~~Cross-binary~~entropy

optimizer = Sgd   Loss= ME   Learning rate .01

This code lacks several line of code that I cannot recall correctly. This code creates a function called Euclidean_distance, which inputs 4 important variables from the provided data. These are dropoff_latitude, Pickup_latitude, dropoff_longitude, and pickup_longitude. The function subtracts the related values from eachother while also applying an exponent of 2 too each outcome. It then adds the outcome and squares the final output. This gives us a 'distance' variable that we can now use to predict the targeted Y value of fare_amount. We then begin to create the deep learning model structure this is done by adding two hidden layer models and an outcome model. I think Standard Gradient descent will provide the best results with this data. We will use this as our optimizer function. We will also use tanh

Code 2

```
Import pandas as pd
Import models from keras
pd.read.df('NYC_taxi_fares.csv)
Print df(is.null)
drop df.isnull values

def euc_distance(lat1,long1,lat2,long2):
    return (((lat1-lat2)**2 +(long1-long2**2) ** .5)
df['distance'] = euc_distance(df['pickup_latitude'], df['Pickup_longitude']
        df['dropoff_latitude], df[dropoff_longitude])
 x_data = df['distance'] df[dropoff_longitude] df[dropoff_latitude] df[pickup_longitude],
df[Pickup_latitude]
 Y_data = df['fare_amount']
→ Split train and test data
 add models.denselayer 16 ('rELU')
 add models.denselayer 8 ('rELU')
 add models.denselayer 1 ('Sigmoid')
 optimizer = Sgd   Loss = Mean Squared   Learning rate = .1
```

This code is the same except we added additional features to the X Data We also Increased the nodes in the hidden layer