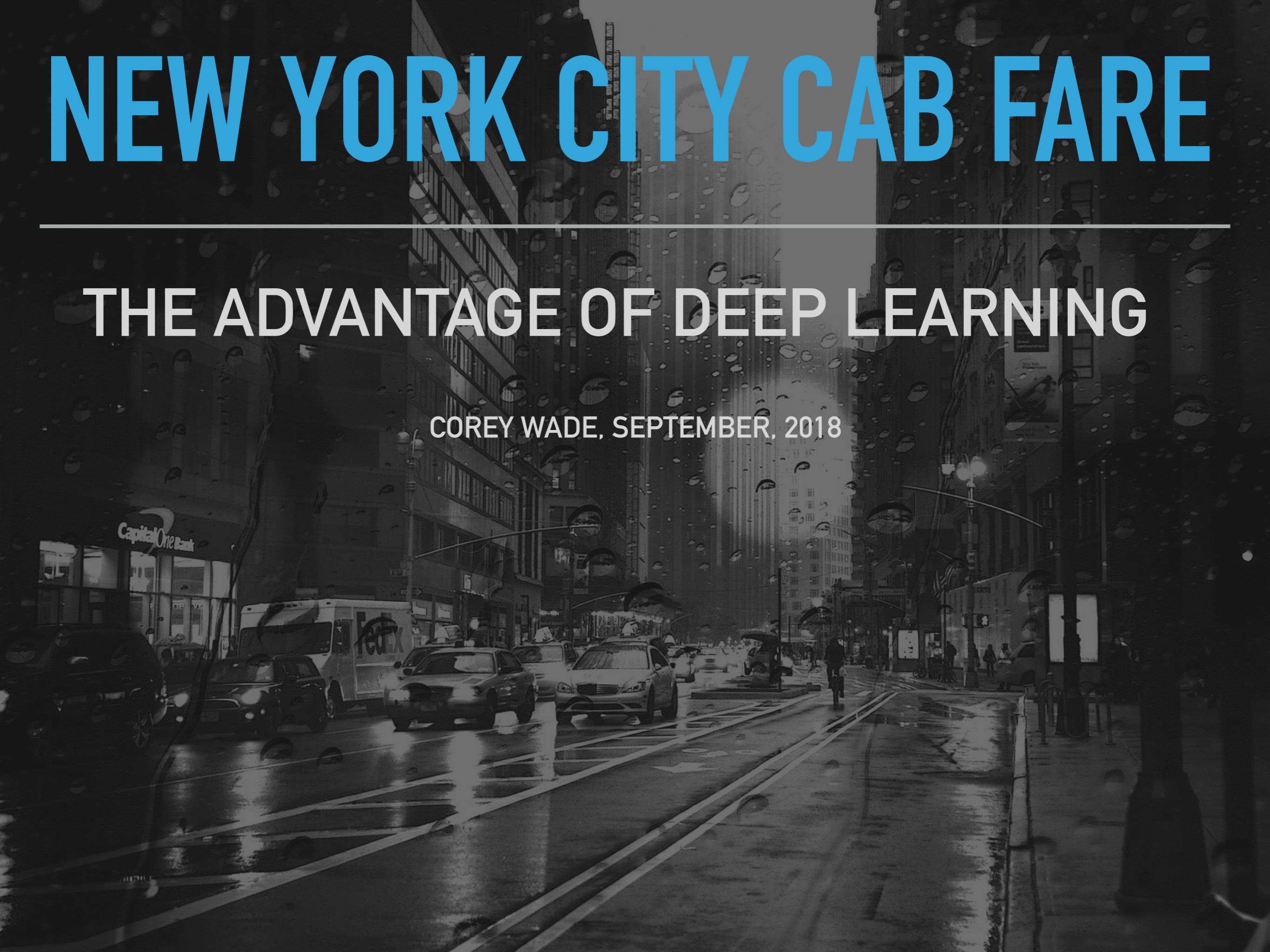


NEW YORK CITY CAB FARE

THE ADVANTAGE OF DEEP LEARNING

COREY WADE, SEPTEMBER, 2018





I LOVE TRAFFIC.
IT'S FANTASTIC.
NEW YORK
TRAFFIC IS SO
RELAXING.

