In [144]:

```
# Predict the price of the Uber ride from a given pickup point to the agreed drop-off location.
# Perform following tasks:
# 1. Pre-process the dataset.
# 2. Identify outliers.
# 3. Check the correlation.
# 4. Implement linear regression and random forest regression models.
# 5. Evaluate the models and compare their respective scores like R2, RMSE, etc
```

In [145]:

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

In [146]:

```
df = pd.read_csv("uber.csv")
```

In [147]:

```
# Explore and visualize
df.head()
```

Out[147]:

	Unnamed: 0	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitud
0	0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.99951
1	1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.99471
2	2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.96256
3	3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.96531
4	4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.97308

In [148]:

```
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   fare_amount        200000 non-null  float64
 2   pickup_datetime    200000 non-null  object 
 3   pickup_longitude   200000 non-null  float64
 4   pickup_latitude    200000 non-null  float64
 5   dropoff_longitude  200000 non-null  float64
 6   dropoff_latitude   200000 non-null  float64
 7   passenger_count    200000 non-null  int64  
 8   distance           200000 non-null  float64
dtypes: float64(6), int64(2), object(1)
memory usage: 13.7+ MB
```

In [149]:

```
df.dtypes
```

Out[149]:

```
Unnamed: 0             int64
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
distance             float64
dtype: object
```

In [150]:

```
df.describe()
```

Out[150]:

	Unnamed: 0	fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	drop
count	200000.000000	200000.000000	200000.000000	200000.000000	200000.000000	200
mean	99999.500000	11.359955	-72.527638	39.935885	-72.525289	
std	57735.171256	9.901776	11.437787	7.720539	13.117375	
min	0.000000	-52.000000	-1340.648410	-74.015515	-3356.666300	-
25%	49999.750000	6.000000	-73.992065	40.734796	-73.991407	
50%	99999.500000	8.500000	-73.981823	40.752592	-73.980093	
75%	149999.250000	12.500000	-73.967154	40.767158	-73.963658	
max	199999.000000	499.000000	57.418457	1644.421482	1153.572603	

In [151]:

```
df.drop(columns = ["Unnamed: 0"], inplace = True)
```

In [152]:

```
df.dtypes
```

Out[152]:

```
fare_amount          float64
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count        int64
distance             float64
dtype: object
```

In [153]:

```
#handling longitude and latitude
df.head()
```

Out[153]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.7
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.7
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.7
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.8
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.7

In [154]:

```
# 1preprocessing
# Handling missing values
```

In [155]:

```
df.isnull().sum()
```

Out[155]:

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
distance             0
dtype: int64
```

In [156]:

```
df["dropoff_latitude"].fillna(int(df["dropoff_latitude"].mean()), inplace = True)
df["dropoff_longitude"].fillna(int(df["dropoff_longitude"].mean()), inplace = True)
df.isnull().sum()
```

Out[156]:

```
fare_amount          0
pickup_datetime      0
pickup_longitude     0
pickup_latitude      0
dropoff_longitude    0
dropoff_latitude     0
passenger_count      0
distance             0
dtype: int64
```

In [157]:

```
# import math
# def hav_distance(lat1, long1, lat2, long2):
#     R = 6731.0
#     lon_1,lon_2, lat1, lat2 = map(np.radians,[long1, long2, lat1,lat2])
#     d_lon = lon_2 - lon_1
#     d_lat = lat2 - lat1

#     #calculating distance
#     km = 2 * 6731 * np.arcsin(np.sqrt(np.sin(d_lat/2.0)**2  + np.cos(lat1) * np.cos(lat2) * np.sin(d
#     return km
```

In [158]:

```
# df["distance"] = hav_distance(df["pickup_latitude"], df["pickup_longitude"], df["dropoff_latitude"]
# df["distance"] = df["distance"].astype(float).round(2)
```

In [159]:

