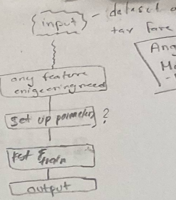
Euclidean Distance $\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$

(x_1, y_1) = coordinate of first point (pickup)

(x_2, y_2) = coordinate of Second point (drop off)



{1}

Angel Martinez-Diaz

```
# import libraries used/needed

  import pandas as pd
  import numpy as np
  import matplotlib as plt
  from keras import Layers
# import Dataset

  df= pd.read_csu('NYC_taxFare.csu')
  df.head() # checking if it imported successfully

# Finding distance of pickup to dropoff

  def euc_distance (lat1, long1, lat2, long2):
          return ((((lat1 - lat2)**2 + (long1 - long2)**2)**.05)

  df('distance') = euc_distance (df['pickup_latitude'], df[pickup_longtitude]
# inserting a new 'distance" column   df['dropoff_latitude'], df['dropoff_longtitude'])
  after calculating pick/drop distance

# dropping columns
data = drop. df(['pickup_latitude'], [df['pickup_longtitde'], df['dropoff_latitude'], df['dropoff_longtitude'], df['pickup_datetime']], axis = 1)
  data.head()
# train the model
# from sklearn.model_selection import train_test_split
  x_train, x_test, y_test, y_tran = train_test_split
 From keras models import Sequential

# compile your model
model = models.Sequential()
model.add(layers.Dense(10, activation='relu', input_shape=(33,)))
model.add(layers.Dense(10, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
model.compile(optimizer='adam', loss='binary_crossentropy, metrics=['accuracy']
model.fit(x_train, y_train, epochs=10, batch = 33)
 results = model.evaluate (x_test, y_test)
```

So, in order to write a program, that can help predict tax fare, we must First, calculate the distance from pick up w/ drop off, to see the total distance driven (assume charge per mile). So in order to do so we used a Formula called 'Euclidean Distance', where we take the coordinates (long. & lat.) of pick up & drop off to figure out the distance that was driven for each ride.

After, writing the code for the Euclidean Distance, we have to remove any columns that don't factor in