Advanced Feature Engineering in Keras

Learning Objectives

1. Process temporal feature columns in Keras
2. Use Lambda layers to perform feature engineering on geolocation features
3. Create bucketized and crossed feature columns

Introduction

In this notebook, we use Keras to build a taxifare price prediction model and utilize feature engineering to improve the fare amount prediction for NYC taxi cab rides.

Each learning objective will correspond to a **#TODO** in this student lab notebook -- try to complete this notebook first and then review the Solution Notebook for reference.

Set up environment variables and load necessary libraries

We will start by importing the necessary libraries for this lab.

In [1]:
```
!sudo chown -R jupyter:jupyter /home/jupyter/training-data-analyst
```

In [2]:
```
# Ensure the right version of Tensorflow is installed.
!pip freeze | grep tensorflow==2.1 || pip install --user tensorflow==2.1
```

```
Collecting tensorflow==2.1
  Downloading tensorflow-2.1.0-cp37-cp37m-manylinux2010_x86_64.whl (421.8 MB)
     |████████████████████████████████| 421.8 MB 10 kB/s s eta 0:00:01�██        | 331.
8 MB 63.8 MB/s eta 0:00:02
Requirement already satisfied: google-pasta>=0.1.6 in /opt/conda/lib/python3.7/site-pack
ages (from tensorflow==2.1) (0.2.0)
Requirement already satisfied: absl-py>=0.7.0 in /opt/conda/lib/python3.7/site-packages
(from tensorflow==2.1) (0.12.0)
Collecting scipy==1.4.1
  Downloading scipy-1.4.1-cp37-cp37m-manylinux1_x86_64.whl (26.1 MB)
     |████████████████████████████████| 26.1 MB 46.3 MB/s eta 0:00:01
Requirement already satisfied: numpy<2.0,>=1.16.0 in /opt/conda/lib/python3.7/site-packa
ges (from tensorflow==2.1) (1.19.5)
Collecting tensorflow-estimator<2.2.0,>=2.1.0rc0
  Downloading tensorflow_estimator-2.1.0-py2.py3-none-any.whl (448 kB)
     |████████████████████████████████| 448 kB 43.5 MB/s eta 0:00:01
Requirement already satisfied: wheel>=0.26 in /opt/conda/lib/python3.7/site-packages (fr
om tensorflow==2.1) (0.36.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /opt/conda/lib/python3.7/site-packag
es (from tensorflow==2.1) (3.3.0)
Collecting keras-applications>=1.0.8
  Downloading Keras_Applications-1.0.8-py3-none-any.whl (50 kB)
     |████████████████████████████████| 50 kB 7.5 MB/s  eta 0:00:01
Requirement already satisfied: protobuf>=3.8.0 in /opt/conda/lib/python3.7/site-packages
(from tensorflow==2.1) (3.16.0)
```

```
Collecting astor>=0.6.0
  Downloading astor-0.8.1-py2.py3-none-any.whl (27 kB)
Collecting tensorboard<2.2.0,>=2.1.0
  Downloading tensorboard-2.1.1-py3-none-any.whl (3.8 MB)
     |████████████████████████████████| 3.8 MB 52.7 MB/s eta 0:00:01
Requirement already satisfied: six>=1.12.0 in /opt/conda/lib/python3.7/site-packages (fr
om tensorflow==2.1) (1.16.0)
Requirement already satisfied: grpcio>=1.8.6 in /opt/conda/lib/python3.7/site-packages
(from tensorflow==2.1) (1.37.1)
Requirement already satisfied: keras-preprocessing>=1.1.0 in /opt/conda/lib/python3.7/si
te-packages (from tensorflow==2.1) (1.1.2)
Requirement already satisfied: wrapt>=1.11.1 in /opt/conda/lib/python3.7/site-packages
(from tensorflow==2.1) (1.12.1)
Requirement already satisfied: termcolor>=1.1.0 in /opt/conda/lib/python3.7/site-package
s (from tensorflow==2.1) (1.1.0)
Collecting gast==0.2.2
  Downloading gast-0.2.2.tar.gz (10 kB)
Requirement already satisfied: h5py in /opt/conda/lib/python3.7/site-packages (from kera
s-applications>=1.0.8->tensorflow==2.1) (3.1.0)
Requirement already satisfied: werkzeug>=0.11.15 in /opt/conda/lib/python3.7/site-packag
es (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (2.0.0)
Requirement already satisfied: google-auth<2,>=1.6.3 in /opt/conda/lib/python3.7/site-pa
ckages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (1.30.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /opt/conda/lib/python
3.7/site-packages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (0.4.4)
Requirement already satisfied: requests<3,>=2.21.0 in /opt/conda/lib/python3.7/site-pack
ages (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (2.25.1)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/lib/python3.7/site-packages
(from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (3.3.4)
Requirement already satisfied: setuptools>=41.0.0 in /opt/conda/lib/python3.7/site-packa
ges (from tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (49.6.0.post20210108)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /opt/conda/lib/python3.7/site-pa
ckages (from google-auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (0.2.7)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in /opt/conda/lib/python3.7/site-p
ackages (from google-auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (4.2.2)
Requirement already satisfied: rsa<5,>=3.1.4 in /opt/conda/lib/python3.7/site-packages
(from google-auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (4.7.2)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /opt/conda/lib/python3.7/site
-packages (from google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.2.0,>=2.1.0->tensorflow=
=2.1) (1.3.0)
Requirement already satisfied: importlib-metadata in /opt/conda/lib/python3.7/site-packa
ges (from markdown>=2.6.8->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (4.0.1)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in /opt/conda/lib/python3.7/site-pac
kages (from pyasn1-modules>=0.2.1->google-auth<2,>=1.6.3->tensorboard<2.2.0,>=2.1.0->ten
sorflow==2.1) (0.4.8)
Requirement already satisfied: chardet<5,>=3.0.2 in /opt/conda/lib/python3.7/site-packag
es (from requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /opt/conda/lib/python3.7/site-packages (f
rom requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packa
ges (from requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (2020.12.5)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-pa
ckages (from requests<3,>=2.21.0->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (1.26.4)
Requirement already satisfied: oauthlib>=3.0.0 in /opt/conda/lib/python3.7/site-packages
(from requests-oauthlib>=0.7.0->google-auth-oauthlib<0.5,>=0.4.1->tensorboard<2.2.0,>=2.
1.0->tensorflow==2.1) (3.0.1)
Requirement already satisfied: cached-property in /opt/conda/lib/python3.7/site-packages
(from h5py->keras-applications>=1.0.8->tensorflow==2.1) (1.5.2)
Requirement already satisfied: typing-extensions>=3.6.4 in /opt/conda/lib/python3.7/site
-packages (from importlib-metadata->markdown>=2.6.8->tensorboard<2.2.0,>=2.1.0->tensorfl
ow==2.1) (3.7.4.3)
Requirement already satisfied: zipp>=0.5 in /opt/conda/lib/python3.7/site-packages (from
importlib-metadata->markdown>=2.6.8->tensorboard<2.2.0,>=2.1.0->tensorflow==2.1) (3.4.1)
Building wheels for collected packages: gast
  Building wheel for gast (setup.py) ... done
```

```
  Created wheel for gast: filename=gast-0.2.2-py3-none-any.whl size=7538 sha256=fd516d48
4b6df48f724ea01b27f927613e4ce642a02ee310e530139d7bc81690
  Stored in directory: /home/jupyter/.cache/pip/wheels/21/7f/02/420f32a803f7d0967b48dd82
3da3f558c5166991bfd204eef3
Successfully built gast
Installing collected packages: tensorflow-estimator, tensorboard, scipy, keras-applicati
ons, gast, astor, tensorflow
  WARNING: The script tensorboard is installed in '/home/jupyter/.local/bin' which is no
t on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use
--no-warn-script-location.
  WARNING: The scripts estimator_ckpt_converter, saved_model_cli, tensorboard, tf_upgrad
e_v2, tflite_convert, toco and toco_from_protos are installed in '/home/jupyter/.local/b
in' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use
--no-warn-script-location.
ERROR: pip's dependency resolver does not currently take into account all the packages t
hat are installed. This behaviour is the source of the following dependency conflicts.
tfx-bsl 0.30.0 requires google-api-python-client<2,>=1.7.11, but you have google-api-pyt
hon-client 2.4.0 which is incompatible.
tfx-bsl 0.30.0 requires pyarrow<3,>=1, but you have pyarrow 4.0.0 which is incompatible.
tfx-bsl 0.30.0 requires tensorflow!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,<3,>=1.15.2, but you h
ave tensorflow 2.1.0 which is incompatible.
tensorflow-transform 0.30.0 requires pyarrow<3,>=1, but you have pyarrow 4.0.0 which is
 incompatible.
tensorflow-transform 0.30.0 requires tensorflow!=2.0.*,!=2.1.*,!=2.2.*,!=2.3.*,<2.5,>=1.
15.2, but you have tensorflow 2.1.0 which is incompatible.
tensorflow-serving-api 2.4.0 requires tensorflow<3,>=2.4.0, but you have tensorflow 2.1.
0 which is incompatible.
tensorflow-probability 0.12.2 requires gast>=0.3.2, but you have gast 0.2.2 which is inc
ompatible.
tensorflow-io 0.17.0 requires tensorflow<2.5.0,>=2.4.0, but you have tensorflow 2.1.0 wh
ich is incompatible.
tensorflow-cloud 0.1.14 requires tensorboard>=2.3.0, but you have tensorboard 2.1.1 whic
h is incompatible.
phik 0.11.2 requires scipy>=1.5.2, but you have scipy 1.4.1 which is incompatible.
keras 2.4.0 requires tensorflow>=2.2.0, but you have tensorflow 2.1.0 which is incompati
ble.
Successfully installed astor-0.8.1 gast-0.2.2 keras-applications-1.0.8 scipy-1.4.1 tenso
rboard-2.1.1 tensorflow-2.1.0 tensorflow-estimator-2.1.0
```

