In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv('iris.csv')
```

In [3]:
```
df.head()
```

Out[3]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

In [4]:
```
# need to rename columns as spaces etc will not work with the model
df.columns
```

Out[4]:
```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'target'],
      dtype='object')
```

In [5]:
```
df.columns = ['sepal_length', 'sepal_width', 'petal_length', 'petal_width', 'target']
```

In [6]:
```
df.head()
```

Out[6]:

	sepal_length	sepal_width	petal_length	petal_width	target
0	5.1	3.5	1.4	0.2	0.0
1	4.9	3.0	1.4	0.2	0.0
2	4.7	3.2	1.3	0.2	0.0
3	4.6	3.1	1.5	0.2	0.0
4	5.0	3.6	1.4	0.2	0.0

In [7]:
```
df['target'] = df['target'].apply(int)
```

In [8]:
```
df.head()
```

Out[8]:

	sepal_length	sepal_width	petal_length	petal_width	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [13]:
```
y = df['target']
```

In [14]:
```
X = df.drop('target', axis=1)
```

In [15]:
```
from sklearn.model_selection import train_test_split
```

In [16]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_
state=42)
```

In [17]:
```
import tensorflow as tf
```

In [18]:
```
# Feature Columns

X.columns
```

Out[18]:
```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width'], dtype='obj
ect')
```

In [19]:
```
feat_cols = []

for col in X.columns:
    feat_cols.append(tf.feature_column.numeric_column(col))
```

In [20]:
```
feat_cols
```

Out[20]:
```
[NumericColumn(key='sepal_length', shape=(1,), default_value=None, dtype=tf.floa
t32, normalizer_fn=None),
 NumericColumn(key='sepal_width', shape=(1,), default_value=None, dtype=tf.float
32, normalizer_fn=None),
 NumericColumn(key='petal_length', shape=(1,), default_value=None, dtype=tf.floa
t32, normalizer_fn=None),
 NumericColumn(key='petal_width', shape=(1,), default_value=None, dtype=tf.float
32, normalizer_fn=None)]
```

In [158]:
```
#input
input_func = tf.estimator.inputs.pandas_input_fn(x=X_train,
                                                 y=y_train,
                                                 batch_size=10,
                                                 num_epochs=5,
                                                 shuffle=True)
```

In [159]:
```
#Classifier

classifier = tf.estimator.DNNClassifier(hidden_units = [10,20,10],
                                        n_classes=3,
                                        feature_columns = feat_cols)
```

```
W0827 13:38:06.590025 140736573572032 estimator.py:1811] Using temporary folder
as model directory: /var/folders/bg/2b17ybm53nz575_6xtq1zxnh0000gn/T/tmp3qs2_xpx
```

In [160]:
```
classifier.train(input_fn=input_func,steps=50)
```

Out[160]:
```
<tensorflow_estimator.python.estimator.canned.dnn.DNNClassifier at 0x1a4065a410>
```

In [162]:
```
pred_fn = tf.estimator.inputs.pandas_input_fn(x=X_test,
                                             batch_size=len(X_test),
                                            shuffle=False)
```

In [163]:
```
predictions = list(classifier.predict(input_fn=pred_fn))
```

In [164]:

```
predictions[0]
```

Out[164]:

```
{'logits': array([-2.6655953,  1.5381429,  1.463346 ], dtype=float32),
 'probabilities': array([0.00768946, 0.5147021 , 0.47760847], dtype=float32),
 'class_ids': array([1]),
 'classes': array([b'1'], dtype=object),
 'all_class_ids': array([0, 1, 2], dtype=int32),
 'all_classes': array([b'0', b'1', b'2'], dtype=object)}
```

In [165]:

```
final_preds = []

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

In [166]:

```
#Final_Preds
```

In [167]:

```
from sklearn.metrics import classification_report, confusion_matrix
```

In [168]:

```
print(confusion_matrix(y_test, final_preds))
```

```
[[19  0  0]
 [ 0 10  3]
 [ 0  0 13]]
```

In [169]:

```
print(classification_report(y_test,final_preds))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        19
           1       1.00      0.77      0.87        13
           2       0.81      1.00      0.90        13

    accuracy                           0.93        45
   macro avg       0.94      0.92      0.92        45
weighted avg       0.95      0.93      0.93        45
```

In [ ]:

```

```