Maven Wave

# Demo 1

## Estimating Chicago Taxi Trip Duration with the Functional API in Keras

2019

# Agenda

**Data Exploration**

Review of dependent and independent variables

**Preprocessing**

Steps necessary for preparing the data for modeling

**Modeling**

Overview of architecture and tuning

**Evaluation**

Discussion of evaluation metrics and results

# Improving Chicago Taxi Service by Predicting Trip Duration

Machine Learning can be used to improve the customer experience by predicting the duration of their trip on the onset of the ride

- Millions of past rides used for training and evaluating the model
- Only attributes available at the onset of the ride are used for model building
- Trip durations are predicted within 41% of the actual trip duration on 1000 unseen test rides

# Data Exploration

Raw table:
bigquery-public-data:chicago_taxi_trips.taxi_trips

- 187,002,005 total rows
- 67.71 GB in size

# Exploring Features (Independent Variables)

## Potential features for modeling fall into three different categories

### Predictive and Available

These features are potentially predictive of the target attribute and are available at the start of the ride

| Feature | Type |
|---|---|
| trip_start_timestamp | Timestamp |
| pickup_census_tract | Categorical |
| dropoff_census_tract | Categorical |
| pickup_community_area | Categorical |
| dropoff_community_area | Categorical |
| pickup_latitude | Float |
| pickup_longitude | Float |
| dropoff_latitude | Float |
| dropoff_longitude | Float |

### Not Predictive or Not Available

These features are in the public Chicago Taxi dataset but are either not predictive of trip duration or not available at the start of the ride.

| Feature | Type |
|---|---|
| unique_key | String |
| taxi_id | String |
| trip_end_timestamp | Timestamp |
| trip_miles | Float |
| fare | Float |
| tips | Float |
| toles | Float |
| extras | Float |
| trip_total | Float |
| payment_type | Categorical |
| company | String |

### Engineered

These additional fields are added to the feature set during the preprocessing phase using the available fields.

| Feature | Description |
|---|---|
| distance | The distance between the pickup and dropoff lat / long values using the ST_DISTANCE() function in BigQuery |
| hour_start | Hour of the trip_start_timestamp field |
| month_start | Month of the trip_start_timestamp field |
| weekday | Day of week of the trip_start_timestamp field |

# Feature Selection

## Not all available features are used for modeling

### Predictive and Available

All available and predictive features are included in the model with the exception of trip_start_timestamp, as this is represented by the engineered features

| Feature | Type |
|---|---|
| trip_start_timestamp | Timestamp |
| pickup_census_tract | Categorical |
| dropoff_census_tract | Categorical |
| pickup_community_area | Categorical |
| dropoff_community_area | Categorical |
| pickup_latitude | Float |
| pickup_longitude | Float |
| dropoff_latitude | Float |
| dropoff_longitude | Float |

### Not Predictive or Not Available

These features are not included in the model as they are either not predictive of trip duration or not available at the start of the ride.

| Feature | Type |
|---|---|
| unique_key | String |
| taxi_id | String |
| trip_end_timestamp | Timestamp |
| trip_miles | Float |
| fare | Float |
| tips | Float |
| toles | Float |
| extras | Float |
| trip_total | Float |
| payment_type | Categorical |
| company | String |