Chris Kattan

NEW YORK CITY CAB FARE DATASET

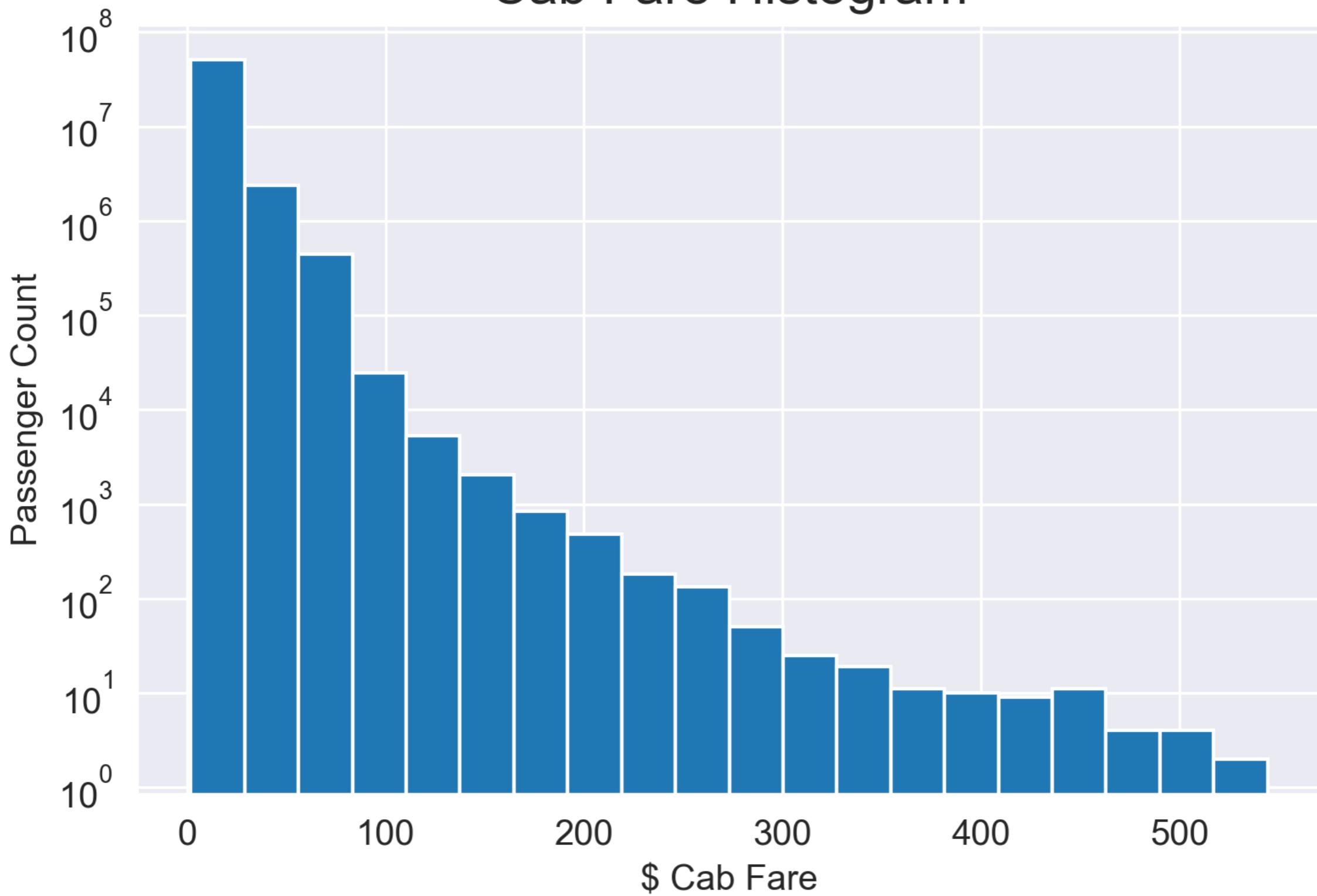
- ▶ Kaggle Competition sponsored by Google/Coursera.
- ▶ 55+ million rows.
- ▶ Years include 2009-2015.
- ▶ Goal: predict cab fare.
- ▶ Given: latitude/longitude of pickup/drop-off, timestamp of pickup, and passenger_count.
- ▶ That's all.

