
 7   dropoff_latitude   199999 non-null  float64
 8   passenger_count    200000 non-null  int64  
dtypes: float64(5), int64(2), object(2)
memory usage: 13.7+ MB
```

In [11]:
```
df.head()
```

Out[11]:

	Unnamed: 0	key	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_l
0	24238194	52:06.0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-7
1	27835199	04:56.0	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-7
2	44984355	45:00.0	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-7
3	25894730	22:21.0	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-7
4	17610152	47:00.0	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-7

In [12]:
```
df.describe()
```

Out[12]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff
count	2.000000e+05	200000.000000	200000.000000	200000.000000	199999.000000	19999
mean	2.771250e+07	11.359955	-72.527638	39.935885	-72.525292	3
std	1.601382e+07	9.901776	11.437787	7.720539	13.117408	
min	1.000000e+00	-52.000000	-1340.648410	-74.015515	-3356.666300	-88
25%	1.382535e+07	6.000000	-73.992065	40.734796	-73.991407	4
50%	2.774550e+07	8.500000	-73.981823	40.752592	-73.980093	4
75%	4.155530e+07	12.500000	-73.967153	40.767158	-73.963659	4
max	5.542357e+07	499.000000	57.418457	1644.421482	1153.572603	87

In [13]:
```
df = df.drop(['Unnamed: 0', 'key'], axis=1)
```

In [14]:
```
df.isna().sum()
```

Out[14]:
```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    1
dropoff_latitude     1
passenger_count      0
dtype: int64
```

In [15]:
```
df.dropna(axis=0,inplace=True)
```

In [24]:
```
df.dtypes
```

Out[24]:
```
fare_amount          float64
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
second                 int64
minute                 int64
hour                   int64
day                    int64
month                  int64
year                   int64
dayofweek              int64
dtype: object
```

In [28]:
```
df.head()
```

Out[28]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_
0	7.5	-73.999817	40.738354	-73.999512	40.723217	
1	7.7	-73.994355	40.728225	-73.994710	40.750325	
2	12.9	-74.005043	40.740770	-73.962565	40.772647	
3	5.3	-73.976124	40.790844	-73.965316	40.803349	
4	16.0	-73.925023	40.744085	-73.973082	40.761247	

In [30]:
```
#haversive formula
```

In [31]:
```
incorrect_coordinates = df.loc[
    (df.pickup_latitude > 90) |(df.pickup_latitude < -90) |
    (df.dropoff_latitude > 90) |(df.dropoff_latitude < -90) |
    (df.pickup_longitude > 180) |(df.pickup_longitude < -180) |
    (df.dropoff_longitude > 90) |(df.dropoff_longitude < -90)
]

df.drop(incorrect_coordinates, inplace = True, errors = 'ignore')
```

In [32]:
```
def distance_transform(longitude1, latitude1, longitude2, latitude2):
    long1, lati1, long2, lati2 = map(np.radians, [longitude1, latitude1, longitude
    dist_long = long2 - long1
    dist_lati = lati2 - lati1
    a = np.sin(dist_lati/2)**2 + np.cos(lati1) * np.cos(lati2) * np.sin(dist_long/
    c = 2 * np.arcsin(np.sqrt(a)) * 6371
    # long1,lati1,long2,lati2 = longitude1[pos],latitude1[pos],longitude2[pos],lat
    # c = sqrt((long2 - long1) ** 2 + (lati2 - lati1) ** 2)asin

