TensorflowProject

December 12, 2017

0.1 CLASSIFICATION EXAMPLE

We'll be working with some California Census Data, we'll be trying to use various features of an individual to predict what class of income they belong in (>50k or <=50k).

The Data:

```
In [8]: import pandas as pd
```

```
In [10]: census = pd.read_csv("C:/Users/DELL/Tensorflow_Project/census_data.csv")
```

```
In [11]: census.head()
```

Out[11]:

	age	workclass	education	education_num	marital_status	occupation	relationship	race	gender	capital_gain	capital_loss	hours_per_week	native_country	income_bracket
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

TensorFlow won't be able to understand strings as labels, so we will convert the Label column to 0s and 1s instead of strings.

```
In [12]: census['income_bracket'].unique()
```

```
Out[12]: array([' <=50K', ' >50K'], dtype=object)
```

```
In [13]: def label_fix(label):
             if label==' <=50K':
                 return 0
             else:
                 return 1
```

```
In [14]: census['income_bracket'] = census['income_bracket'].apply(label_fix)
```

```
In [15]: census['income_bracket'].unique()
```

```
Out[15]: array([0, 1], dtype=int64)
```

0.1.1 Perform a Train Test Split on the Data

```
In [16]: from sklearn.model_selection import train_test_split
```

```
In [17]: x_data = census.drop('income_bracket',axis=1)
         y_labels = census['income_bracket']
         X_train, X_test, y_train, y_test = train_test_split(x_data,y_labels,
                                                             test_size=0.3,
                                                             random_state=101)
```

0.1.2 Create the Feature Columns for tf.esitmator

Take note of categorical vs continuous values!

```
In [18]: census.columns
```

```
Out[18]: Index(['age', 'workclass', 'education', 'education_num', 'marital_status',
                'occupation', 'relationship', 'race', 'gender', 'capital_gain',
                'capital_loss', 'hours_per_week', 'native_country', 'income_bracket'],
               dtype='object')
```

Import Tensorflow

```
In [19]: import tensorflow as tf
```

Create the tf.feature_columns for the categorical values

```
In [20]: gender = tf.feature_column.categorical_column_with_vocabulary_list(
             "gender",["Female", "Male"])
         occupation = tf.feature_column.categorical_column_with_hash_bucket(
             "occupation",hash_bucket_size=1000)
         marital_status = tf.feature_column.categorical_column_with_hash_bucket(
             "marital_status", hash_bucket_size=1000)
         relationship = tf.feature_column.categorical_column_with_hash_bucket(
             "relationship", hash_bucket_size=1000)
         education = tf.feature_column.categorical_column_with_hash_bucket(
```

```
        "education", hash_bucket_size=1000)
    workclass = tf.feature_column.categorical_column_with_hash_bucket(
        "workclass", hash_bucket_size=1000)
    native_country = tf.feature_column.categorical_column_with_hash_bucket(
        "native_country", hash_bucket_size=1000)
```

Create the continuous feature_columns for the continuous values using numeric_column

```
In [21]: age = tf.feature_column.numeric_column("age")
         education_num = tf.feature_column.numeric_column("education_num")
         capital_gain = tf.feature_column.numeric_column("capital_gain")
         capital_loss = tf.feature_column.numeric_column("capital_loss")
         hours_per_week = tf.feature_column.numeric_column("hours_per_week")
```

Put all these variables into a single list with the variable name feat_cols

```
In [22]: feat_cols = [gender,occupation,marital_status,relationship,education,workclass,
                      native_country,age,education_num,capital_gain,capital_loss,
                      hours_per_week]
```

0.1.3 Create Input Function

```
In [23]: input_func = tf.estimator.inputs.pandas_input_fn(x=X_train,y=y_train,
                                                          batch_size=100,
                                                          num_epochs=None,
                                                          shuffle=True)
```

Create the model with tf.estimator (LinearClassifier)

```
In [24]: model = tf.estimator.LinearClassifier(feature_columns=feat_cols)
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\DELL\AppData\Local\Temp\t
INFO:tensorflow:Using config: {'_model_dir': 'C:\\Users\\DELL\\AppData\\Local\\Temp\\tmphd39etm0
```

