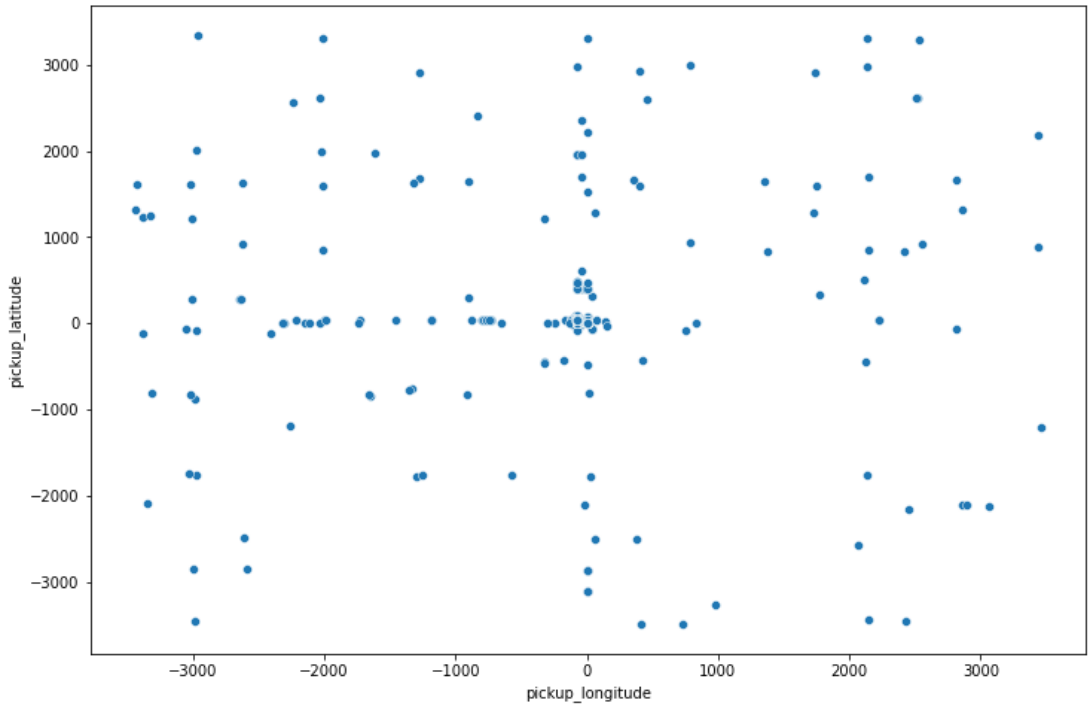
TAXI FARE :

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

import xgboost as xgb
from sklearn.metrics import mean_squared_error as mse
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler

train_df =  pd.read_csv('https://raw.githubusercontent.com/Premalatha-success/Datasets/main/TaxiFare.csv', nrows = 10_000_000)
test_df = pd.read_csv('https://raw.githubusercontent.com/Premalatha-success/Datasets/main/TaxiFare.csv')

train_df.drop('key', axis=1, inplace=True)
```

```
train_df.head()

train_df.isna().sum()
pd.set_option('display.float_format', lambda x: '%.5f' % x)
train_df.describe()

(train_df['fare_amount'] < 0).sum()

((train_df['pickup_longitude'] < -180) | (train_df['pickup_longitude'] > 180)).sum()
((train_df['pickup_latitude'] < -90) | (train_df['pickup_latitude'] > 90)).sum()

((train_df['passenger_count'] == 0) | (train_df['passenger_count'] > 6)).sum()
(train_df['passenger_count'] > 6).sum()

print("Train Passenger count equals 0: ", (train_df['passenger_count'] == 0).sum())
print("Test Passenger count equals 0: ", (test_df['passenger_count'] == 0).sum())

plt.figure(figsize=(10, 6))
sns.histplot(train_df['fare_amount']);
plt.title('Distribution of Fare Amount');