    return c
```

In [33]:
```
df['Distance'] = distance_transform(
    df['pickup_longitude'],
    df['pickup_latitude'],
    df['dropoff_longitude'],
    df['dropoff_latitude']
)
```

In [34]:
```
df.head()
```

Out[34]:

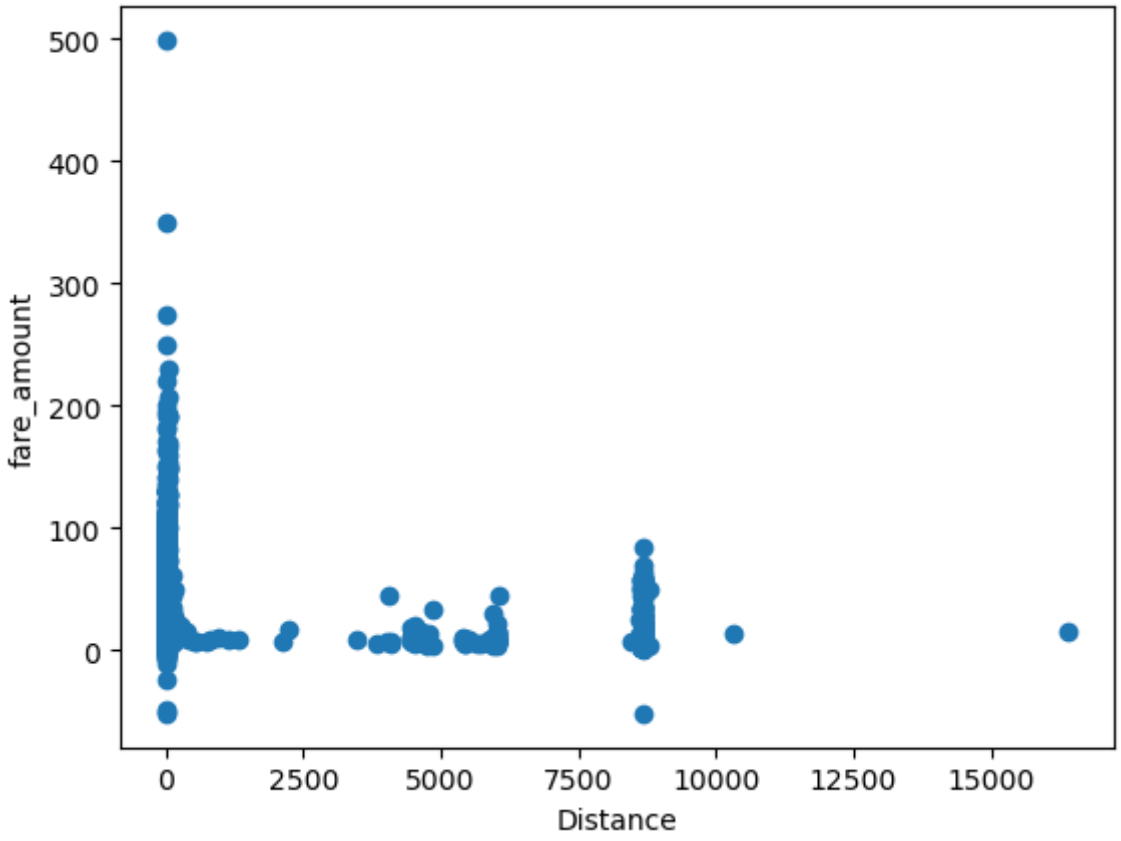
	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_
0	7.5	-73.999817	40.738354	-73.999512	40.723217	
1	7.7	-73.994355	40.728225	-73.994710	40.750325	
2	12.9	-74.005043	40.740770	-73.962565	40.772647	
3	5.3	-73.976124	40.790844	-73.965316	40.803349	
4	16.0	-73.925023	40.744085	-73.973082	40.761247	

In [35]:
```
#Outliers
#We can get rid of the trips with very large distances that are outliers as well as