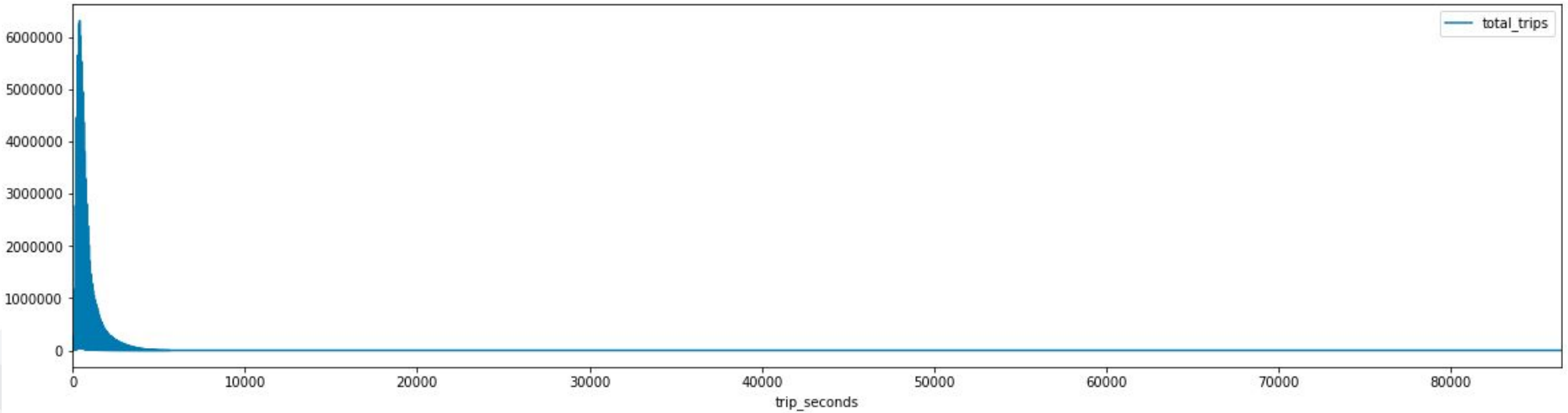
### Engineered

All engineered features are included in the model.

| Feature | Description |
|---|---|
| distance | The distance between the pickup and dropoff lat / long values using the ST_DISTANCE() function in BigQuery |
| hour_start | Hour of the trip_start_timestamp field |
| month_start | Month of the trip_start_timestamp field |
| weekday | Day of week of the trip_start_timestamp field |

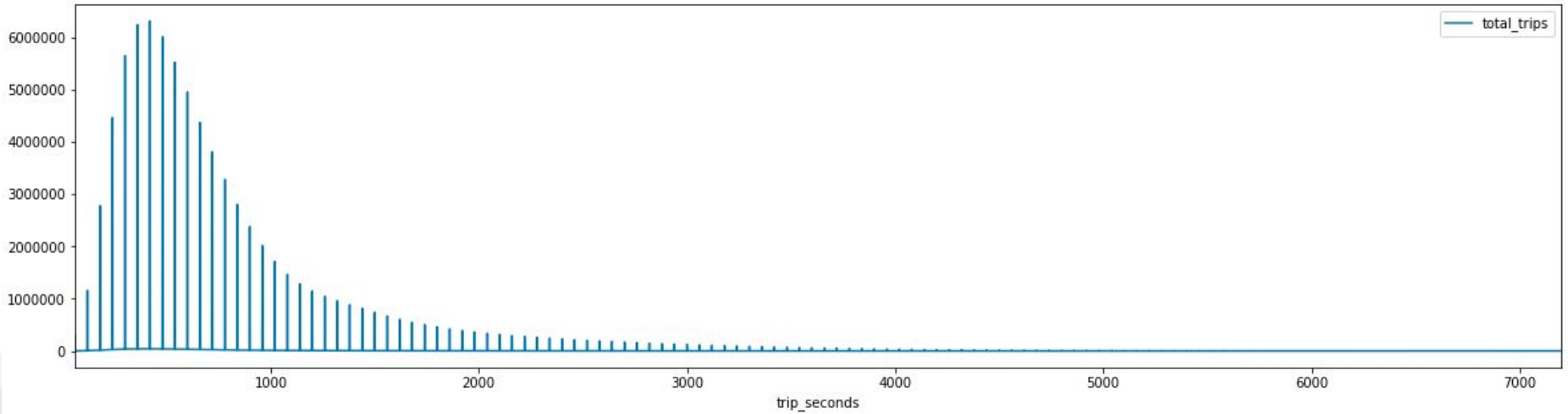
# Exploring the Target Attribute (Dependent Variable): Trip Duration in Seconds

- Overall the distribution of number of trips by duration has a very long tail with some trips in the dataset lasting up to two days
- To remove outliers and improve model performance, only trips with a duration up to two hours were included for training



