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

In [2]:
```
data = pd.read_csv('bank_note_data.csv')
```

Check the head of the Data

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

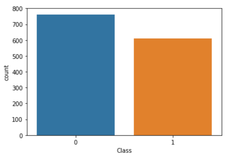
Out[3]:

	Image.Var	Image.Skew	Image.Curt	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

In [4]:
```
import seaborn as sns
%matplotlib inline
```

In [5]:
```
sns.countplot(x='Class',data=data)
```

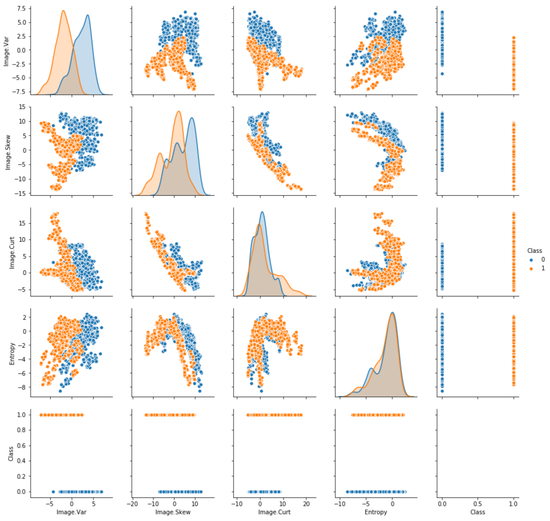
Out[5]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x26bb34edda0>
```



In [6]:
```
sns.pairplot(data,hue='Class')
```

```
C:\Users\Marcial\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:494: RuntimeWarning: invalid value encountered in true_divide
  binned = fast_linbin(X,a,b,gridsize)/(delta*nobs)
C:\Users\Marcial\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.py:34: RuntimeWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
C:\Users\Marcial\Anaconda3\lib\site-packages\numpy\core\_methods.py:26: RuntimeWarning: invalid value encountered in reduce
  return umr_maximum(a, axis, None, out, keepdims)
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x26bb34c9da0>
```



In [7]:
```
from sklearn.preprocessing import StandardScaler
```

In [8]:
```
scaler = StandardScaler()
```

In [9]:
```
scaler.fit(data.drop('Class',axis=1))
```

Out[9]:
```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [10]:
```
scaled_features = scaler.fit_transform(data.drop('Class',axis=1))
```

In [11]:
```
df_feat = pd.DataFrame(scaled_features,columns=data.columns[:-1])
df_feat.head()
```

Out[11]:

	Image.Var	Image.Skew	Image.Curt	Entropy
0	1.121806	1.149455	-0.975970	0.354561
1	1.447066	1.064453	-0.895036	-0.128767
2	1.207810	-0.777352	0.122218	0.618073
3	1.063742	1.295478	-1.255397	-1.144029
4	-0.036772	-1.087038	0.736730	0.096587

In [12]:
```
X = df_feat
```

In [13]:
```
y = data['Class']
```

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

In [16]:
```
import tensorflow as tf
```

```
C:\Users\Marcial\Anaconda3\lib\site-packages\h5py\__init__.py:34: FutureWarning: Conversion of the second argument of issubdtype from `float` to `np.floating` is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

In [17]:
```
df_feat.columns
```

Out[17]:
```
Index(['Image.Var', 'Image.Skew', 'Image.Curt', 'Entropy'], dtype='object')
```

In [18]:
```
image_var = tf.feature_column.numeric_column("Image.Var")
image_skew = tf.feature_column.numeric_column('Image.Skew')
image_curt = tf.feature_column.numeric_column('Image.Curt')
entropy =tf.feature_column.numeric_column('Entropy')
```

In [19]:
```
feat_cols = [image_var,image_skew,image_curt,entropy]
```

In [20]:
```
classifier = tf.estimator.DNNClassifier(hidden_units=[10, 20, 10], n_classes=2,feature_columns=feat_cols)
```