plt.scatter(df['Distance'], df['fare_amount'])
```

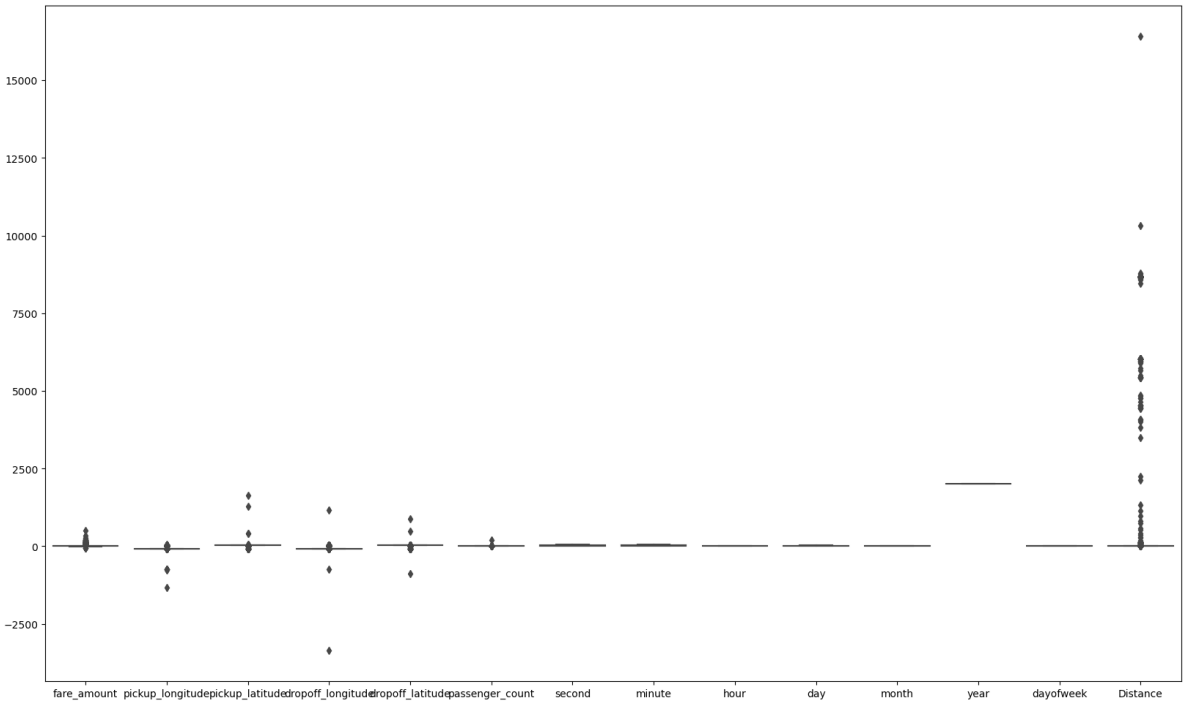
```
plt.xlabel("Distance")
plt.ylabel("fare_amount")
```

Out[35]:
```
Text(0, 0.5, 'fare_amount')
```



In [36]:
```
plt.figure(figsize=(20,12))
sns.boxplot(data = df)
```

Out[36]:
```
<Axes: >
```

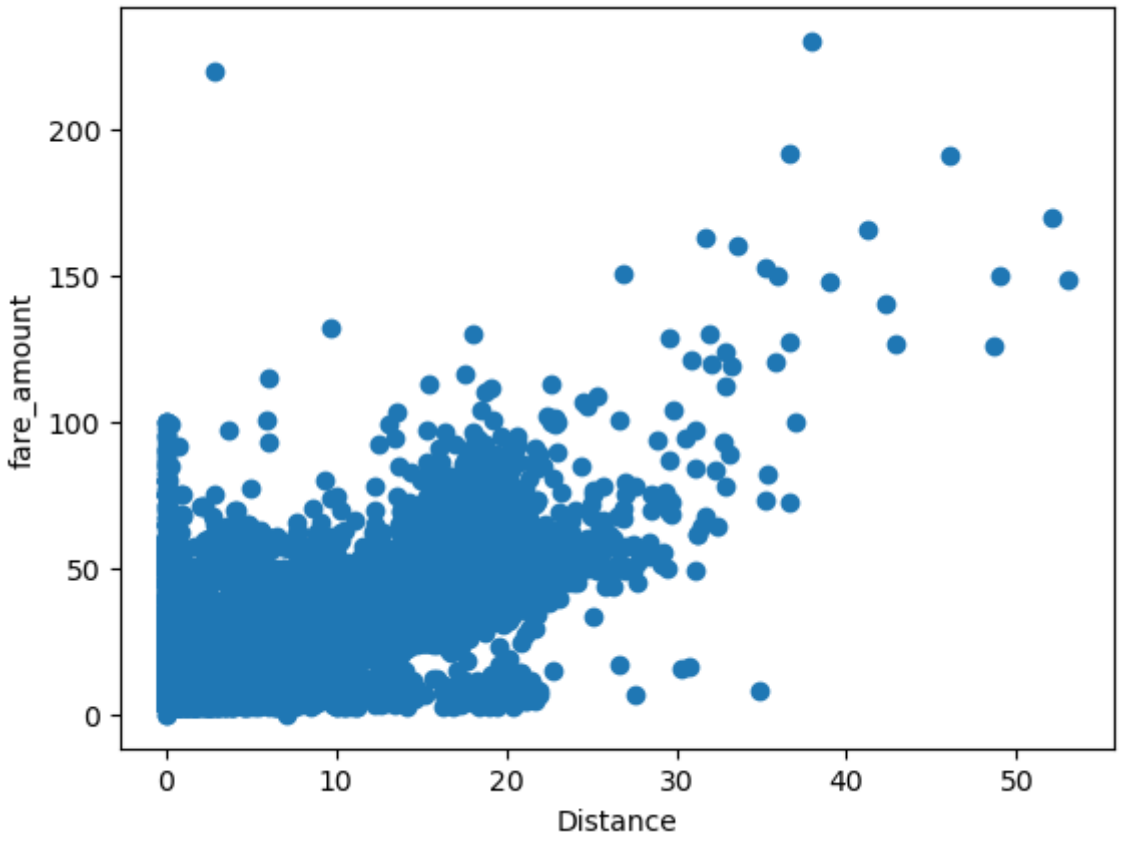


In [37]:
```
df.drop(df[df['Distance'] >= 60].index, inplace = True)
df.drop(df[df['fare_amount'] <= 0].index, inplace = True)