```
# df.head()
```

In [160]:

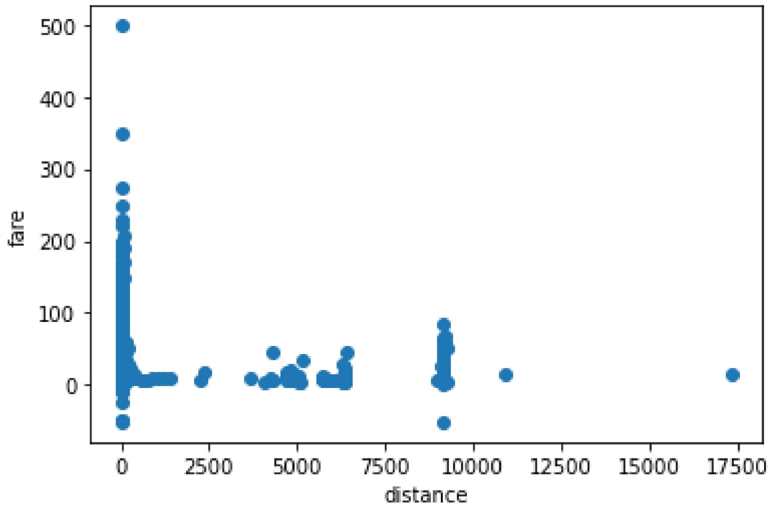
```
# df.to_csv("ubers.csv")
```

In [161]:

```
plt.scatter(df["distance"], df["fare_amount"])
plt.xlabel("distance")
plt.ylabel("fare")
```

Out[161]:

```
Text(0, 0.5, 'fare')
```



In [162]:

```
# def handling_outliers(df, column):
#     Q1 = df[column].quantile(0.25)
#     Q3 = df[column].quantile(0.75)
#     iqr = Q3-Q1

#     lower = Q1 - 1.5*iqr
#     upper = Q3 + 1.5*iqr
#     df = df[ (df[column] > lower) & (df[column] < upper) ]
#     return df
# # this method is no good
```

In [163]:

```
# df = handling_outliers(df,"fare_amount")
```

In [164]:

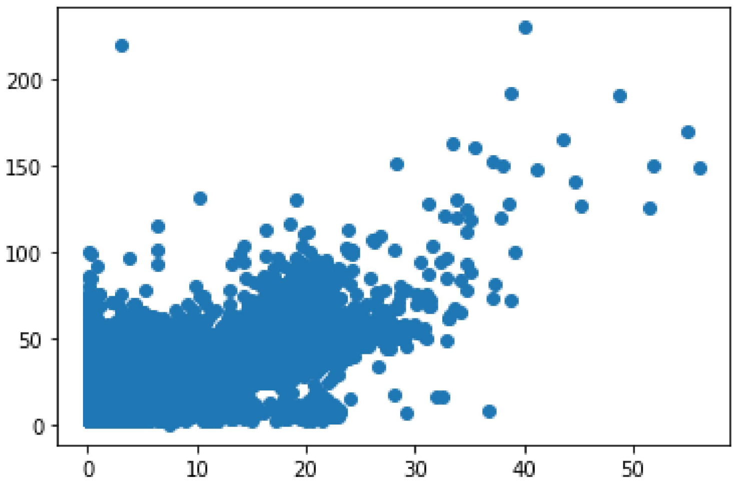
```
# df = handling_outliers(df,"distance")
```

In [165]:

```
#2 Handling outliers
df.drop(df[df["distance"] > 60].index, inplace = True)
df.drop(df[df["distance"] == 0].index, inplace = True)

df.drop(df[df["fare_amount"] == 0].index, inplace = True)
df.drop(df[df["fare_amount"] < 0].index, inplace = True)