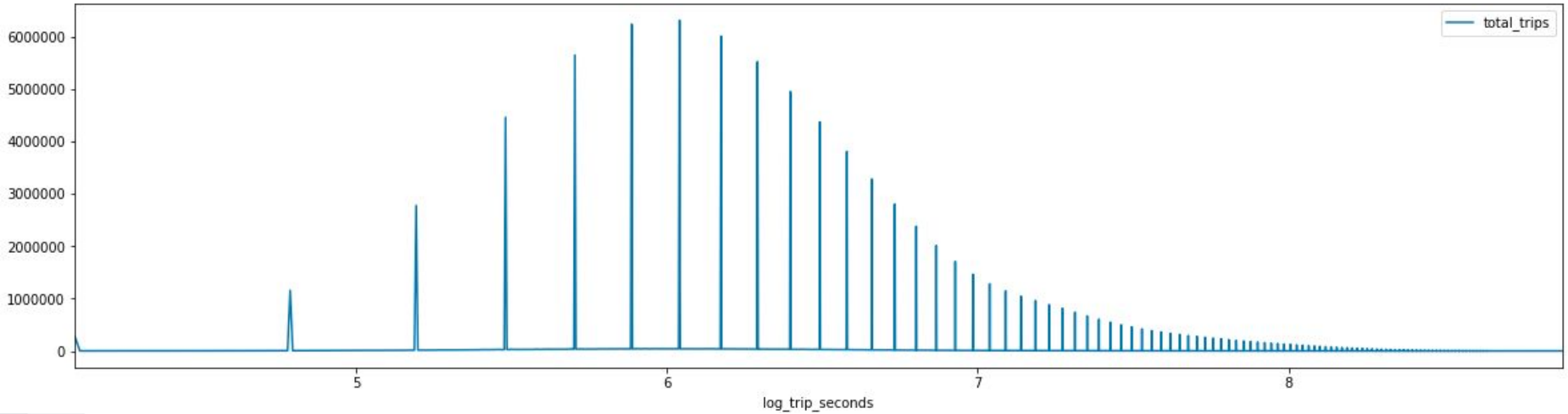
# Exploring the Target Attribute: Log-normal Distribution

- Even after controlling for outliers, our target attribute still has a long tail, or a “log-normal” distribution
- Training a regression model where the target attribute has this type of distribution can lead to large coefficients and slow or no learning



# Exploring the Target Attribute: Normalization

- To control for this, we take the natural logarithm of the trip duration as a preprocessing step prior to training. The result is that our target attribution has a much more normal distribution.
- Note this means that our predicted values will also be the natural logarithm of seconds. To convert this back to seconds, we implement a custom prediction method to return predicted trip duration in seconds by passing the initial prediction through the exponential function.



# Feature Selection Summary

| Continuous Features | Categorical Features | Target Attribute |
| --- | --- | --- |
| distance | weekday | log_trip_seconds |
| hour_start | pickup_census_tract | |
| month_start | dropoff_census_tract | |
| pickup_latitude | pickup_community_area | |
| pickup_longitude | dropoff_community_area | |
| dropoff_latitude | | |
| dropoff_longitude | | |

# Preprocessing

Preprocessing occurs in two stages:

- Generating artifacts necessary for training and inference
- Preprocessing rows prior to training and / or passing for inference

# Preprocessing Artifact: Training, Test, and Validation Sets

## Preprocess and split raw data using BigQuery

- Transformations such as calculating logarithms and distances are implemented directly in BigQuery syntax
- Raw data is filtered based on requirements for target attribute
- Missing values are addressed
- Extracted features are split into training, test, and validation tables in BigQuery and files in Cloud Storage

**train_results**

113,148,861 rows (approx. 80%)

**test_results**

14,139,546 rows (approx 10%)

**val_resuts**

14,145,927 rows (approx. 10%)

**full_train_results.csv**

13.25 GB

**full_test_results.csv**

1.66 GB

**full_val_resuts.csv**

1.66 GB

# Preprocessing Artifact: MinMax Normalizer

- Our feature set includes 7 continuous features with vastly different scales. Training a neural net without first scaling numerical input can lead to erratic or no learning.
- To account for this, we normalize continuous features to a zero to one scale. The scaler must be fit using all 113M rows in the training set, and the same scaler fit on the training set must be saved and used to scale continuous features in the validation and test sets for evaluation and inference respectively.