CAB FARE FACTS & FABRICATIONS

- ▶ Mean fare: \$11.35.
- ▶ Median fare: \$8.50.
- ▶ Max fare: \$93,963.36.
- ▶ Min fare: -\$300.00.
- ▶ According to city data, current min cab fare is \$2.50.
- ▶ Errors/Outliers need work.

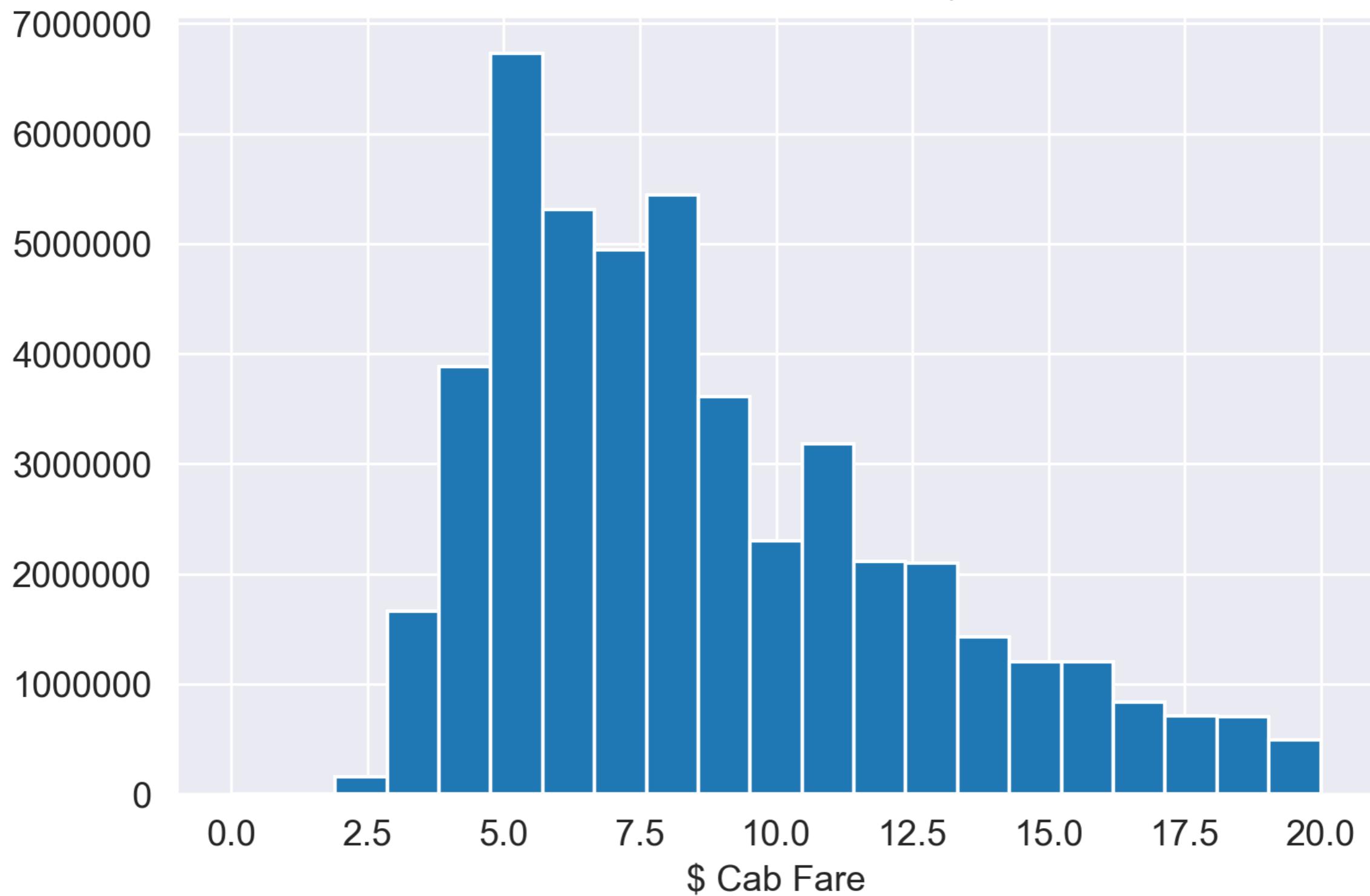


Cab Fare Histogram



Note that the graph is a log scale. Cab fares are very right skewed.

Cab Fares Under \$20



It looks like the majority of fares are between 4 and 10 dollars.