Train the model on the Data, for 5000 steps

```
In [25]: model.train(input_fn = input_func,steps=5000)
```

```
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Saving checkpoints for 1 into C:\Users\DELL\AppData\Local\Temp\tmphd39etm0\model
INFO:tensorflow:loss = 69.3147, step = 1
INFO:tensorflow:global_step/sec: 168.08
INFO:tensorflow:loss = 343.425, step = 101 (0.599 sec)
INFO:tensorflow:global_step/sec: 281.338
INFO:tensorflow:loss = 113.153, step = 201 (0.355 sec)
INFO:tensorflow:global_step/sec: 254.424
INFO:tensorflow:loss = 78.6489, step = 301 (0.394 sec)
```

```
INFO:tensorflow:global_step/sec: 264.268
INFO:tensorflow:loss = 60.7092, step = 401 (0.378 sec)
INFO:tensorflow:global_step/sec: 285.362
INFO:tensorflow:loss = 56.0403, step = 501 (0.350 sec)
INFO:tensorflow:global_step/sec: 295.071
INFO:tensorflow:loss = 91.9415, step = 601 (0.339 sec)
INFO:tensorflow:global_step/sec: 300.854
INFO:tensorflow:loss = 27.7362, step = 701 (0.332 sec)
INFO:tensorflow:global_step/sec: 284.953
INFO:tensorflow:loss = 202.947, step = 801 (0.351 sec)
INFO:tensorflow:global_step/sec: 286.276
INFO:tensorflow:loss = 133.695, step = 901 (0.349 sec)
INFO:tensorflow:global_step/sec: 295.071
INFO:tensorflow:loss = 311.527, step = 1001 (0.339 sec)
INFO:tensorflow:global_step/sec: 271.284
INFO:tensorflow:loss = 56.3449, step = 1101 (0.369 sec)
INFO:tensorflow:global_step/sec: 250.588
INFO:tensorflow:loss = 47.9128, step = 1201 (0.401 sec)
INFO:tensorflow:global_step/sec: 242.957
INFO:tensorflow:loss = 34.5754, step = 1301 (0.414 sec)
INFO:tensorflow:global_step/sec: 287.418
INFO:tensorflow:loss = 51.5021, step = 1401 (0.344 sec)
INFO:tensorflow:global_step/sec: 300.401
INFO:tensorflow:loss = 56.6525, step = 1501 (0.333 sec)
INFO:tensorflow:global_step/sec: 299.484
INFO:tensorflow:loss = 129.215, step = 1601 (0.332 sec)
INFO:tensorflow:global_step/sec: 297.269
INFO:tensorflow:loss = 95.1743, step = 1701 (0.338 sec)
INFO:tensorflow:global_step/sec: 251.311
INFO:tensorflow:loss = 157.886, step = 1801 (0.398 sec)
INFO:tensorflow:global_step/sec: 262.811
INFO:tensorflow:loss = 97.9833, step = 1901 (0.380 sec)
INFO:tensorflow:global_step/sec: 293.334
INFO:tensorflow:loss = 47.5423, step = 2001 (0.340 sec)
INFO:tensorflow:global_step/sec: 281.338
INFO:tensorflow:loss = 36.3391, step = 2101 (0.356 sec)
INFO:tensorflow:global_step/sec: 282.133
INFO:tensorflow:loss = 425.897, step = 2201 (0.354 sec)
INFO:tensorflow:global_step/sec: 294.2
INFO:tensorflow:loss = 274.893, step = 2301 (0.343 sec)
INFO:tensorflow:global_step/sec: 275.129
INFO:tensorflow:loss = 176.666, step = 2401 (0.361 sec)
INFO:tensorflow:global_step/sec: 290.16
INFO:tensorflow:loss = 70.1193, step = 2501 (0.345 sec)
INFO:tensorflow:global_step/sec: 283.886
INFO:tensorflow:loss = 227.321, step = 2601 (0.352 sec)
INFO:tensorflow:global_step/sec: 295.2
INFO:tensorflow:loss = 339.602, step = 2701 (0.339 sec)
```

```
INFO:tensorflow:global_step/sec: 298.159
INFO:tensorflow:loss = 56.6977, step = 2801 (0.335 sec)
INFO:tensorflow:global_step/sec: 292.904
INFO:tensorflow:loss = 42.8753, step = 2901 (0.342 sec)
INFO:tensorflow:global_step/sec: 290.429
INFO:tensorflow:loss = 39.4871, step = 3001 (0.343 sec)
INFO:tensorflow:global_step/sec: 273.244
INFO:tensorflow:loss = 37.2008, step = 3101 (0.367 sec)
INFO:tensorflow:global_step/sec: 292.475
INFO:tensorflow:loss = 608.344, step = 3201 (0.342 sec)
INFO:tensorflow:global_step/sec: 266.668
INFO:tensorflow:loss = 38.7863, step = 3301 (0.373 sec)
INFO:tensorflow:global_step/sec: 269.916
INFO:tensorflow:loss = 97.2443, step = 3401 (0.370 sec)
INFO:tensorflow:global_step/sec: 292.047
INFO:tensorflow:loss = 97.0952, step = 3501 (0.343 sec)
INFO:tensorflow:global_step/sec: 260.062
INFO:tensorflow:loss = 41.7583, step = 3601 (0.387 sec)
INFO:tensorflow:global_step/sec: 289.504
INFO:tensorflow:loss = 42.4794, step = 3701 (0.342 sec)
INFO:tensorflow:global_step/sec: 289.085
INFO:tensorflow:loss = 81.957, step = 3801 (0.347 sec)
INFO:tensorflow:global_step/sec: 273.243
INFO:tensorflow:loss = 29.4515, step = 3901 (0.365 sec)
INFO:tensorflow:global_step/sec: 280.938
INFO:tensorflow:loss = 33.5829, step = 4001 (0.357 sec)
INFO:tensorflow:global_step/sec: 299.459
INFO:tensorflow:loss = 103.37, step = 4101 (0.333 sec)
INFO:tensorflow:global_step/sec: 273.995
INFO:tensorflow:loss = 67.7202, step = 4201 (0.364 sec)
INFO:tensorflow:global_step/sec: 280.546
INFO:tensorflow:loss = 71.9568, step = 4301 (0.357 sec)
INFO:tensorflow:global_step/sec: 272.126
INFO:tensorflow:loss = 35.866, step = 4401 (0.368 sec)
INFO:tensorflow:global_step/sec: 269.916
INFO:tensorflow:loss = 83.3985, step = 4501 (0.369 sec)
INFO:tensorflow:global_step/sec: 288.665
INFO:tensorflow:loss = 69.2387, step = 4601 (0.346 sec)
INFO:tensorflow:global_step/sec: 297.713
INFO:tensorflow:loss = 78.3916, step = 4701 (0.337 sec)
INFO:tensorflow:global_step/sec: 261.638
INFO:tensorflow:loss = 41.9503, step = 4801 (0.382 sec)
INFO:tensorflow:global_step/sec: 264.898
INFO:tensorflow:loss = 126.531, step = 4901 (0.377 sec)
INFO:tensorflow:Saving checkpoints for 5000 into C:\Users\DELL\AppData\Local\Temp\tmphd39etm0\mo
INFO:tensorflow:Loss for final step: 58.3278.
```

```
Out[25]: <tensorflow.python.estimator.canned.linear.LinearClassifier at 0x2a34dab748>
```