def clean_df(df):
    new_df = df[
        ((df['fare_amount'] > 0) & (df['fare_amount'] <= 200)) &
        ((df['pickup_longitude'] > -75) & (df['pickup_longitude'] < -73)) &
```

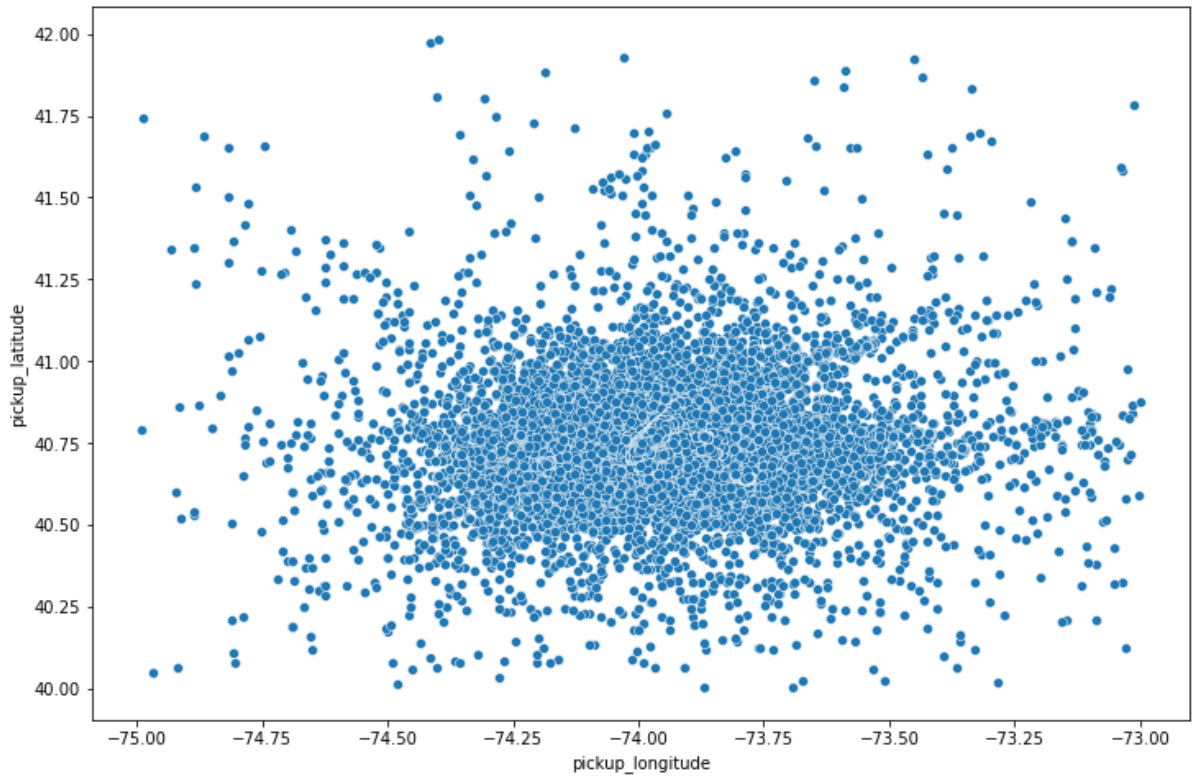
```
        ((df['pickup_latitude'] > 40) & (df['pickup_latitude'] < 42)) &
        ((df['dropoff_longitude'] > -75) & (df['dropoff_longitude'] < -73)) &
        ((df['dropoff_latitude'] > 40 & (df['dropoff_latitude'] < 42))) &
        ((df['passenger_count'] > 0) & (df['passenger_count'] <= 6))
    ]