Note: After executing the above cell you will see the output `tensorflow==2.1.0` that is the installed version of tensorflow. You may ignore specific incompatibility errors and warnings.

Restart the kernel (click on the reload button above).

In [3]:

```python
import datetime
import logging
import os

import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf

from tensorflow import feature_column as fc
from tensorflow.keras import layers
from tensorflow.keras import models

# set TF error log verbosity
logging.getLogger("tensorflow").setLevel(logging.ERROR)

print(tf.version.VERSION)
```

```
2.5.0
```

Load taxifare dataset

The Taxi Fare dataset for this lab is 106,545 rows and has been pre-processed and split for use in this lab. Note that the dataset is the same as used in the Big Query feature engineering labs. The fare_amount is the target, the continuous value we'll train a model to predict.

First, let's download the .csv data by copying the data from a cloud storage bucket.

In [4]:

```
if not os.path.isdir("../data"):
    os.makedirs("../data")
```

In [5]:

```
!gsutil cp gs://cloud-training-demos/feat_eng/data/*.csv ../data
```

```
Copying gs://cloud-training-demos/feat_eng/data/churn_data.csv...
Copying gs://cloud-training-demos/feat_eng/data/customer_data.csv...
Copying gs://cloud-training-demos/feat_eng/data/internet_data.csv...
Copying gs://cloud-training-demos/feat_eng/data/taxi-test.csv...
- [4 files][  2.2 MiB/  2.2 MiB]
==> NOTE: You are performing a sequence of gsutil operations that may
run significantly faster if you instead use gsutil -m cp ... Please
see the -m section under "gsutil help options" for further information
about when gsutil -m can be advantageous.

Copying gs://cloud-training-demos/feat_eng/data/taxi-train.csv...
Copying gs://cloud-training-demos/feat_eng/data/taxi-valid.csv...
Copying gs://cloud-training-demos/feat_eng/data/telco_customer_churn.csv...
- [7 files][  7.3 MiB/  7.3 MiB]
Operation completed over 7 objects/7.3 MiB.
```

Let's check that the files were copied correctly and look like we expect them to.

