Simona Rahi

Loading Data and Classifying

```
import tensorflow as tf
from tensorflow import keras
import functools
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import metrics
```

Loading test and train data (each is 50% of data)

```
data_URL = "https://raw.githubusercontent.com/simonarahi/MachineLearning-TensorFlow/ma

train_data_URL = "https://raw.githubusercontent.com/simonarahi/MachineLearning-TensorF
test_data_URL = "https://raw.githubusercontent.com/simonarahi/MachineLearning-TensorFl

train_file_path = tf.keras.utils.get_file("train.csv", train_data_URL)
test_file_path = tf.keras.utils.get_file("test.csv", test_data_URL)
```

```
Downloading data from https://raw.githubusercontent.com/simonarahi/MachineLearnin
1425408/1419730 [==============================] - 0s 0us/step
Downloading data from https://raw.githubusercontent.com/simonarahi/MachineLearnin
1425408/1420197 [==============================] - 0s 0us/step
```

```
np.set_printoptions(precision=3, suppress=True)
```

Inspecting the data

```
df = pd.read_csv(train_file_path)
df.head()
#df['injSeverity'].min()
#df['injSeverity'].max()
```

Specifying column to be classified

```
LABEL_COLUMN = 'injSeverity'
LABELS = [0, 1, 2, 3, 4, 5, 6]
```

Read csv data from file and create dataset

```
def get_dataset(file_path, **kwargs):
  dataset = tf.data.experimental.make_csv_dataset(
      file_path,
      batch_size=5, # Artificially small to make examples easier to show.
      label_name=LABEL_COLUMN,
      na_value="?",
      select_columns = ['dvcat', 'weight', 'dead', 'airbag', 'seatbelt', 'frontal', 's
      num_epochs=1,
      ignore_errors=True,
      **kwargs)
  return dataset

raw_train_data = get_dataset(train_file_path)
raw_test_data = get_dataset(test_file_path)


def show_batch(dataset):
  for batch, label in dataset.take(1):
    for key, value in batch.items():
      print("{:20s}: {}".format(key,value.numpy()))

show_batch(raw_train_data)
```

Data Preprocessing

Since we have mixed data types, we will start by seperating the numeric features and pack them into c

```
class PackNumericFeatures(object):
  def __init__(self, names):
    self.names = names

  def __call__(self, features, labels):
    numeric_features = [features.pop(name) for name in self.names]
    numeric_features = [tf.cast(feat, tf.float32) for feat in numeric_features]
    numeric_features = tf.stack(numeric_features, axis=-1)
    features['numeric'] = numeric_features

    return features, labels


NUMERIC_FEATURES = ['weight','frontal','ageOFocc', 'yearacc', 'yearVeh', 'deploy']

packed_train_data = raw_train_data.map(
    PackNumericFeatures(NUMERIC_FEATURES))

packed_test_data = raw_test_data.map(
    PackNumericFeatures(NUMERIC_FEATURES))


show_batch(packed_train_data)
```

```
example_batch, labels_batch = next(iter(packed_train_data))
```

Normalizing our continuous data

```
import pandas as pd
desc = pd.read_csv(train_file_path)[NUMERIC_FEATURES].describe()
desc
```

```
MEAN = np.array(desc.T['mean'])
STD = np.array(desc.T['std'])

def normalize_numeric_data(data, mean, std):
  # Center the data
  return (data-mean)/std
```

Creating a numeric column with the normalized data

```
normalizer = functools.partial(normalize_numeric_data, mean=MEAN, std=STD)

numeric_column = tf.feature_column.numeric_column('numeric', normalizer_fn=normalizer,
numeric_columns = [numeric_column]
numeric_column
```

We will include this feature column 'numeric' in our training process

```
example_batch['numeric']
```

```
numeric_layer = tf.keras.layers.DenseFeatures(numeric_columns)
numeric_layer(example_batch).numpy()
```

Now dealing with our categorical variables

```
CATEGORIES = {
    'dvcat': ['1-9km/h', '10-24', '25-39', '40-54', '55+'],
    'dead' : ['alive', 'dead'],
    'airbag' : ['airbag', 'none'],
    'seatbelt' : ['belted', 'none'],
    'sex' : ['f', 'm'],
    'abcat' : ['deploy', 'nodeploy', 'unavail'],
    'occRole' : ['driver', 'pass']
}


categorical_columns = []
for feature, vocab in CATEGORIES.items():
  cat_col = tf.feature_column.categorical_column_with_vocabulary_list(
        key=feature, vocabulary_list=vocab)
  categorical_columns.append(tf.feature_column.indicator_column(cat_col))

categorical_columns
```

This layer below will be part of the data processing input layer in our model

```
categorical_layer = tf.keras.layers.DenseFeatures(categorical_columns)
print(categorical_layer(example_batch).numpy()[0])
```

Combined Preprocessing Layer: Adding the two feature column we created (continuous and categori extract and preprocess both input types

```
preprocessing_layer = tf.keras.layers.DenseFeatures(categorical_columns+numeric_columr
print(preprocessing_layer(example_batch).numpy()[0])
```

Build the Model

```
model = tf.keras.Sequential([
  preprocessing_layer,
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(1),
])

model.compile(
    loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True),
    optimizer='adam',
    metrics=['accuracy'])
```

now we can start training

```
train_data = packed_train_data.shuffle(500)
test_data = packed_test_data

model.fit(train_data, epochs=20)
```

Now that we trained our model, we can test the accuracy on the test set

```
test_loss, test_accuracy = model.evaluate(test_data)

print('\n\nTest Loss {}, Test Accuracy {}'.format(test_loss, test_accuracy))
```

Test accuracy is almost 21.8% which is very low

```
predictions = model.predict(test_data)
print(predictions)
```

Confusion Matrix

```
df1 = pd.read_csv(test_file_path)
cm = metrics.confusion_matrix(df1['injSeverity'], predictions)
print(cm)
tf.confusion_matrix
```