```
INFO:tensorflow:Using default config.
WARNING:tensorflow:Using temporary folder as model directory: C:\Users\Marcial\AppData\Local\Temp\tmpw8v7z_z6
INFO:tensorflow:Using config: {'_model_dir': 'C:\\Users\\Marcial\\AppData\\Local\\Temp\\tmpw8v7z_z6', '_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_config': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_train_distribute': None, '_device_fn': None, '_service': None, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec object at 0x000026BBA0F9FD0>, '_task_type': 'worker', '_task_id': 0, '_global_id_in_cluster': 0, '_master': '', '_evaluation_master': '', '_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
```

In [21]:
```
input_func = tf.estimator.inputs.pandas_input_fn(x=X_train,y=y_train,batch_size=20,shuffle=True)
```

In [22]:
```
classifier.train(input_fn=input_func,steps=500)
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Create CheckpointSaverHook.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
INFO:tensorflow:Saving checkpoints for 0 into C:\Users\Marcial\AppData\Local\Temp\tmpw8v7z_z6\model.ckpt.
INFO:tensorflow:loss = 13.792015, step = 1
INFO:tensorflow:Saving checkpoints for 48 into C:\Users\Marcial\AppData\Local\Temp\tmpw8v7z_z6\model.ckpt.
INFO:tensorflow:Loss for final step: 0.47980386.
```

Out[22]:
```
<tensorflow.python.estimator.canned.dnn.DNNClassifier at 0x26bbacc7c18>
```

Model Evaluation

In [23]:
```
pred_fn = tf.estimator.inputs.pandas_input_fn(x=X_test,batch_size=len(X_test),shuffle=False)
```

In [24]:
```
note_predictions = list(classifier.predict(input_fn=pred_fn))
```

```
INFO:tensorflow:Calling model_fn.
INFO:tensorflow:Done calling model_fn.
INFO:tensorflow:Graph was finalized.
INFO:tensorflow:Restoring parameters from C:\Users\Marcial\AppData\Local\Temp\tmpw8v7z_z6\model.ckpt-48
INFO:tensorflow:Running local_init_op.
INFO:tensorflow:Done running local_init_op.
```

In [25]:
```
note_predictions[0]
```

Out[25]:
```
{'class_ids': array([0], dtype=int64),
 'classes': array([b'0'], dtype=object),
 'logistic': array([0.00157453], dtype=float32),
 'logits': array([-6.4522204], dtype=float32),
 'probabilities': array([0.9984255 , 0.00157453], dtype=float32)}
```

In [26]:
```
final_preds  = []
for pred in note_predictions:
    final_preds.append(pred['class_ids'][0])
```

In [27]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [28]:
```
print(confusion_matrix(y_test,final_preds))
```

```
[[213   2]
 [ 10 187]]
```

In [29]:
```
print(classification_report(y_test,final_preds))
```

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97       215
           1       0.99      0.95      0.97       197

avg / total       0.97      0.97      0.97       412
```

In [30]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [31]:
```
rfc = RandomForestClassifier(n_estimators=200)
```

In [32]:
```
rfc.fit(X_train,y_train)
```

Out[32]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_split=1e-07, min_samples_leaf=1,
            min_samples_split=2, min_weight_fraction_leaf=0.0,
            n_estimators=200, n_jobs=1, oob_score=False, random_state=None,
            verbose=0, warm_start=False)
```

In [33]:
```
rfc_preds = rfc.predict(X_test)
```

In [34]:
```
print(classification_report(y_test,rfc_preds))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       215
           1       0.99      0.99      0.99       197

avg / total       0.99      0.99      0.99       412
```

In [35]:
```
print(confusion_matrix(y_test,rfc_preds))
```

```
[[214   1]
 [  2 195]]
```

It should have also done very well, possibly perfect! Hopefully you have seen the power of DNN!