In [6]:

```
!ls -l ../data/*.csv
```

```
-rw-r--r-- 1 jupyter jupyter  491383 May 24 21:29 ../data/churn_data.csv
-rw-r--r-- 1 jupyter jupyter  188590 May 24 21:29 ../data/customer_data.csv
-rw-r--r-- 1 jupyter jupyter  466434 May 24 21:29 ../data/internet_data.csv
-rw-r--r-- 1 jupyter jupyter 1113292 May 24 21:29 ../data/taxi-test.csv
-rw-r--r-- 1 jupyter jupyter 3551735 May 24 21:29 ../data/taxi-train.csv
-rw-r--r-- 1 jupyter jupyter  888648 May 24 21:29 ../data/taxi-valid.csv
-rw-r--r-- 1 jupyter jupyter  977501 May 24 21:29 ../data/telco_customer_churn.csv
```

In [7]:

```
!head ../data/*.csv
```

```
==> ../data/churn_data.csv <==
customerID,tenure,PhoneService,Contract,PaperlessBilling,PaymentMethod,MonthlyCharges,To
talCharges,Churn
7590-VHVEG,1,No,Month-to-month,Yes,Electronic check,29.85,29.85,No
5575-GNVDE,34,Yes,One year,No,Mailed check,56.95,1889.5,No
3668-QPYBK,2,Yes,Month-to-month,Yes,Mailed check,53.85,108.15,Yes
7795-CFOCW,45,No,One year,No,Bank transfer (automatic),42.3,1840.75,No
9237-HQITU,2,Yes,Month-to-month,Yes,Electronic check,70.7,151.65,Yes
9305-CDSKC,8,Yes,Month-to-month,Yes,Electronic check,99.65,820.5,Yes
1452-KIOVK,22,Yes,Month-to-month,Yes,Credit card (automatic),89.1,1949.4,No
6713-OKOMC,10,No,Month-to-month,No,Mailed check,29.75,301.9,No
7892-POOKP,28,Yes,Month-to-month,Yes,Electronic check,104.8,3046.05,Yes
```

```
==> ../data/customer_data.csv <==
customerID,gender,SeniorCitizen,Partner,Dependents
7590-VHVEG,Female,0,Yes,No
5575-GNVDE,Male,0,No,No
3668-QPYBK,Male,0,No,No
7795-CFOCW,Male,0,No,No
9237-HQITU,Female,0,No,No
9305-CDSKC,Female,0,No,No
1452-KIOVK,Male,0,No,Yes
6713-OKOMC,Female,0,No,No
7892-POOKP,Female,0,Yes,No

==> ../data/internet_data.csv <==
customerID,MultipleLines,InternetService,OnlineSecurity,OnlineBackup,DeviceProtection,Te
chSupport,StreamingTV,StreamingMovies
7590-VHVEG,No phone service,DSL,No,Yes,No,No,No,No
5575-GNVDE,No,DSL,Yes,No,Yes,No,No,No
3668-QPYBK,No,DSL,Yes,Yes,No,No,No,No
7795-CFOCW,No phone service,DSL,Yes,No,Yes,Yes,No,No
9237-HQITU,No,Fiber optic,No,No,No,No,No,No
9305-CDSKC,Yes,Fiber optic,No,No,Yes,No,Yes,Yes
1452-KIOVK,Yes,Fiber optic,No,Yes,No,No,Yes,No
6713-OKOMC,No phone service,DSL,Yes,No,No,No,No,No
7892-POOKP,Yes,Fiber optic,No,No,Yes,Yes,Yes,Yes

==> ../data/taxi-test.csv <==
fare_amount,passenger_count,pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_l
atitude,hourofday,dayofweek
7.7,4,-73.987998,40.764815000000006,-73.980602,40.744547999999995,15,3
11.5,1,-73.99378,40.75575,-73.97917,40.72579,22,4
7.3,1,-73.97940899999999,40.781647,-73.955749,40.772529999999996,12,6
8.5,1,-73.98869300000001,40.727032,-73.998685,40.73437,21,3
6.5,1,-73.947773,40.790307,-73.95327900000001,40.778389000000004,23,2
4.9,2,-74.000364,40.728729,-74.00884599999999,40.725924,11,1
3.7,3,-73.98424,40.755512,-73.990392,40.752039,7,3
4.9,4,-73.989798,40.762527,-73.989848,40.773913,16,2
11.7,2,-73.990573,40.728769,-73.938254,40.720831,8,6

==> ../data/taxi-train.csv <==
fare_amount,passenger_count,pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_l
atitude,hourofday,dayofweek
8.1,1,-73.973731,40.791909999999994,-73.962737,40.767317999999996,14,4
4.5,2,-73.98649499999999,40.739278000000006,-73.986083,40.730933,10,6
2.9,1,-73.95604300000001,40.772026000000004,-73.956245,40.773934000000004,22,3
7.0,1,-74.006557,40.705797,-73.980017,40.713617,6,3
6.5,1,-73.98644300000001,40.741611999999996,-73.990215,40.746466999999996,10,2
15.0,1,-73.96014404,40.7789917,-73.98536682,40.73873138,17,6
5.5,1,-73.981625,40.74957,-73.97639000000001,40.754807,18,0
9.0,3,-73.99884,40.734719,-73.978865,40.72422,21,1
14.0,2,-73.986827,40.742839000000004,-73.94695899999999,40.780063,19,1

==> ../data/taxi-valid.csv <==
fare_amount,passenger_count,pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_l
atitude,hourofday,dayofweek
15.5,3,-73.98902199999999,40.718837,-73.974645,40.761427000000005,18,3
7.5,1,-73.97336578,40.76422882,-73.98596954,40.75464249,20,0
3.3,2,-73.961372,40.760443,-73.956565,40.767154999999995,2,6
13.5,6,-73.989437,40.757082000000004,-73.98344200000001,40.725312,18,4
7.5,2,-73.981992,40.740312,-73.965375,40.752895,13,4
7.3,1,-73.99553,40.759465000000006,-73.978126,40.752683000000005,20,2
13.5,5,-73.99021149,40.75678253,-73.95141602,40.7700119,7,3
6.1,1,-74.005032,40.72971,-73.985805,40.722916999999995,19,4
10.0,6,-73.95796800000001,40.765155,-73.960672,40.781222,12,1
```

```
==> ../data/telco_customer_churn.csv <==
customerID,gender,SeniorCitizen,Partner,Dependents,tenure,PhoneService,MultipleLines,Int
ernetService,OnlineSecurity,OnlineBackup,DeviceProtection,TechSupport,StreamingTV,Stream
ingMovies,Contract,PaperlessBilling,PaymentMethod,MonthlyCharges,TotalCharges,Churn
7590-VHVEG,Female,0,Yes,No,1,No,No phone service,DSL,No,Yes,No,No,No,No,Month-to-month,Y
es,Electronic check,29.85,29.85,No
5575-GNVDE,Male,0,No,No,34,Yes,No,DSL,Yes,No,Yes,No,No,No,One year,No,Mailed check,56.9
5,1889.5,No
3668-QPYBK,Male,0,No,No,2,Yes,No,DSL,Yes,Yes,No,No,No,No,Month-to-month,Yes,Mailed chec
k,53.85,108.15,Yes
7795-CFOCW,Male,0,No,No,45,No,No phone service,DSL,Yes,No,Yes,Yes,No,No,One year,No,Bank
transfer (automatic),42.3,1840.75,No
9237-HQITU,Female,0,No,No,2,Yes,No,Fiber optic,No,No,No,No,No,No,Month-to-month,Yes,Elec
tronic check,70.7,151.65,Yes
9305-CDSKC,Female,0,No,No,8,Yes,Yes,Fiber optic,No,No,Yes,No,Yes,Yes,Month-to-month,Yes,
Electronic check,99.65,820.5,Yes
1452-KIOVK,Male,0,No,Yes,22,Yes,Yes,Fiber optic,No,Yes,No,No,Yes,No,Month-to-month,Yes,C
redit card (automatic),89.1,1949.4,No
6713-OKOMC,Female,0,No,No,10,No,No phone service,DSL,Yes,No,No,No,No,No,Month-to-month,N
o,Mailed check,29.75,301.9,No
7892-POOKP,Female,0,Yes,No,28,Yes,Yes,Fiber optic,No,No,Yes,Yes,Yes,Yes,Month-to-month,Y
es,Electronic check,104.8,3046.05,Yes
```