df.drop(df[(df['fare_amount']>100) & (df['Distance']<1)].index, inplace = True )
```

```
df.drop(df[(df['fare_amount']<100) & (df['Distance']>100)].index, inplace = True )
plt.scatter(df['Distance'], df['fare_amount'])
plt.xlabel("Distance")
plt.ylabel("fare_amount")
```

Out[37]:
```
Text(0, 0.5, 'fare_amount')
```



In [38]:
```
#Coorelation Matrix
#To find the two variables that have the most inter-dependence
```

In [39]:
```
corr = df.corr()

corr.style.background_gradient(cmap='BuGn')
```

Out[39]:

	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_lati
fare_amount	1.000000	0.005885	-0.006253	0.005501	-0.00
pickup_longitude	0.005885	1.000000	-0.973204	0.999992	-0.98
pickup_latitude	-0.006253	-0.973204	1.000000	-0.973206	0.99
dropoff_longitude	0.005501	0.999992	-0.973206	1.000000	-0.98
dropoff_latitude	-0.006142	-0.981941	0.991076	-0.981942	1.00
passenger_count	0.011693	-0.000649	-0.001190	-0.000650	-0.00
second	-0.000995	-0.014677	0.016809	-0.014638	0.01
minute	-0.007795	0.002796	-0.002295	0.002803	-0.00
hour	-0.020692	0.001547	-0.001823	0.001316	-0.00
day	0.001059	0.005300	-0.008901	0.005307	-0.00
month	0.023759	-0.002667	0.004098	-0.002656	0.00
year	0.121195	0.005907	-0.008466	0.005878	-0.00
dayofweek	0.006181	0.003006	-0.004787	0.003082	-0.00
Distance	0.857729	-0.117044	0.110843	-0.117282	0.10

In [41]:
```
#train and test set
X = df['Distance'].values.reshape(-1, 1)        #Independent Variable
y = df['fare_amount'].values.reshape(-1, 1)     #Dependent Variable
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
y_std = std.fit_transform(y)
print(y_std)