| distance | hour_start | month_start | pickup_latitude | pickup_longitude | dropoff_latitude | dropoff_longitude |
|---|---|---|---|---|---|---|
| 4092.426273 | 20 | 10 | 41.949829 | -87.643965 | 41.919225 | -87.671446 |
| 2239.665961 | 21 | 1 | 42.001571 | -87.695013 | 42.009623 | -87.670167 |
| 3622.427784 | 18 | 11 | 41.946511 | -87.806020 | 41.954028 | -87.763399 |
| 4671.111033 | 3 | 1 | | | | |
| 2239.665961 | 1 | 7 | | | | |

| distance | hour_start | month_start | pickup_latitude | pickup_longitude | dropoff_latitude | dropoff_longitude |
|---|---|---|---|---|---|---|
| 0.7620 | 0.95 | 0.9 | 0.104399 | 1.000000 | 0.000000 | 0.903663 |
| 0.0000 | 1.00 | 0.0 | 0.879398 | 0.685000 | 1.000000 | 0.916232 |
| 0.5687 | 0.85 | 1.0 | 0.054702 | 0.000000 | 0.384994 | 0.000000 |
| 1.0000 | 0.10 | 0.0 | 0.000000 | 0.546212 | 0.330046 | 1.000000 |
| 0.0000 | 0.00 | 0.6 | 1.000000 | 0.838317 | 0.910929 | 0.672062 |

# Preprocessing During Training and Inference

## Normalization and one-hot encoding

- During training and prior to inference, continuous features are normalized using the MinMax Normalizer artifact
- To be properly fed into the model, categorical features must be transformed using “one-hot” encoding so that each distinct categorical value becomes a column where the rows are either zero or one. It’s important that all one-hot encoding reflect the categorical values and order that are available in the *training* set. For example, if a categorical value is found in the validation set that is not found in the training set, it cannot be included in the transformed one-hot encoded matrix.

| pickup_census_tract |
|---|
| 9999 |
| 17031031400 |
| 17031063100 |
| 17031062800 |
| 17031062000 |

| pickup_census_tract_17031031400 | pickup_census_tract_17031062000 | pickup_census_tract_17031062800 | pickup_census_tract_17031063100 | pickup_census_tract_9999 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |

# Preprocessing Data Pipeline

## One-time and Recurring steps

**Create Datasets**

**Train, Validation, Test**

Includes feature engineering such as calculating distance and logarithm conversion of the target attribute

**Fit Scaler**

**MinMax Normalizer**

Scaler fit across all continuous features in the training set

**Occurs once prior to training**

**Transform Features**

**Scaling, One-hot encoding**

Rows from the training set are transformed in batches during training

**Train**

**Features passed to model**

Transformed features are now prepared to be passed into the model for training

**Iterative during training**

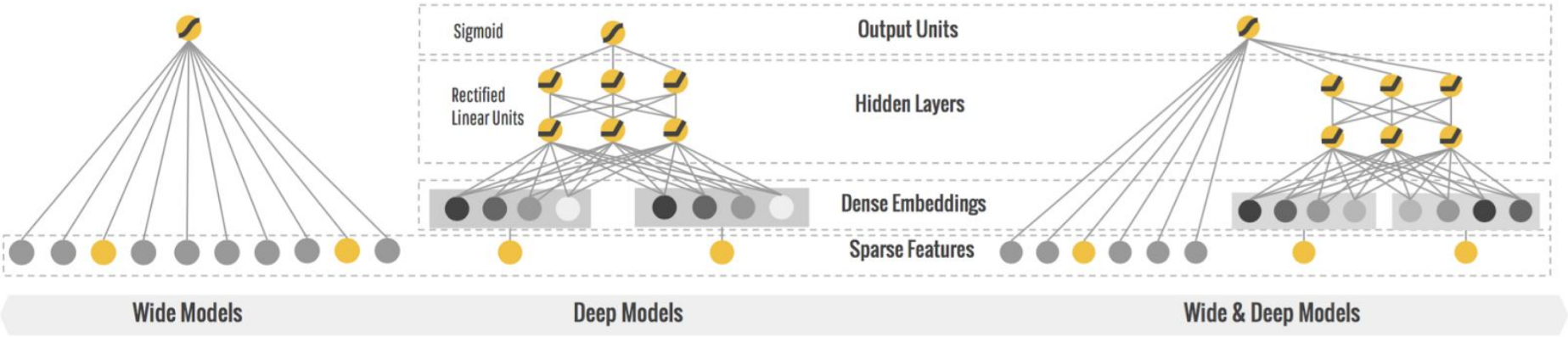
# Modeling

Wide and Deep Neural Network

- Wide layers to encode the categorical features
- Deep layers to learn the continuous features
- Architecture optimized with hyperparameter tuning
- Training and Hosting on GCP