    return new_df
plt.figure(figsize=(12,8))
sns.scatterplot(x=train_df['pickup_longitude'], y=train_df['pickup_latitude'])
```



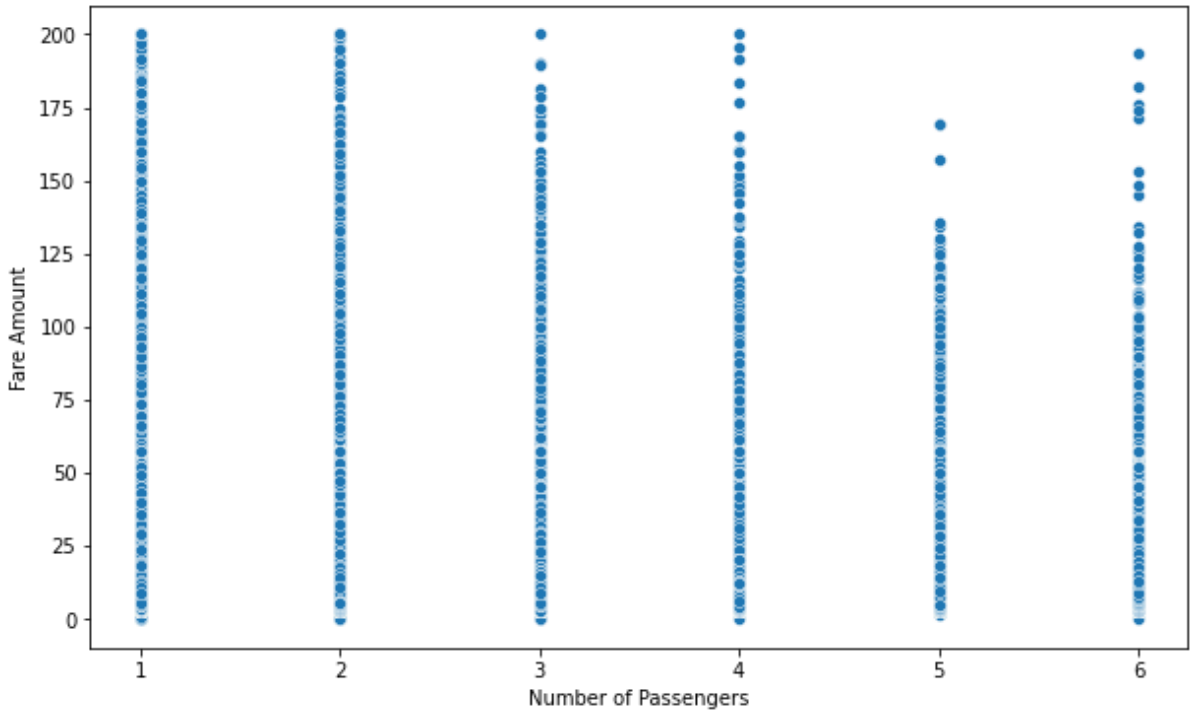

```
print("Before:", len(train_df))
train_df = clean_df(train_df)
print("After:", len(train_df))
```

Before: 10000000
After: 9754583

```
plt.figure(figsize=(12,8))
sns.scatterplot(x=train_df['pickup_longitude'], y=train_df['pickup_latitude'])
```




```
plt.figure(figsize=(10, 6))
sns.scatterplot(x=train_df['passenger_count'], y=train_df['fare_amount'])
plt.xlabel('Number of Passengers')
plt.ylabel('Fare Amount')
```




```
train_df.describe()
```

```
def manhattan_dist(lat_p, long_p, lat_d, long_d):
    distance = np.abs(lat_d - lat_p) + np.abs(long_d - long_p)

    return distance
```

```
def add_datetime_info(df, transform_datetime=False):
    if transform_datetime:
        df['pickup_datetime'] = pd.to_datetime(df['pickup_datetime'], format="%Y-%m-%d %H:%M:
%S UTC")

    df['hour'] = df['pickup_datetime'].dt.hour
    df['day'] = df['pickup_datetime'].dt.day
    df['month'] = df['pickup_datetime'].dt.month
    df['year'] = df['pickup_datetime'].dt.year
    df.drop('pickup_datetime', axis=1, inplace=True)
```

```
def transform(df, transform_datetime):
    add_datetime_info(df, transform_datetime)
    add_airport_info(df)
    df['manhattan_dist'] = manhattan_dist(df['pickup_latitude'], df['pickup_longitude'], df['dropof
f_latitude'], df['dropoff_longitude'])
    return df

train_df = transform(train_df, transform_datetime=True)
```

```
X = train_df.drop(['fare_amount'], axis=1)
y = train_df['fare_amount']
```

In [20]:

```
X_train, X_val, y_train, y_val = train_test_split(X, y, random_state=42, test_size=0.05)
del(X)
del(y)
```

```
print("Train set error: ", np.sqrt(mse(y_train, y_train_pred)))
print("Validation set error: ", np.sqrt(mse(y_val, y_val_pred)))
```

```
Train set error:  3.1490281686166424
Validation set error:  3.4571341119704186
```

```
submit=sample_new.to_csv("submit.csv", index=False)
```