# inplausible
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)
```

In [166]:
```
plt.scatter(df["distance"], df["fare_amount"]) # the perfect data in this assign
```

Out[166]:
```
<matplotlib.collections.PathCollection at 0x1a522092f10>
```



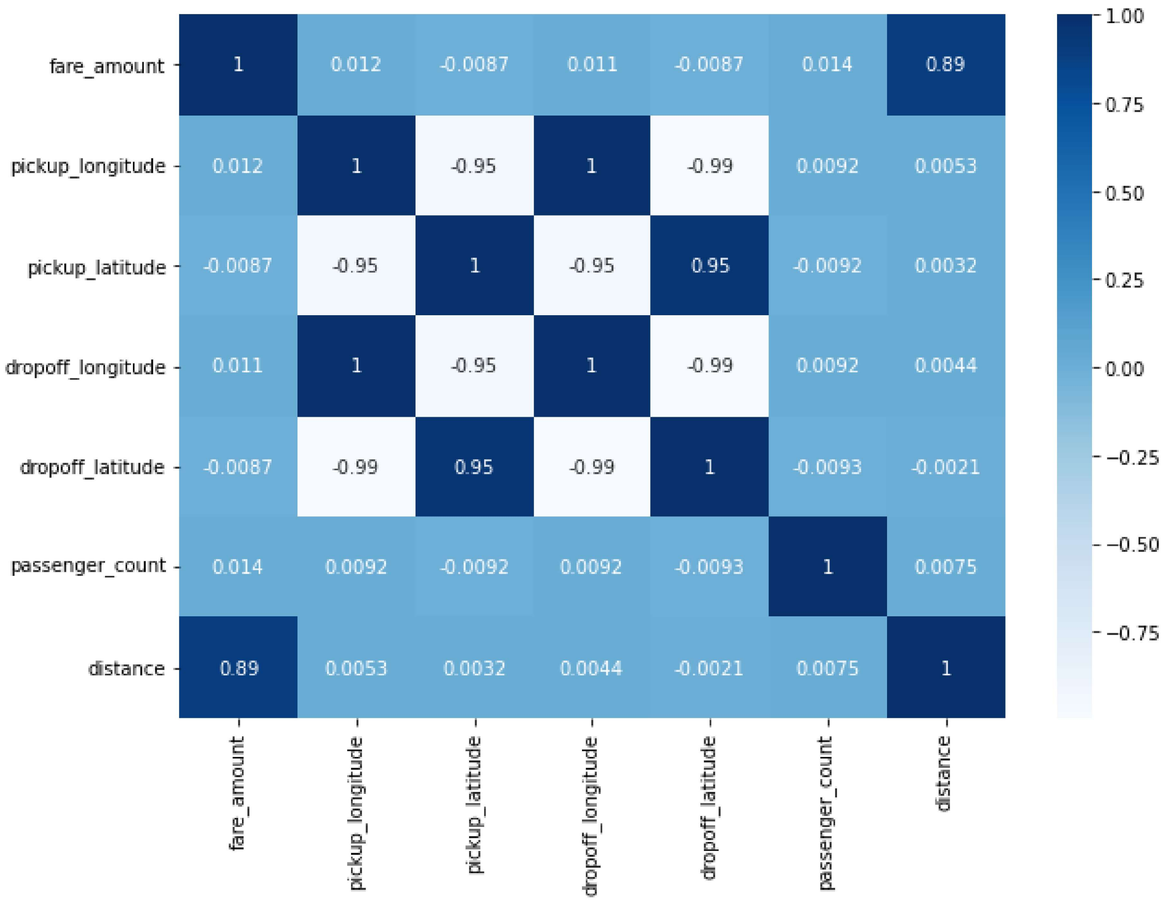
In [167]:
```
# df["pickup_datetime"] = pd.to_datetime(df["pickup_datetime"])
# df["year"] = df["pickup_datetime"].apply(lambda time: time.year)
# df["month"] = df["pickup_datetime"].apply(lambda time: time.month)
# df["date"] = df["pickup_datetime"].apply(lambda time: time.day)
# df["day of week"] = df["pickup_datetime"].apply(lambda time: time.dayofweek)
# df["day of week_num"] = df["pickup_datetime"].apply(lambda time: time.dayofweek)
# df["hour"] = df["pickup_datetime"].apply(lambda time: time.hour)

# day_map = {0: "Mon", 1: "Tue", 2 : "Wed", 3: "Thu", 4: "Fri", 5 : "Sat", 6: "Sun"}
# df["day_of_week"] = df["day of week"].map(day_map)
```

In [168]:

```
#3  check the correlation
cov_matrix = df.corr()
plt.figure(figsize=(10,7))
sns.heatmap(cov_matrix,annot = True, cmap = "Blues")
```

Out[168]:

```
<AxesSubplot:>
```



In [169]:
```
# 4.1 Training the LR model
X = df[["distance"]]
y = df["fare_amount"]

X
```

Out[169]:

	distance
0	1.78
1	2.60
2	5.32
3	1.76
4	4.73
...	...
199995	0.12
199996	1.98
199997	13.58
199998	3.74
199999	5.72

193489 rows × 1 columns

In [170]:
```
from sklearn.preprocessing import StandardScaler
std = StandardScaler()
X = std.fit_transform(X)
X = pd.DataFrame(X, columns = std.get_feature_names_out())
```

In [171]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.25)
```

In [172]:
```
from sklearn.linear_model import LinearRegression
model1 = LinearRegression()
model1.fit(X_train, y_train)
```