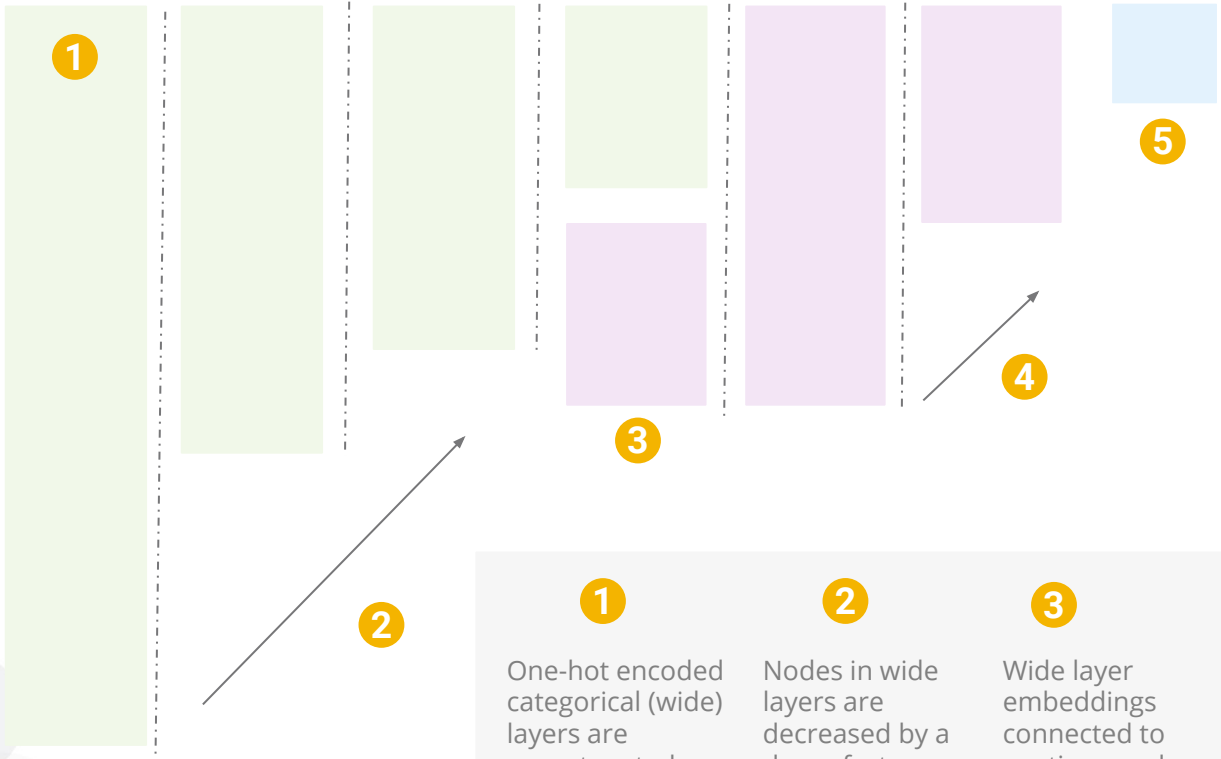
# Wide and Deep Neural Networks

- By jointly training a wide linear model (for memorization of categorical features) alongside a deep neural network (for generalization of continuous features), we can combine the strengths of both to make better predictions
- Wide and deep model architectures are useful for generic large-scale regression and classification problems with sparse inputs (categorical features with a large number of possible feature values), such as we have in this dataset with the pickup and dropoff census tract and community area fields.




source: https://ai.googleblog.com/2016/06/wide-deep-learning-better-together-with.html

# Network Architecture




1. One-hot encoded categorical (wide) layers are concatenated
2. Nodes in wide layers are decreased by a decay factor
3. Wide layer embeddings connected to continuous deep features
4. Nodes in deep layers decreased until number of deep layers is met
5. Single node linear prediction layer