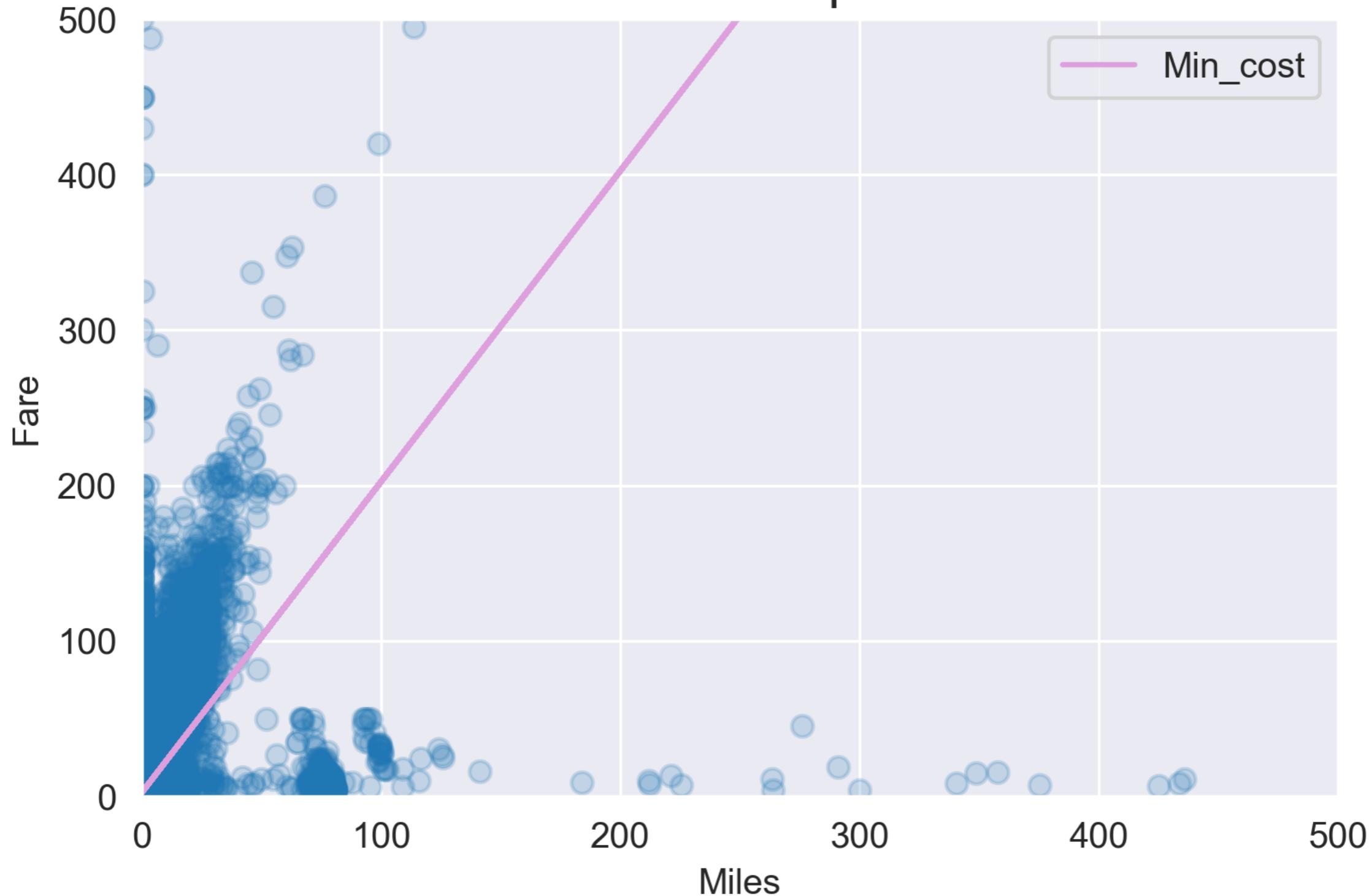
ELIMINATING ERRORS & OUTLIERS

- ▶ Eliminate all NaN values.
 - ▶ With 55,000 rows, this is safe.
- ▶ Eliminate cab fares under \$2.50.
 - ▶ This min is revealed by the histogram.
- ▶ Eliminate cab fares that do not start or end in NYC.
 - ▶ Need to find NYC latitude & longitudes
- ▶ Convert latitude & longitude pickup & drop-off to distance metric
 - ▶ Can be used to find unrealistic outliers

DISTANCE METRICS

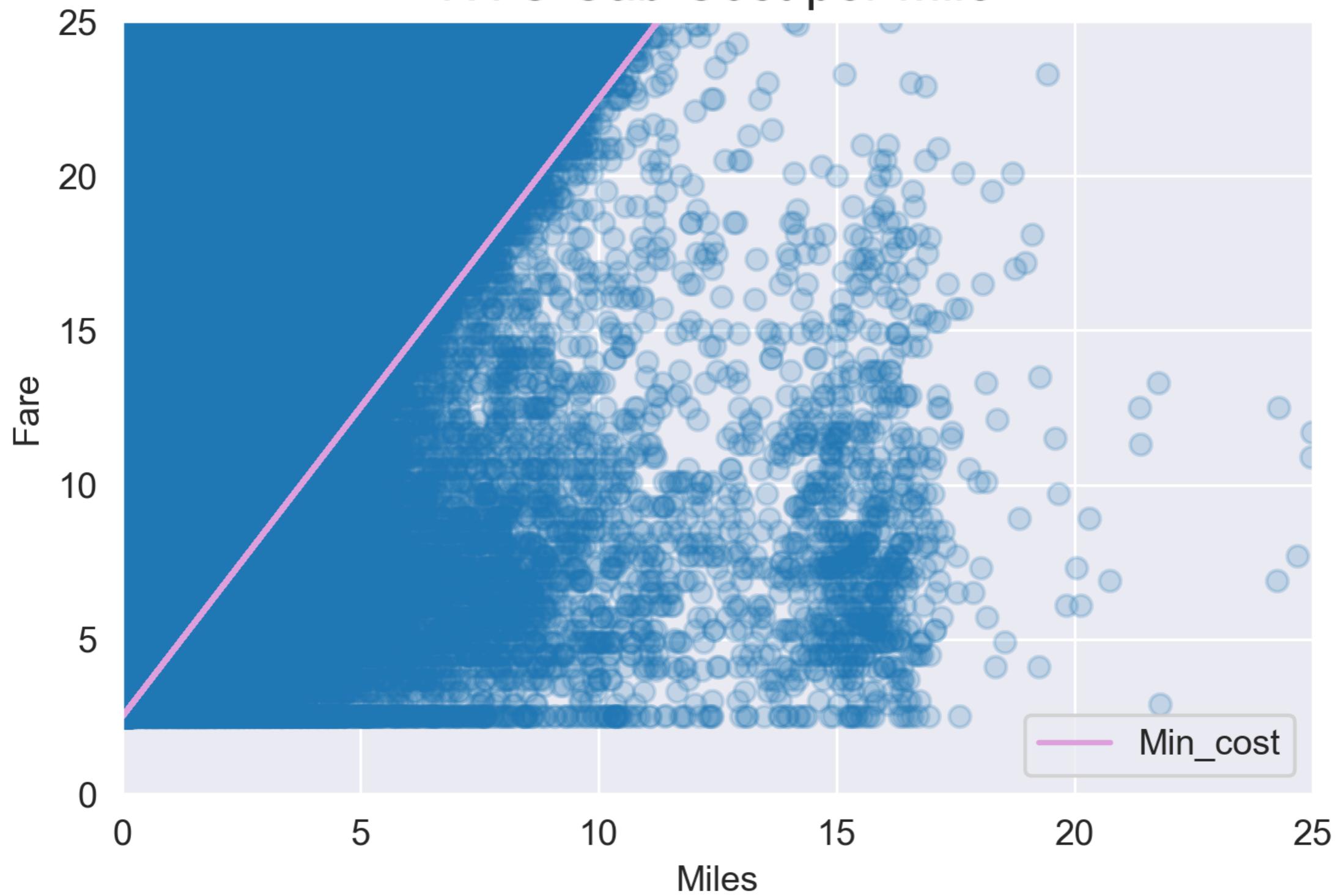
- ▶ Euclidean Distance measures the distance squared (Pythagorean) between 2 points. Ideal for flat surfaces.
- ▶ Taxicab Distance (Manhattan) measures the absolute value distance between 2 point. Realistic measure for city blocks.
- ▶ Spherical Distance measures the geometric distance between 2 points on a spherical surface.
- ▶ Vincenty Distance makes adjustments to spherical distance to account for actual curvature of the Earth.
- ▶ Euclidean Distance, perhaps not the most accurate, gave the best results. This may due to Euclidean Line Functions used later.