Out[172]:
```
LinearRegression()
```

In [173]:
```
y_pred = model1.predict(X_test)
y_pred
```

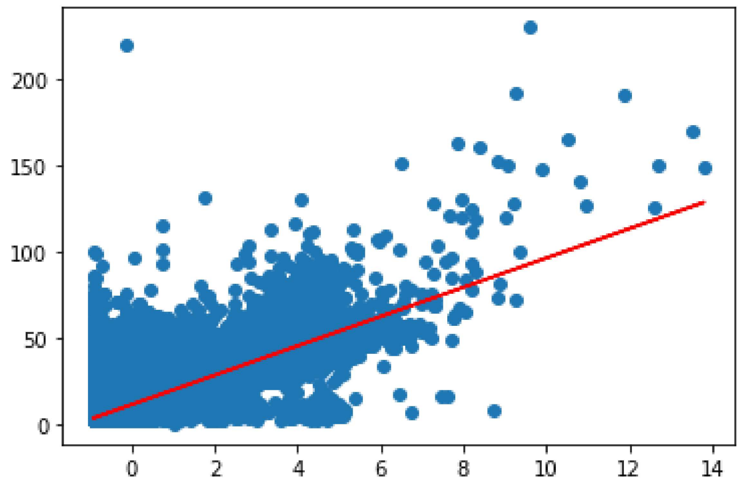
Out[173]:
```
array([ 8.08449705, 14.82697664,  8.41938842, ..., 12.90693278,
        6.49934456, 15.07256365])
```

In [174]:

```
plt.scatter(X,y)
plt.plot(X_train["distance"].values, model1.predict(X_train), color = "red")
```

Out[174]:

```
[<matplotlib.lines.Line2D at 0x1a52231c400>]
```



In [175]:

```
print("Training score: ",model1.score(X_train,y_train))
LR = model1.score(X_train,y_train)
LR
```

```
Training score:  0.7948803352187008
```

Out[175]:

```
0.7948803352187008
```

In [176]:

```
from sklearn.metrics import mean_absolute_error
linear_score = mean_absolute_error(y_pred,y_test)
linear_score
```

Out[176]:

```
2.270182647597846
```

In [177]:

```
from sklearn.ensemble import RandomForestRegressor
df.head()
```

Out[177]:

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_la
0	7.5	2015-05-07 19:52:06 UTC	-73.999817	40.738354	-73.999512	40.7
1	7.7	2009-07-17 20:04:56 UTC	-73.994355	40.728225	-73.994710	40.7
2	12.9	2009-08-24 21:45:00 UTC	-74.005043	40.740770	-73.962565	40.7
3	5.3	2009-06-26 08:22:21 UTC	-73.976124	40.790844	-73.965316	40.8
4	16.0	2014-08-28 17:47:00 UTC	-73.925023	40.744085	-73.973082	40.7

In [178]:

```
X1 = df[["pickup_longitude", "pickup_latitude", "dropoff_latitude", "passenger_count", "distance"]]
y1 = df["fare_amount"]
```

In [179]:

```
X1_train, X1_test, y1_train, y1_test = train_test_split(X1,y1, test_size = 0.25)
X1_train.head()
```

Out[179]:

	pickup_longitude	pickup_latitude	dropoff_latitude	passenger_count	distance
36695	-73.949072	40.711233	40.792264	1	9.77
61452	-73.979142	40.762402	40.766948	1	2.30
49029	-73.873028	40.774132	40.645303	1	24.15
120760	-73.984505	40.745483	40.764572	1	2.55
122162	-74.005195	40.740427	40.757148	2	2.85

In [ ]:

```
model2 = RandomForestRegressor()
model2.fit(X1_train,y1_train)
```

In [ ]:

```
y1_pred = model2.predict(X1_test)
```

In [ ]:

```
obs = pd.DataFrame({"actual": y1_test, "predicted": y1_pred})
plt.scatter(X1_test["distance"], obs["predicted"])
plt.scatter(X1_test["distance"], obs["actual"], color = "red")
plt.show()
Rf = model2.score(X1_train, y1_train)
```

In [ ]:

```
from sklearn.metrics import mean_squared_error
rmse = np.sqrt(mean_squared_error(y1_test, y1_pred))
print(rmse)
```

In [ ]:

```
models = ["LinearRegression", "RandomForest"]
scores = [LR*100,Rf*100 ]
plt.bar(models,scores )
```

In [ ]:

```

```