# Hyperparameter Tuning

## Optimizing Model Performance

- Our model can be optimized by tuning hyperparameters at the model level such as the learning rate, as well as those that impact the architecture of the model itself such as the number of nodes in the first wide layer and the dropout rate.
- GCP's AI Platform can be used to run trials that iterate through different configurations of the hyperparameters provided with the goal of minimizing or maximizing a target metric - in this case, we minimize the Mean Squared Error on the validation set.

| parameter | data type | range | scaling method |
|---|---|---|---|
| train-batch-size | INTEGER | (8, 256) | UNIT_LINEAR_SCALE |
| learning-rate | DOUBLE | (0.001, 0.05) | UNIT_LOG_SCALE |
| num-deep-layers | INTEGER | (1, 2, 3, 4) | DISCRETE |
| first-deep-layer-size | INTEGER | (20, 30, 40, 50) | DISCRETE |
| first-wide-layer-size | INTEGER | (700, 5000) | UNIT_LINEAR_SCALE |
| wide-scale-factor | DOUBLE | (0.001, 0.5) | UNIT_LOG_SCALE |
| dropout-rate | DOUBLE | (0.01, 0.2, 0.3, 0.4, 0.5) | DISCRETE |

**HyperTune trials**

| | Trial ID | val_mse ↑ | Training step | train-batch-size | learning-rate | num-deep-layers | first-deep-layer-size | first-wide-laye |
|---|---|---|---|---|---|---|---|---|
| ○ | 31 | 0.185 | 40 | 132 | 0.003 | 2 | 30 | 1,233 |
| ○ | 39 | 0.19 | 40 | 59 | 0.024 | 2 | 20 | 2,028 |
| ○ | 13 | 0.193 | 40 | 229 | 0.023 | 3 | 20 | 3,376 |
| ○ | 43 | 0.198 | 49 | 155 | 0.007 | 3 | 50 | 2,333 |
| ○ | 5 | 0.202 | 37 | 194 | 0.014 | 1 | 30 | 3,142 |
| ○ | 9 | 0.203 | 49 | 11 | 0.005 | 3 | 40 | 4,488 |
| ○ | 28 | 0.203 | 49 | 168 | 0.004 | 3 | 50 | 1,162 |

# Modeling on Google Cloud Platform

## Traceability and Scale with AI Platform

- Training, hyperparameter, and hosting all are completed using AI Platform on Google Cloud Platform
- Access to the models is secured and controlled using IAM roles
- The roles described below define which users have access to which AI Platform resources

| Role Title | Role Name | Capabilities |
|---|---|---|
| AI Platform Admin | `roles/ml.admin` | Full control of AI Platform project, and its jobs, operations, models, and versions.<br><br>**Note:** The primitive project **Editor** role is equivalent to **`roles/ml.admin`**. |
| AI Platform Developer | `roles/ml.developer` | Create training and prediction jobs, models and versions, and send online prediction requests. |
| AI Platform Viewer | `roles/ml.viewer` | Read-only access to AI Platform resources. |