x_std = std.fit_transform(X)
print(x_std)
```

```
[[-0.39820843]
 [-0.37738556]
 [ 0.1640092 ]
 ...
 [ 2.03806797]
 [ 0.3305922 ]
 [ 0.28894645]]
[[-0.43819765]
 [-0.22258873]
 [ 0.49552213]
 ...
 [ 2.67145829]
 [ 0.07874892]
 [ 0.60173174]]
```

In [42]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_std, y_std, test_size=0.2, r
```

In [43]:
```
from sklearn.linear_model import LinearRegression
l_reg = LinearRegression()
l_reg.fit(X_train, y_train)

print("Training set score: {:.2f}".format(l_reg.score(X_train, y_train)))
print("Test set score: {:.7f}".format(l_reg.score(X_test, y_test)))
```

```
Training set score: 0.74
Test set score: 0.7340468
```

In [44]:
```
y_pred = l_reg.predict(X_test)

result = pd.DataFrame()
result[['Actual']] = y_test
result[['Predicted']] = y_pred

result.sample(10)
```

Out[44]:

	Actual	Predicted
33844	-0.502323	-0.437225
17130	-0.419031	-0.203444
2194	-0.294094	-0.305168
3565	-0.137922	-0.201751
30904	0.278535	0.365894
32825	-0.543969	-0.233684
31334	-0.627260	-0.393634
2240	0.247301	-0.150093
30833	0.018249	0.121950
13469	-0.189980	-0.049918

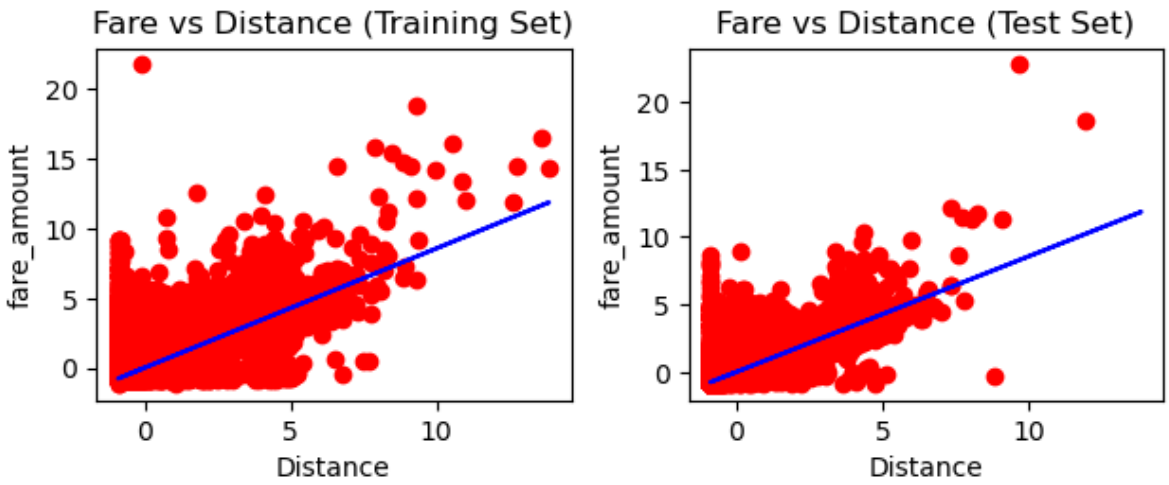
In [45]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Absolute % Error:', metrics.mean_absolute_percentage_error(y_test, y_pr
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pre
print('R Squared (R²):', np.sqrt(metrics.r2_score(y_test, y_pred)))
```