Create an input pipeline

Typically, you will use a two step process to build the pipeline. Step one is to define the columns of data; i.e., which column we're predicting for, and the default values. Step 2 is to define two functions - a function to define the features and label you want to use and a function to load the training data. Also, note that pickup_datetime is a string and we will need to handle this in our feature engineered model.

In [12]:

```
CSV_COLUMNS = [
    'fare_amount',
    'pickup_datetime',
    'pickup_longitude',
    'pickup_latitude',
    'dropoff_longitude',
    'dropoff_latitude',
    'passenger_count',
    'key',
]
LABEL_COLUMN = 'fare_amount'
STRING_COLS = ['pickup_datetime']
NUMERIC_COLS = ['pickup_longitude', 'pickup_latitude',
                'dropoff_longitude', 'dropoff_latitude',
                'passenger_count']
DEFAULTS = [[0.0], ['na'], [0.0], [0.0], [0.0], [0.0], [0.0], ['na']]
DAYS = ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']
```

In [13]:

```
# A function to define features and labesl
def features_and_labels(row_data):
    for unwanted_col in ['key']:
        row_data.pop(unwanted_col)
    label = row_data.pop(LABEL_COLUMN)
    return row_data, label
```

```
# A utility method to create a tf.data dataset from a Pandas Dataframe
def load_dataset(pattern, batch_size=1, mode='eval'):
    dataset = tf.data.experimental.make_csv_dataset(pattern,
                                                    batch_size,
                                                    CSV_COLUMNS,
                                                    DEFAULTS)
    dataset = dataset.map(features_and_labels)  # features, label
    if mode == 'train':
        dataset = dataset.shuffle(1000).repeat()
        # take advantage of multi-threading; 1=AUTOTUNE
        dataset = dataset.prefetch(1)
    return dataset
```

Create a Baseline DNN Model in Keras

Now let's build the Deep Neural Network (DNN) model in Keras using the functional API. Unlike the sequential API, we will need to specify the input and hidden layers. Note that we are creating a linear regression baseline model with no feature engineering. Recall that a baseline model is a solution to a problem without applying any machine learning techniques.

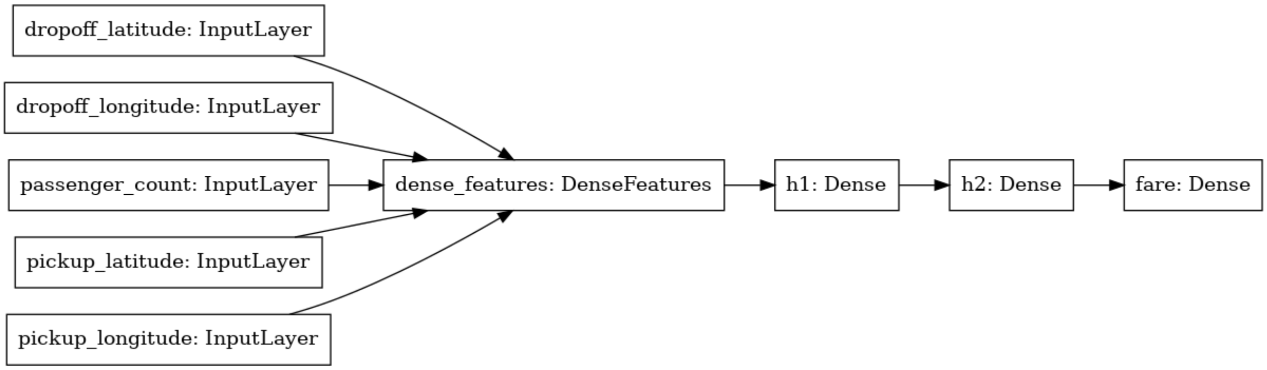
In [15]:

```
# Build a simple Keras DNN using its Functional API
def rmse(y_true, y_pred):  # Root mean square error
    return tf.sqrt(tf.reduce_mean(tf.square(y_pred - y_true)))


def build_dnn_model():
    # input layer
    inputs = {
        colname: layers.Input(name=colname, shape=(), dtype='float32')
        for colname in NUMERIC_COLS
    }

    # feature_columns
    feature_columns = {
        colname: fc.numeric_column(colname)
        for colname in NUMERIC_COLS
    }

    # Constructor for DenseFeatures takes a list of numeric columns
    dnn_inputs = layers.DenseFeatures(feature_columns.values())(inputs)

    # two hidden layers of [32, 8] just in like the BQML DNN
    h1 = layers.Dense(32, activation='relu', name='h1')(dnn_inputs)
    h2 = layers.Dense(8, activation='relu', name='h2')(h1)

    # final output is a linear activation because this is regression
    output = layers.Dense(1, activation='linear', name='fare')(h2)
    model = models.Model(inputs, output)

    # compile model
    model.compile(optimizer='adam', loss='mse', metrics=[rmse, 'mse'])

    return model
```