source: https://cloud.google.com/ml-engine/docs/access-control

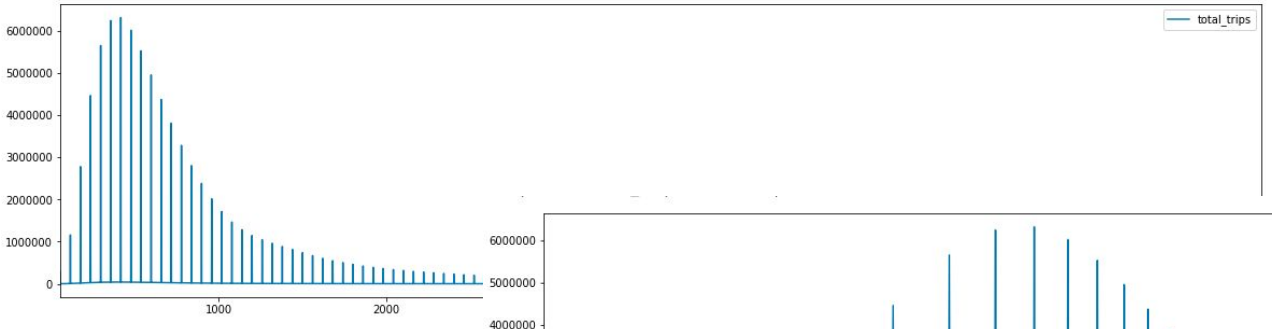
# Evaluation

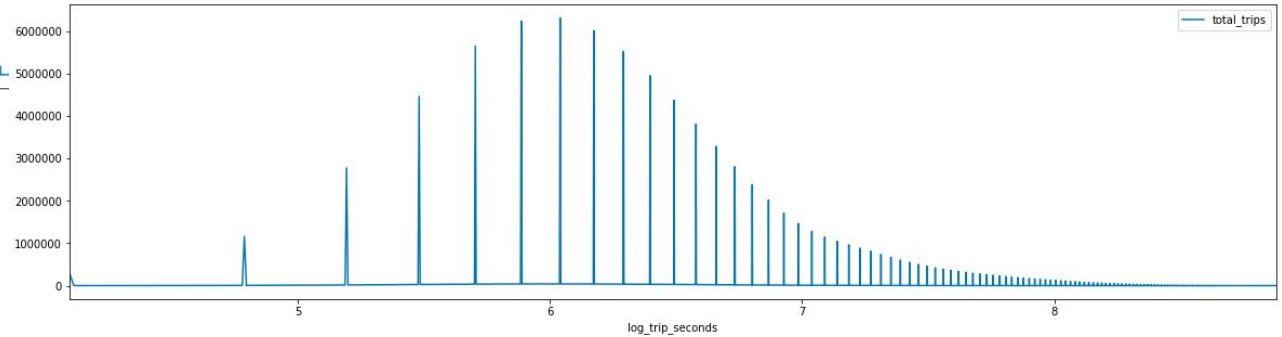
- Target Variable Transformations
- Evaluation metrics and results
- Suggestions for improvement

# Impact of Target Variable Transformation

## Training Metrics vs Evaluation Metrics

- As a reminder, we transformed the target attribute (trip duration in seconds) prior to training by taking the natural log of the actual duration to give it a more normal distribution.
- As a result, all metrics generated during training are also associated with the log transform of trip duration
- To assess metrics associated with the actual trip duration in seconds, we'll need to first make predictions for our test set using the trained model



# Choosing an Evaluation Metric

## Finding the Right Performance Metric for your Use Case

### Accuracy

Accuracy is the most commonly considered and well understood evaluation metric - how often are we right? However, accuracy is not a great metric for continuous the target variables like we have in prediction because we won't often be exactly right. Fortunately there are additional metrics can help us understand how close our metrics are to the ground truth.

### Root Mean Squared Error (RMSE)

RMSE is a common evaluation metric in prediction problems because it has the benefit of being in the same units as the target variable. For example we might say that, on average, our predictions are within "X" seconds of the actual trip duration. As we've, some trips on our dataset are much longer than others. Being off by 5 seconds on a trip that is 90 minutes is not the same as if the trip were 3 minutes.

### Mean Absolute Percentage Error (MAPE)

Fortunately there are metrics that can help us account for these differences in scale. The Mean Absolute Percentage error tells us, on average, by what percentage our predictions were off the mark.

### Mean Absolute Percentage Error

#### Samples

- 5000 trips from the "unseen" test set were used to approximate a MAPE value for our model

#### Methodology

- Each test sample was passed through preprocessing and then through our trained model so that the prediction could be made and compared against the actual value

#### Results

- For the subset in question, our predictions were, on average, within **41%** of the actual trip duration in seconds.

# Recommendations for Improvement

## Optimizing Model Performance is Ongoing

- Additional / third party data sources can be included to add more predictive power to the model. Some examples that may be helpful in this scenario are:
  - Weather data
  - Events data
  - Google Distance Matrix API ;)

- Further exploration with feature selection and feature engineering
  - Does a model work better with only the census tract and not the community area detail?
  - Are there “cross features” - such as a combined field representing the census tract pickup - to - census tract dropoff that get better performance than considering these fields independently?

- Continued hyperparameter and model architecture tuning. Note that the range of hyperparameters included in our hyperparameter search were limited to those we passed in using the configuration. It’s likely that there are additional model architectures and / or parameters that would improve model performance.