NYC Cab Cost per Mile



The Min_cost line separates the data nicely. All data points underneath the line are unrealistic.

NYC Cab Cost per Mile



Zooming in, we can see a striking difference between above and below the Min_cost line.

DISTANCE SCATTER PLOTS

- ▶ The min cost is a base-fare of \$2.50 plus \$2.00 per additional mile. All points under this line can be eliminated.
- ▶ The most expensive rides traveled very low distances. This is possible if people take long rides and get dropped at the same place.
- ▶ All cab fares with distances of 0 miles were eliminated.

MORE FEATURE ENGINEERING

- ▶ Create Manhattan column by circumscribing Manhattan with quadrilateral using lat/lon coordinates.
- ▶ Create Newark, JFK columns with same approach.
- ▶ Create standard time columns using minutes, hours, day_of_week, month, and total seconds.
- ▶ Create additional time columns for morning rush, weekday surcharge, night surcharge, 15-minute intervals.
- ▶ Eliminate unrealistic max fares.

A DEEP LEARNING
SYSTEM DOESN'T
HAVE ANY
EXPLANATORY
POWER.

Geoffery Hinton



MACHINE LEARNING MODELS

- ▶ Linear Regression
- ▶ Ridge Regression
- ▶ Decision Trees
- ▶ Random Forest
- ▶ Light GBM
- ▶ Deep Learning Sequential
- ▶ Top Performers in Blue



DEEP LEARNING INTRODUCTION

- ▶ Keras is a high-level deep learning introduction to Tensor Flow.
- ▶ Standard inputs: X, y, nodes, batch_size, activation, optimizer, loss.
- ▶ The nodes are the number of hidden layers in an array. For example, [100,100,50], or [64, 32]. Any length and number is possible.
- ▶ Batch-size is the number of training examples used in each iteration, in my case, 32.
- ▶ Activation is the function used to start the run, like 'relu'.
- ▶ A popular optimizer is 'adam'.
- ▶ The loss for this particular example is the 'mean_squared_error'.

```
DEF DEEP_LEARNING(X_TRAIN, Y_TRAIN, NODES=[100,50,50], BATCH_SIZE=32,
ACTIVATION='RELU', OPTIMIZER='ADAM', LOSS='MEAN_SQUARED_ERROR'):
```

```
X, X_val, y, y_val = train_test_split(X_train, y_train, test_size=0.05) # Split into training/validation sets  
n_cols = X.shape[1] # Save the number of columns in predictors: n_cols  
model = Sequential() # Set up the model  
model.add(Dense(nodes[0], activation=activation, input_shape=(n_cols,))) # Add the first layer  
for i in range(len(nodes)-1): # Add additional layers  
    model.add(Dense(nodes[i+1], activation=activation, kernel_constraint=maxnorm(3)))  
    model.add(Dropout(0.2)) # Select dropout for regularization  
model.add(Dense(1)) # Add the output layer  
model.compile(optimizer=optimizer, loss=loss) # Compile the model  
early_stopping_monitor = EarlyStopping(patience=3) # Define early_stopping_monitor  
model.fit(X, y, validation_split=0.05, epochs=1000, batch_size=batch_size,  
callbacks=[early_stopping_monitor])  
score = model.evaluate(X_val, y_val)  
rmse = np.sqrt(score) # Get root mean squared error
```