0.1.4 Evaluation

Create a prediction input function

```
In [26]: pred_fn = tf.estimator.inputs.pandas_input_fn(x=X_test,batch_size=len(X_test),
                                                       shuffle=False)
```

We will use model.predict() and pass in the input function. This will produce a generator of predictions, which we can then transform into a list, with list()

```
In [27]: predictions = list(model.predict(input_fn=pred_fn))
```

```
INFO:tensorflow:Restoring parameters from C:\Users\DELL\AppData\Local\Temp\tmphd39etm0\model.ckp
```

Each item in the list will look like this:

```
In [28]: predictions[0]
```

```
Out[28]: {'class_ids': array([0], dtype=int64),
          'classes': array([b'0'], dtype=object),
          'logistic': array([ 0.37067723], dtype=float32),
          'logits': array([-0.52931267], dtype=float32),
          'probabilities': array([ 0.62932283,  0.3706772 ], dtype=float32)}
```

Create a list of only the class_ids key values from the prediction list of dictionaries, these are the predictions we will use to compare against the real y_test values.

```
In [29]: final_preds = []
         for pred in predictions:
             final_preds.append(pred['class_ids'][0])
```

```
In [30]: final_preds[:10]
```

```
Out[30]: [0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
```

Import classification_report from sklearn.metrics and then see if we can figure out how to use it to easily get a full report of our model's performance on the test data

```
In [31]: from sklearn.metrics import classification_report
```

```
In [52]: print(classification_report(y_test,final_preds))
```

```
             precision    recall  f1-score   support

          0       0.88      0.93      0.91      7436
          1       0.73      0.60      0.66      2333

avg / total       0.85      0.85      0.85      9769
```