We'll build our DNN model and inspect the model architecture.

In [16]:
```
model = build_dnn_model()

tf.keras.utils.plot_model(model, 'dnn_model.png', show_shapes=False, rankdir='LR')
```

Out[16]:



Train the model

To train the model, simply call model.fit(). Note that we should really use many more NUM_TRAIN_EXAMPLES (i.e. a larger dataset). We shouldn't make assumptions about the quality of the model based on training/evaluating it on a small sample of the full data.

We start by setting up the environment variables for training, creating the input pipeline datasets, and then train our baseline DNN model.

In [17]:
```
TRAIN_BATCH_SIZE = 32
NUM_TRAIN_EXAMPLES = 59621 * 5
NUM_EVALS = 5
NUM_EVAL_EXAMPLES = 14906
```

In [18]:
```
trainds = load_dataset('../data/taxi-train*',
                       TRAIN_BATCH_SIZE,
                       'train')
evalds = load_dataset('../data/taxi-valid*',
                      1000,
                      'eval').take(NUM_EVAL_EXAMPLES//1000)

steps_per_epoch = NUM_TRAIN_EXAMPLES // (TRAIN_BATCH_SIZE * NUM_EVALS)

history = model.fit(trainds,
                    validation_data=evalds,
                    epochs=NUM_EVALS,
                    steps_per_epoch=steps_per_epoch)
```

```
Epoch 1/5
/opt/conda/lib/python3.7/site-packages/tensorflow/python/keras/engine/functional.py:591:
UserWarning: Input dict contained keys ['pickup_datetime'] which did not match any model
input. They will be ignored by the model.
  [n for n in tensors.keys() if n not in ref_input_names])
1863/1863 [==============================] - 15s 7ms/step - loss: 103.7294 - rmse: 9.581
9 - mse: 103.7294 - val_loss: 99.0964 - val_rmse: 9.9381 - val_mse: 99.0964
Epoch 2/5
1863/1863 [==============================] - 13s 7ms/step - loss: 101.3948 - rmse: 9.522
6 - mse: 101.3948 - val_loss: 102.7344 - val_rmse: 10.1013 - val_mse: 102.7344
Epoch 3/5
```

```
1863/1863 [==============================] - 13s 7ms/step - loss: 103.0851 - rmse: 9.547
9 - mse: 103.0851 - val_loss: 98.1772 - val_rmse: 9.8926 - val_mse: 98.1772
Epoch 4/5
1863/1863 [==============================] - 14s 7ms/step - loss: 100.2398 - rmse: 9.475
4 - mse: 100.2398 - val_loss: 100.6264 - val_rmse: 10.0169 - val_mse: 100.6264
Epoch 5/5
1863/1863 [==============================] - 14s 7ms/step - loss: 105.5943 - rmse: 9.678
6 - mse: 105.5943 - val_loss: 99.8934 - val_rmse: 9.9702 - val_mse: 99.8934
```

Visualize the model loss curve

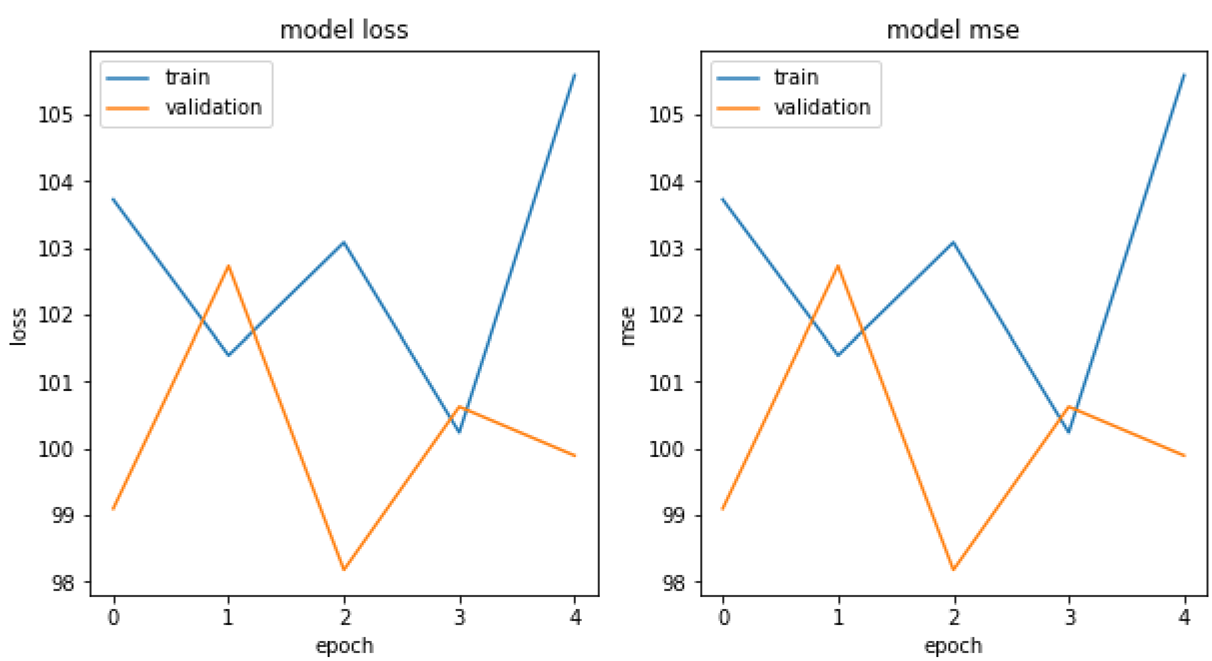
Next, we will use matplotlib to draw the model's loss curves for training and validation. A line plot is also created showing the mean squared error loss over the training epochs for both the train (blue) and test (orange) sets.