DEEP LEARNING HIGHLIGHTS

- ▶ Nodes are fun to experiment with. They can be long and low, [8, 8, 8, 8, 8, 8, 8, 8] or short and high, [32000].
- ▶ Epochs are the number of times the model is trained. Early Stopping is a method for stopping the number of epochs when no improvement is shown over several rounds.
- ▶ Dropout is a standard regularization method. It selects a percentage of neurons to randomly remove over each round.
- ▶ The first layer and last layer need their own specifications.

DEEP LEARNING RESULTS

Train on 2429118 samples, validate on 127849 samples

Epoch 1/1000

2429118/2429118 [=====] - 107s 44us/step - loss: 12.4999 - val_loss: 9.7576

Epoch 2/1000

2429118/2429118 [=====] - 108s 45us/step - loss: 10.7824 - val_loss: 9.6408

Epoch 3/1000

2429118/2429118 [=====] - 111s 46us/step - loss: 10.5082 - val_loss: 9.3295

Epoch 4/1000

2429118/2429118 [=====] - 100s 41us/step - loss: 10.3397 - val_loss: 9.0330

Epoch 5/1000

2429118/2429118 [=====] - 96s 40us/step - loss: 10.2256 - val_loss: 9.4364

Epoch 6/1000

2429118/2429118 [=====] - 96s 40us/step - loss: 10.1093 - val_loss: 8.9241

Epoch 7/1000

2429118/2429118 [=====] - 97s 40us/step - loss: 9.9892 - val_loss: 8.9832

Epoch 8/1000

2429118/2429118 [=====] - 97s 40us/step - loss: 9.9665 - val_loss: 12.1911

Epoch 9/1000

2429118/2429118 [=====] - 97s 40us/step - loss: 9.9324 - val_loss: 9.0024

134578/134578 [=====] - 2s 12us/step

3.0603275437869937

Saved deep learning model as 'dl_model.json'



DEEP LEARNING ADVANTAGE

- ▶ I used 10% of the data due to the size. Deep Learning will perform better with more data.
- ▶ I achieved a min RMSE of about 2.8 with Deep Learning, very close to optimized Random Forests and LightGBM.
- ▶ Deep Learning consistently outperformed Linear Regression and Decision Trees.
- ▶ Deep Learning has more layers of experimentation.
- ▶ Since more data means better results, deep learning has the edge.



AND WHAT ABOUT
KAGGLE?

...Top Third Finish!

REFERENCES

- ▶ Github Repository: [https://github.com/coreyjwade/
NYC_Cab_Fare.](https://github.com/coreyjwade/NYC_Cab_Fare)
- ▶ Kaggle Competition Link: [https://www.kaggle.com/c/new-
york-city-taxi-fare-prediction.](https://www.kaggle.com/c/new-york-city-taxi-fare-prediction)
- ▶ Additional Photo Credits: [https://www.flickr.com/photos/
ugod/5329854658](https://www.flickr.com/photos/ugod/5329854658), [https://creativecommons.org/
licenses/by/2.0/](https://creativecommons.org/licenses/by/2.0/), [https://commons.wikimedia.org/wiki/
File:52nd_Street,_New_York_City,_NY_0001_original.jpg](https://commons.wikimedia.org/wiki/File:52nd_Street,_New_York_City,_NY_0001_original.jpg), William P. Gottlieb.