```
Mean Absolute Error: 0.2662129874635625
Mean Absolute % Error: 1.9830747643544173
Mean Squared Error: 0.27052435082793674
Root Mean Squared Error: 0.5201195543602805
R Squared (R²): 0.8567653082255872
```

In [46]:
```
plt.subplot(2, 2, 1)
plt.scatter(X_train, y_train, color = 'red')
plt.plot(X_train, l_reg.predict(X_train), color ="blue")
plt.title("Fare vs Distance (Training Set)")
plt.ylabel("fare_amount")
plt.xlabel("Distance")

plt.subplot(2, 2, 2)
plt.scatter(X_test, y_test, color = 'red')
plt.plot(X_train, l_reg.predict(X_train), color ="blue")
plt.ylabel("fare_amount")
plt.xlabel("Distance")
plt.title("Fare vs Distance (Test Set)")

plt.tight_layout()
plt.show()
```



In [47]:
```
cols = ['Model', 'RMSE', 'R-Squared']

# create a empty dataframe of the colums
# columns: specifies the columns to be selected
result_tabulation = pd.DataFrame(columns = cols)

# compile the required information
linreg_metrics = pd.DataFrame([[
     "Linear Regresion model",
     np.sqrt(metrics.mean_squared_error(y_test, y_pred)),
     np.sqrt(metrics.r2_score(y_test, y_pred))
]], columns = cols)

result_tabulation = pd.concat([result_tabulation, linreg_metrics], ignore_index=Tru

result_tabulation
```

Out[47]:

	Model	RMSE	R-Squared
0	Linear Regresion model	0.52012	0.856765

In [48]:
```
#RandomForestRegressor
```

In [49]:
```
rf_reg = RandomForestRegressor(n_estimators=100, random_state=10)

# fit the regressor with training dataset
rf_reg.fit(X_train, y_train)
```

```
C:\Users\vaishnavi\AppData\Local\Temp\ipykernel_18076\125726749.py:4: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please chang
e the shape of y to (n_samples,), for example using ravel().
  rf_reg.fit(X_train, y_train)
```

Out[49]:
```
RandomForestRegressor
RandomForestRegressor(random_state=10)
```

In [50]:
```
# predict the values on test dataset using predict()
y_pred_RF = rf_reg.predict(X_test)

result = pd.DataFrame()
result[['Actual']] = y_test
result['Predicted'] = y_pred_RF