In [19]:

```python
def plot_curves(history, metrics):
    nrows = 1
    ncols = 2
    fig = plt.figure(figsize=(10, 5))

    for idx, key in enumerate(metrics):
        ax = fig.add_subplot(nrows, ncols, idx+1)
        plt.plot(history.history[key])
        plt.plot(history.history['val_{}'.format(key)])
        plt.title('model {}'.format(key))
        plt.ylabel(key)
        plt.xlabel('epoch')
        plt.legend(['train', 'validation'], loc='upper left');
```

In [20]:

```python
plot_curves(history, ['loss', 'mse'])
```



Predict with the model locally

To predict with Keras, you simply call model.predict() and pass in the cab ride you want to predict the fare amount for. Next we note the fare price at this geolocation and pickup_datetime.

In [21]:

```
model.predict({
    'pickup_longitude': tf.convert_to_tensor([-73.982683]),
    'pickup_latitude': tf.convert_to_tensor([40.742104]),
    'dropoff_longitude': tf.convert_to_tensor([-73.983766]),
    'dropoff_latitude': tf.convert_to_tensor([40.755174]),
    'passenger_count': tf.convert_to_tensor([3.0]),
    'pickup_datetime': tf.convert_to_tensor(['2010-02-08 09:17:00 UTC'], dtype=tf.strin
}, steps=1)
```

Out[21]:

```
array([[12.142072]], dtype=float32)
```

Improve Model Performance Using Feature Engineering

We now improve our model's performance by creating the following feature engineering types: Temporal, Categorical, and Geolocation.

Temporal Feature Columns

Lab Task #1: Processing temporal feature columns in Keras

We incorporate the temporal feature pickup_datetime. As noted earlier, pickup_datetime is a string and we will need to handle this within the model. First, you will include the pickup_datetime as a feature and then you will need to modify the model to handle our string feature.

In [22]:

```
# TODO 1a - Your code here

def parse_datetime(s):
    if type(s) is not str:
        s = s.numpy().decode('utf-8')
    return datetime.datetime.strptime(s, "%Y-%m-%d %H:%M:%S %Z")

# TODO 1b - Your code here

def get_dayofweek(s):
    ts = parse_datetime(s)
    return DAYS[ts.weekday()]

# TODO 1c - Your code here
@tf.function
def dayofweek(ts_in):
    return tf.map_fn(
        lambda s: tf.py_function(get_dayofweek, inp=[s], Tout=tf.string),
        ts_in)
```

Geolocation/Coordinate Feature Columns

The pick-up/drop-off longitude and latitude data are crucial to predicting the fare amount as fare amounts in NYC taxis are largely determined by the distance traveled. As such, we need to teach the model the Euclidean distance between the pick-up and drop-off points.

Recall that latitude and longitude allows us to specify any location on Earth using a set of coordinates. In our training data set, we restricted our data points to only pickups and drop offs within NYC. New York city has an approximate longitude range of -74.05 to -73.75 and a latitude range of 40.63 to 40.85.

Computing Euclidean distance

The dataset contains information regarding the pickup and drop off coordinates. However, there is no information regarding the distance between the pickup and drop off points. Therefore, we create a new feature that calculates the distance between each pair of pickup and drop off points. We can do this using the Euclidean Distance, which is the straight-line distance between any two coordinate points.

In [23]:

```
def euclidean(params):
    lon1, lat1, lon2, lat2 = params
    londiff = lon2 - lon1
    latdiff = lat2 - lat1
    return tf.sqrt(londiff*londiff + latdiff*latdiff)
```

Scaling latitude and longitude

It is very important for numerical variables to get scaled before they are "fed" into the neural network. Here we use min-max scaling (also called normalization) on the geolocation features. Later in our model, you will see that these values are shifted and rescaled so that they end up ranging from 0 to 1.

First, we create a function named 'scale_longitude', where we pass in all the longitudinal values and add 78 to each value. Note that our scaling longitude ranges from -70 to -78. Thus, the value 78 is the maximum longitudinal value. The delta or difference between -70 and -78 is 8. We add 78 to each longitudinal value and then divide by 8 to return a scaled value.

In [24]:

```
def scale_longitude(lon_column):
    return (lon_column + 78)/8.
```

Next, we create a function named 'scale_latitude', where we pass in all the latitudinal values and subtract 37 from each value. Note that our scaling latitude ranges from -37 to -45. Thus, the value 37 is the minimal latitudinal value. The delta or difference between -37 and -45 is 8. We subtract 37 from each latitudinal value and then divide by 8 to return a scaled value.

In [25]:

```
def scale_latitude(lat_column):
    return (lat_column - 37)/8.
```

Putting it all together

We will create a function called "euclidean" to initialize our geolocation parameters. We then create a function called transform. The transform function passes our numerical and string column features as inputs to the model, scales geolocation features, then creates the Euclidean distance as a transformed variable with the geolocation features. Lastly, we bucketize the latitude and longitude features.

Lab Task #2: We will use Lambda layers to create two new "geo" functions for our model.

Lab Task #3: Creating the bucketized and crossed feature columns

In [29]:
```
def transform(inputs, numeric_cols, string_cols, nbuckets):
    print("Inputs before features transformation: {}".format(inputs.keys()))

    # Pass-through columns
    transformed = inputs.copy()
    del transformed['pickup_datetime']

    feature_columns = {
        colname: tf.feature_column.numeric_column(colname)
        for colname in numeric_cols
    }

    # Scaling longitude from range [-70, -78] to [0, 1]
    # TODO 2a
    # TODO -- Your code here.
    for lon_col in ['pickup_longitude', 'dropoff_longitude']:
        transformed[lon_col] = layers.Lambda(
            scale_longitude,
            name="scale_{}".format(lon_col))(inputs[lon_col])
    # Scaling latitude from range [37, 45] to [0, 1]
    # TODO 2b
    # TODO -- Your code here.
    for lat_col in ['pickup_latitude', 'dropoff_latitude']:
        transformed[lat_col] = layers.Lambda(
            scale_latitude,
            name='scale_{}'.format(lat_col))(inputs[lat_col])
    # add Euclidean distance
    transformed['euclidean'] = layers.Lambda(
        euclidean,
        name='euclidean')([inputs['pickup_longitude'],
                           inputs['pickup_latitude'],
                           inputs['dropoff_longitude'],
                           inputs['dropoff_latitude']])
    feature_columns['euclidean'] = fc.numeric_column('euclidean')

    # TODO 3a
    # TODO -- Your code here.
    latbuckets = np.linspace(0, 1, nbuckets).tolist()
    lonbuckets = np.linspace(0, 1, nbuckets).tolist()
    b_plat = fc.bucketized_column(
        feature_columns['pickup_latitude'], latbuckets)
    b_dlat = fc.bucketized_column(
        feature_columns['dropoff_latitude'], latbuckets)
    b_plon = fc.bucketized_column(
        feature_columns['pickup_longitude'], lonbuckets)
    b_dlon = fc.bucketized_column(
        feature_columns['dropoff_longitude'], lonbuckets)
    # TODO 3b
    # TODO -- Your code here.
    ploc = fc.crossed_column([b_plat, b_plon], nbuckets * nbuckets)
    dloc = fc.crossed_column([b_dlat, b_dlon], nbuckets * nbuckets)
    pd_pair = fc.crossed_column([ploc, dloc], nbuckets ** 4)
    # create embedding columns
```

```
    feature_columns['pickup_and_dropoff'] = fc.embedding_column(pd_pair, 100)

    print("Transformed features: {}".format(transformed.keys()))
    print("Feature columns: {}".format(feature_columns.keys()))
    return transformed, feature_columns
```

In [ ]:

Next, we'll create our DNN model now with the engineered features. We'll set `NBUCKETS = 10` to specify 10 buckets when bucketizing the latitude and longitude.

In [31]:

```
NBUCKETS = 10


# DNN MODEL
def rmse(y_true, y_pred):
    return tf.sqrt(tf.reduce_mean(tf.square(y_pred - y_true)))


def build_dnn_model():
    # input layer is all float except for pickup_datetime which is a string
    inputs = {
        colname: layers.Input(name=colname, shape=(), dtype='float32')
        for colname in NUMERIC_COLS
    }
    inputs.update({
        colname: tf.keras.layers.Input(name=colname, shape=(), dtype='string')
        for colname in STRING_COLS
    })

    # transforms
    transformed, feature_columns = transform(inputs,
                                             numeric_cols=NUMERIC_COLS,
                                             string_cols=STRING_COLS,
                                             nbuckets=NBUCKETS)
    dnn_inputs = layers.DenseFeatures(feature_columns.values())(transformed)

    # two hidden layers of [32, 8] just in like the BQML DNN
    h1 = layers.Dense(32, activation='relu', name='h1')(dnn_inputs)
    h2 = layers.Dense(8, activation='relu', name='h2')(h1)

    # final output is a linear activation because this is regression
    output = layers.Dense(1, activation='linear', name='fare')(h2)
    model = models.Model(inputs, output)

    # Compile model
    model.compile(optimizer='adam', loss='mse', metrics=[rmse, 'mse'])
    return model
```

In [32]:

```
model = build_dnn_model()
```

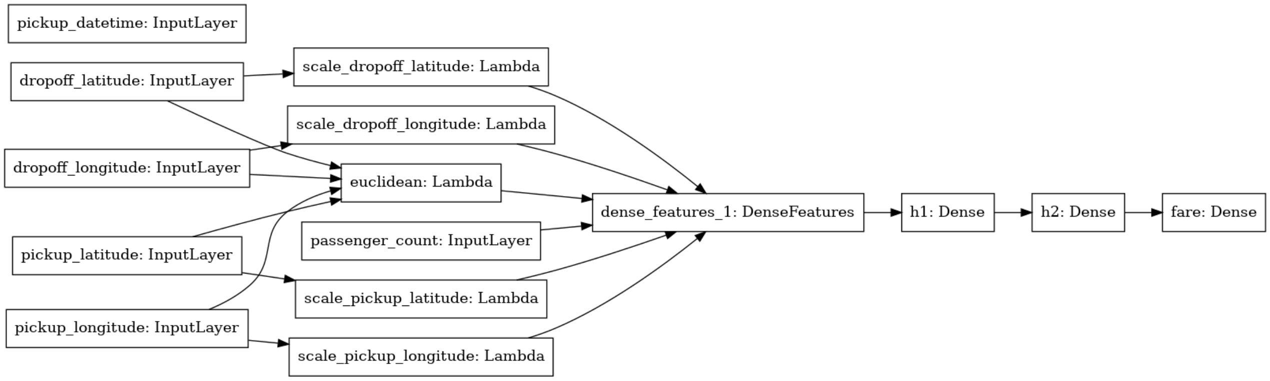
```
Inputs before features transformation: dict_keys(['pickup_longitude', 'pickup_latitude',
'dropoff_longitude', 'dropoff_latitude', 'passenger_count', 'pickup_datetime'])
Transformed features: dict_keys(['pickup_longitude', 'pickup_latitude', 'dropoff_longitu
de', 'dropoff_latitude', 'passenger_count', 'euclidean'])
Feature columns: dict_keys(['pickup_longitude', 'pickup_latitude', 'dropoff_longitude',
'dropoff_latitude', 'passenger_count', 'euclidean', 'pickup_and_dropoff'])
```

Let's see how our model architecture has changed now.

In [33]:

```
tf.keras.utils.plot_model(model, 'dnn_model_engineered.png', show_shapes=False, rankdir
```

Out[33]:



In [34]:

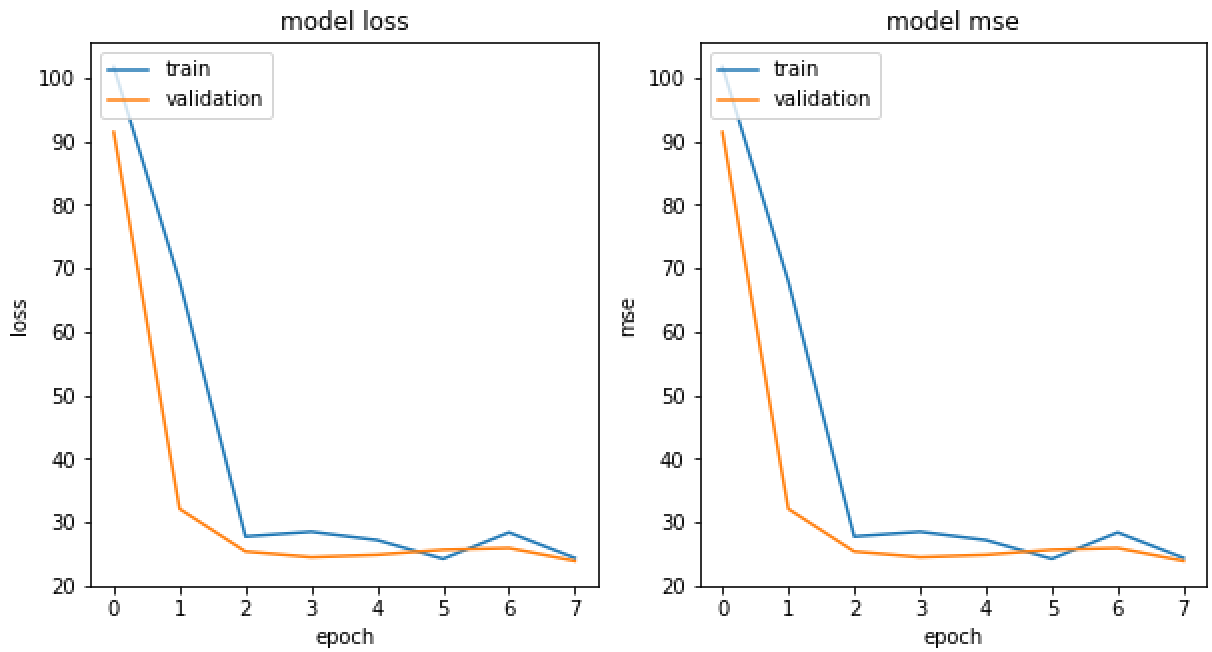
```
trainds = load_dataset('../data/taxi-train*',
                       TRAIN_BATCH_SIZE,
                       'train')
evalds = load_dataset('../data/taxi-valid*',
                      1000,
                      'eval').take(NUM_EVAL_EXAMPLES//1000)

steps_per_epoch = NUM_TRAIN_EXAMPLES // (TRAIN_BATCH_SIZE * NUM_EVALS)

history = model.fit(trainds,
                    validation_data=evalds,
                    epochs=NUM_EVALS+3,
                    steps_per_epoch=steps_per_epoch)
```

```
Epoch 1/8
1863/1863 [==============================] - 44s 23ms/step - loss: 101.7966 - rmse: 9.48
11 - mse: 101.7966 - val_loss: 91.4950 - val_rmse: 9.5509 - val_mse: 91.4950
Epoch 2/8
1863/1863 [==============================] - 42s 22ms/step - loss: 67.9857 - rmse: 7.531
3 - mse: 67.9857 - val_loss: 32.0920 - val_rmse: 5.6100 - val_mse: 32.0920
Epoch 3/8
1863/1863 [==============================] - 42s 22ms/step - loss: 27.7385 - rmse: 4.394
0 - mse: 27.7385 - val_loss: 25.3686 - val_rmse: 4.9718 - val_mse: 25.3686
Epoch 4/8
1863/1863 [==============================] - 41s 22ms/step - loss: 28.4843 - rmse: 4.356
7 - mse: 28.4843 - val_loss: 24.4854 - val_rmse: 4.8798 - val_mse: 24.4854
Epoch 5/8
1863/1863 [==============================] - 40s 22ms/step - loss: 27.1694 - rmse: 4.291
5 - mse: 27.1694 - val_loss: 24.8521 - val_rmse: 4.9453 - val_mse: 24.8521
Epoch 6/8
1863/1863 [==============================] - 40s 22ms/step - loss: 24.2229 - rmse: 4.209
6 - mse: 24.2229 - val_loss: 25.6447 - val_rmse: 4.9948 - val_mse: 25.6447
Epoch 7/8
1863/1863 [==============================] - 39s 21ms/step - loss: 28.3702 - rmse: 4.372
2 - mse: 28.3702 - val_loss: 25.9197 - val_rmse: 5.0171 - val_mse: 25.9197
Epoch 8/8
1863/1863 [==============================] - 42s 23ms/step - loss: 24.3563 - rmse: 4.157
6 - mse: 24.3563 - val_loss: 23.8904 - val_rmse: 4.8354 - val_mse: 23.8904
```

As before, let's visualize the DNN model layers.

In [35]:

```
plot_curves(history, ['loss', 'mse'])
```



Let's a prediction with this new model with engineered features on the example we had above.

In [36]:
```
model.predict({
    'pickup_longitude': tf.convert_to_tensor([-73.982683]),
    'pickup_latitude': tf.convert_to_tensor([40.742104]),
    'dropoff_longitude': tf.convert_to_tensor([-73.983766]),
    'dropoff_latitude': tf.convert_to_tensor([40.755174]),
    'passenger_count': tf.convert_to_tensor([3.0]),
    'pickup_datetime': tf.convert_to_tensor(['2010-02-08 09:17:00 UTC'], dtype=tf.strin
}, steps=1)
```

Out[36]:
```
array([[7.4384675]], dtype=float32)
```

Below we summarize our training results comparing our baseline model with our model with engineered features.

Model	Taxi Fare	Description
Baseline	value?	Baseline model - no feature engineering
Feature Engineered	value?	Feature Engineered Model

Copyright 2020 Google Inc. Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.