result.sample(10)
```

Out[50]:

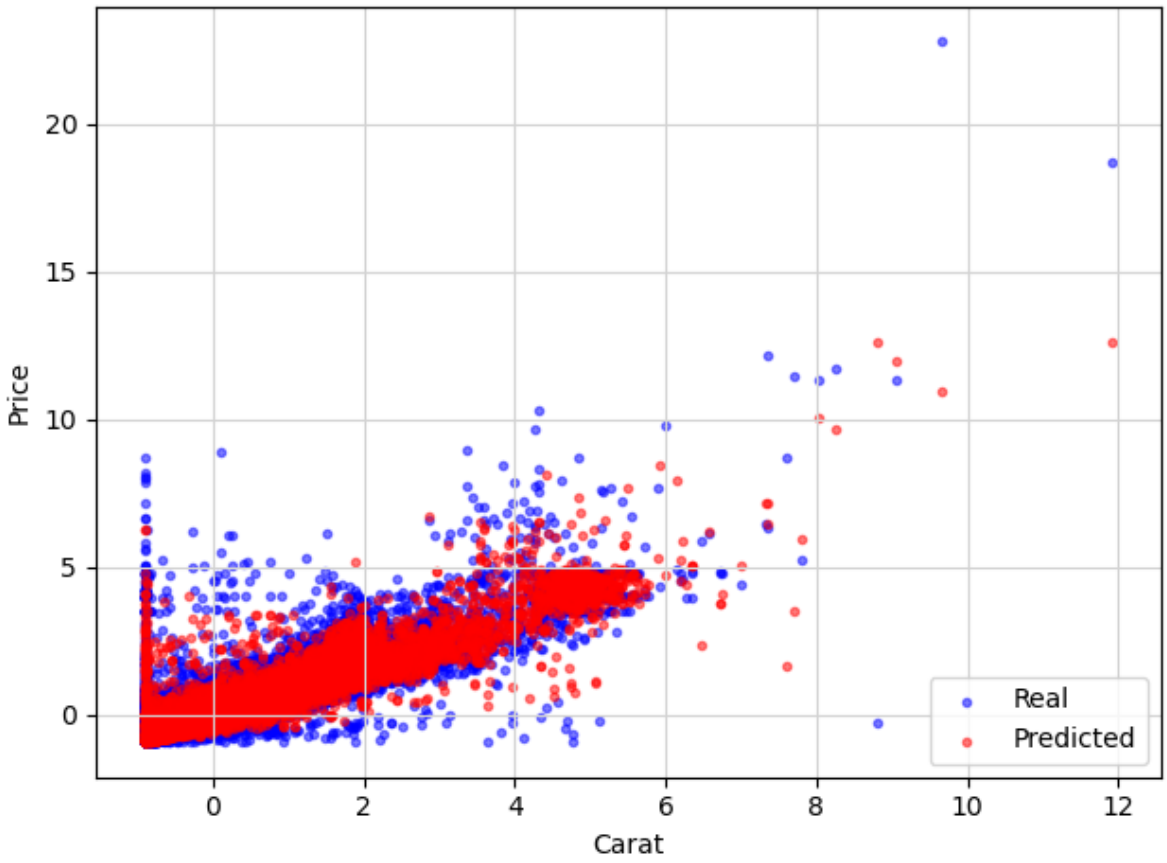
	Actual	Predicted
10195	1.614322	0.894684
34627	-0.502323	-0.703055
36684	0.018249	-0.173321
38479	-0.793843	-0.362914
26733	0.205655	0.449595
13348	-0.377386	-0.642669
39555	-0.502323	-0.437876
29023	-0.294094	-0.275458
30594	0.330592	-0.322309
2375	4.005830	4.539020

In [51]:
```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred_RF))
print('Mean Absolute % Error:', metrics.mean_absolute_percentage_error(y_test, y_pr
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred_RF))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pre
print('R Squared (R²):', np.sqrt(metrics.r2_score(y_test, y_pred_RF)))
```

```
Mean Absolute Error: 0.3077750884962444
Mean Absolute % Error: 2.162840407033828
Mean Squared Error: 0.33323701819885143
Root Mean Squared Error: 0.5772668518101931
R Squared (R²): 0.8199962218191474
```

In [52]:
```
# Build scatterplot
plt.scatter(X_test, y_test, c = 'b', alpha = 0.5, marker = '.', label = 'Real')
plt.scatter(X_test, y_pred_RF, c = 'r', alpha = 0.5, marker = '.', label = 'Predic
plt.xlabel('Carat')
plt.ylabel('Price')
plt.grid(color = '#D3D3D3', linestyle = 'solid')
plt.legend(loc = 'lower right')


plt.tight_layout()
plt.show()
```



In [53]:
```
# compile the required information
random_forest_metrics = pd.DataFrame([[
     "Random Forest Regressor model",
     np.sqrt(metrics.mean_squared_error(y_test, y_pred_RF)),
     np.sqrt(metrics.r2_score(y_test, y_pred_RF))
]], columns = cols)

result_tabulation = pd.concat([result_tabulation, random_forest_metrics], ignore_i

result_tabulation
```

Out[53]:

	Model	RMSE	R-Squared
0	Linear Regresion model	0.520120	0.856765
1	Random Forest Regressor model	0.577267	0.819996

In [ ]: